3.1　サーバ仮想化・クライアント仮想化のアーキテクチャ

最初にサーバ仮想化およびクライアント仮想化のアーキテクチャを見ていきます。サーバとクライアントの仮想化の仕組みを知ることは、仮想化全般を理解するうえでとても重要なことです。

3.1.1　ホストOS型・ハイパーバイザ型

これまでにVMM（Virtual Machine Monitor：仮想マシンモニタ）という用語は何度が登場してきましたが改めて説明しますと、1台のコンピュータに複数の仮想マシンを作成して動作させるためのソフトウェアのことを指します。そのVMMを動作させるための方法として大きく分けると2つの方式があります。ホストOS型とハイパーバイザ型です。2つの大きな違いは、ホストOS型はアプリケーションとして動作しますが、ハイパーバイザ型はOSとして動作します。ここでは、各々の違いを見ていきます。

■ ホストOS型

ハードウェアに直接インストールされたOSのことをホストOSと呼びます。一般にコンピュータでは、WindowsやLinuxをハードウェアに直接インストールして利用しますが、このWindowsやLinuxのことをホストOSと呼んでいます。ホストOS型とは、VMMがホストOS上で動作する方式のことを指します。ホストOS上で動作するとは、アプリケーションであるという意味であり、ホストOSの一つのプロセスとして動作します。VMMには、他のアプリケーションと同じようにタスクが起動されメモリとCPUが割り当てられます。

VMM上では、仮想マシンが動作します。仮想マシンの中で動作するOSのことをゲストOSと呼んでいます。

図3.1－1にホストOS型の構成を示します。VMMはホストOS上で動作して他のアプリケーションと同様に動作します。VMMには2つの仮想マシンが作られています。仮想マシンのOS上で動作するアプリケーションからハードウェアにアクセスする場合、ゲストOS→VMM→ホストOSと複数のOSやアプリケーションを経由する必要があるため、オーバーヘッドが大きくなることが分かります。ホストOS型の欠点はこのようにオーバーヘッドが大きく性能に影響しているということです。また、もしホストOSにトラブルが発生すると、全ての仮想マシンに影響を与えてしまうことになります。

ただし、ホストOS型は、コンピュータ上で動作しているOSを変更することなく、さらにそのOSで動作するアプリケーションも変更することなく利用できることなどから、手軽にインストールして利用できることが利点といえます。ホストOS型は、主にクライアント仮想化に利用されている方式です。

ホストOS型のVMMは、Windows版やLinux版など、インストールするホストOSに対応したものを選定する必要があります。

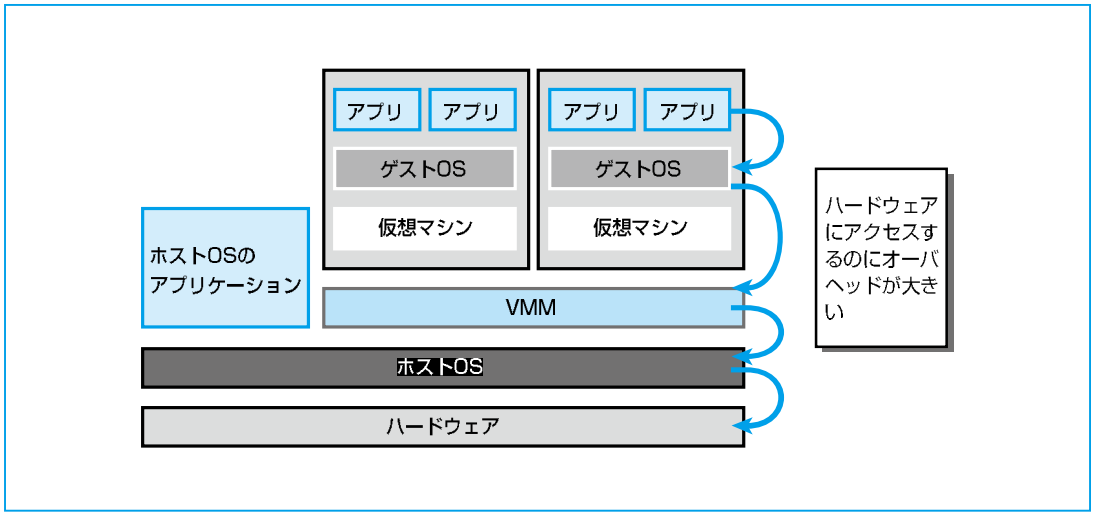

図3.1－1　ホストOS型の構成

ホストOS型を実際に構成した例を示します。図3.1－2に具体的な構成例を、さらにその構成で動作させたコンピュータ上の画面を画面3.1－1に示します。

この構成はホストOSにWindows XPを、VMMにOracle VM VirtualBox（以降VirtualBox）を使っています。VMM上には2つの仮想マシンが作られ、それぞれWindows 7とWindows Server 2003が動作しています。

VMMはWndowsXPのアプリケーションの一つですので、Windows XP上で直接動作している他のアプリケーションと切り替えながら動作することになります。

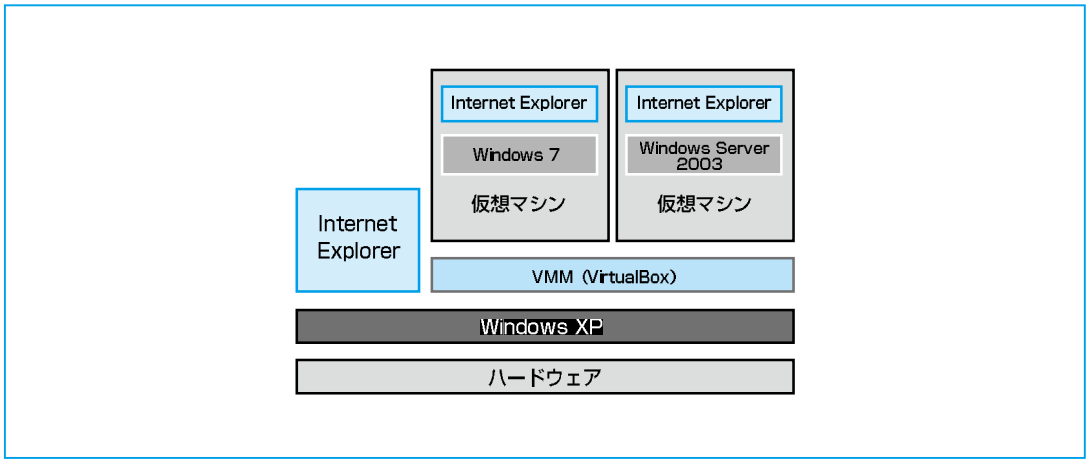

図3.1－2　ホストOS型の構成例

画面3.1－1には3つのウインドウが開いています。一つはホストOSであるWindows XPのIEウインドウ（画面①）です。二つ目は、ゲストOSのWindows 7が動作しているウインドウ（画面②）の画面です。そのウインドウ内にはWindows 7上で動作するIEが確認できます。さらに三つ目は、ゲストOSのWindows Server 2003が動作しているウインドウ（画面③）の画面です。そのウインドウ内にはWindows Server 2003上で動作するIEが確認できます。VMMであるVirtualBoxのウインドウは最少化されておりますので表示はされていません。

NOTE

Oracle VM VirtualBox：Oracle VM VirtualBoxは、クライアント向けのホストOS型の仮想化ソフトウェアです。当初はドイツのInnotek社が開発を行い、Sun Microsystems社が引き継ぎ、現在は同社を買収したOracle社からリリースされています。サポートするホストOSにはLinux、Mac OS、Windowsなど幅広くサポートしています。ゲストOSもLinux、OS/2、Windows、Solarisなどが含まれます。

このように、動作する仮想マシンはホストOS上のウインドウとして表示されます。
少し複雑ですので、利用時はウインドウを間違わないようにして下さい。

画面3.1－1　ホストOS型の画面例

■ ハイパーバイザ型

ハイパーバイザ型は、VMMを直接ハードウェアにインストールして利用します。VMMがホストOSの代わりをしますので、VMM自体がOSといえます。ホストOS型と比べると、ホストOSが無くなる分オーバーヘッドが少なくなるのが特徴です。ハイパーバイザ型は主にサーバ仮想化で利用されています。

ハイパーバイザ型の種類はさらに2つに分かれます。モノリシックカーネル型とマイクロカーネル型です。次にこの2つの違いを見ていきます。

モノリシックカーネル型

モノリシックカーネル型はVMMの中にデバイスドライバを持っています。ホストOS型のデバイスドライバは汎用OS内にありますので多くのハードウェアに対応していますが、モノリシックカーネル型のデバイスドライバは比較的限られたハードウェアにしか対応していないため、動作しないものもあります。ただ、VMMとデバイスドライバは密接に連携して動作するため、性能は高いものになります。

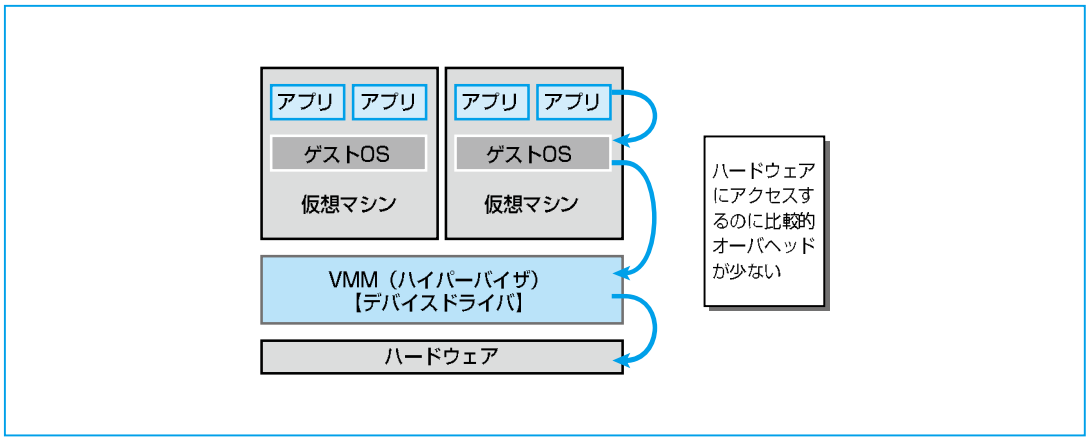

図3.1－3　ハイパーバイザ（モノリシックカーネル）型

マイクロカーネル型

マイクロカーネル型は、管理OSといわれるOSが仮想マシンの一つとして動作し、デバイスドライバはその管理OS中にあります。仮想マシンからハードウェアにアクセスしたい場合は、一旦管理OSを経由して行われますので、モノリシックカーネル型より性能は落ちてしまいます。管理OSは汎用OSを利用する場合もあり、デバイスドライバが対応するハードウェアも多くなっています。

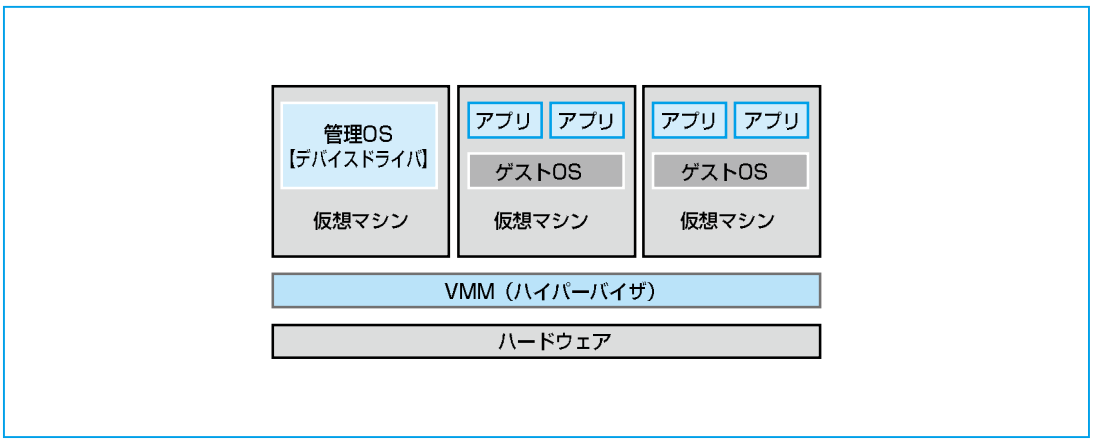

図3.1－4　ハイパーバイザ（マイクロカーネル）型

マイクロカーネル型の動作をもう少し詳しく見ていきます。図3.1－5に示すように、管理OSは汎用OSを使う場合も多くVMM上で仮想マシン①として動作しています。仮想マシンからハードウェアにアクセスする場合は一旦管理OSを経由してデバイスドライバ経由でハードウェアをアクセスします。仮想マシン②上で動作しているアプリケーションが、ハードウェアのあるデバイスにアクセスする必要がある場合、ゲストOSにアクセスが依頼されます。ゲストOSでは仮想デバイスドライバを経由しVMMに渡されます。VMMは管理OSにその依頼を渡し、管理OS上のデバイスドライバを利用してハードウェア上のデバイスにアクセスします。

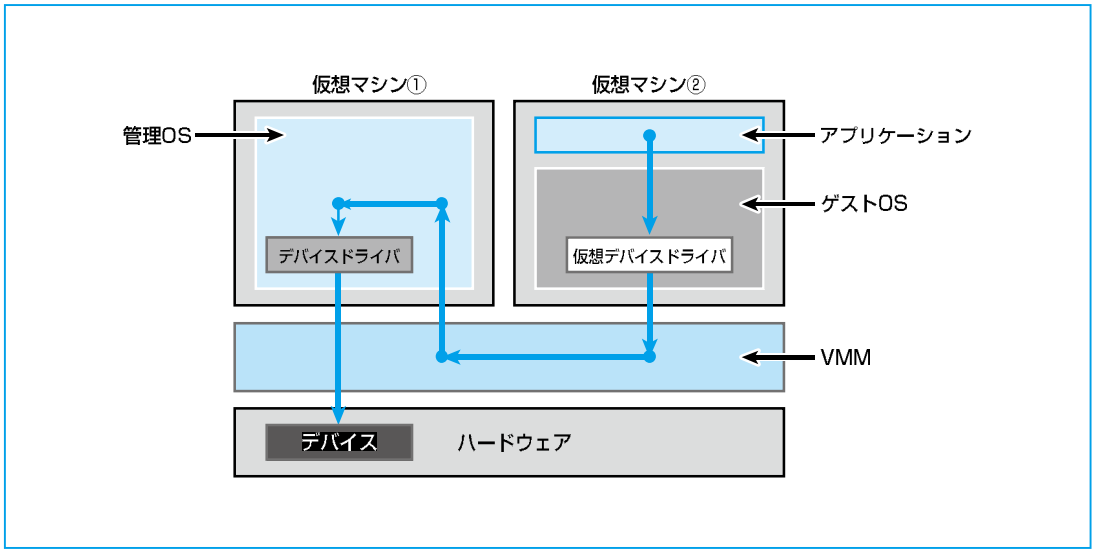

図3.1－5　ハイパーバイザ（マイクロカーネル）型の動作

■ 完全仮想化と準仮想化

前述の説明のようにホストOS型とハイパーバイザ型とを比較すると性能的にはハイパーバイザ型が有利ですが、さらにこのハイパーバイザ型の中でも性能に違いがある「完全仮想化」と「準仮想化」の2つの方式に分かれます。

完全仮想化とはハイパーバイザ上で動作するゲストOSの中身に何も変更を加えないで動作させる方式を指します。他方、準仮想化とはゲストOSに変更を加えて性能向上を図った方式を指します。具体的には、OSがハードウェアにアクセスする際、ハイパーバイザに処理が遷移するようにプログラムに変更を加えます。

完全仮想化では、ゲストOSから直接ハードウェアをアクセスすると特権違反の割り込みが発生してハイパーバイザ内の例外処理に遷移しますが、その部分をハイパーバイザに直接処理が渡るように変更を加えるのが準仮想化です。完全仮想化ではクリティカル命令を変換するバイナリトランスレーションで動的に処理しますので性能が極端に落ちますが、準仮想化では静的に変換を行い予め修正しますので性能には影響しません。このように、予めゲストOSに手を加えることにより性能アップを図れるのが準仮想化の特徴になります。

準仮想化はLinuxのようなオープンソースとなっているOSであれば修正が可能ですが、Windowsのようにソースファイルに手を加えることができないものがあります。そのようなOSは完全仮想化に対応するものでしか利用できないことになります。

実際には完全仮想化もCPUの仮想化支援機能を利用することにより、実用上問題のない性能を提供できるようになりました。

図3.1－6に完全仮想化と準仮想化の処理の流れを示します。

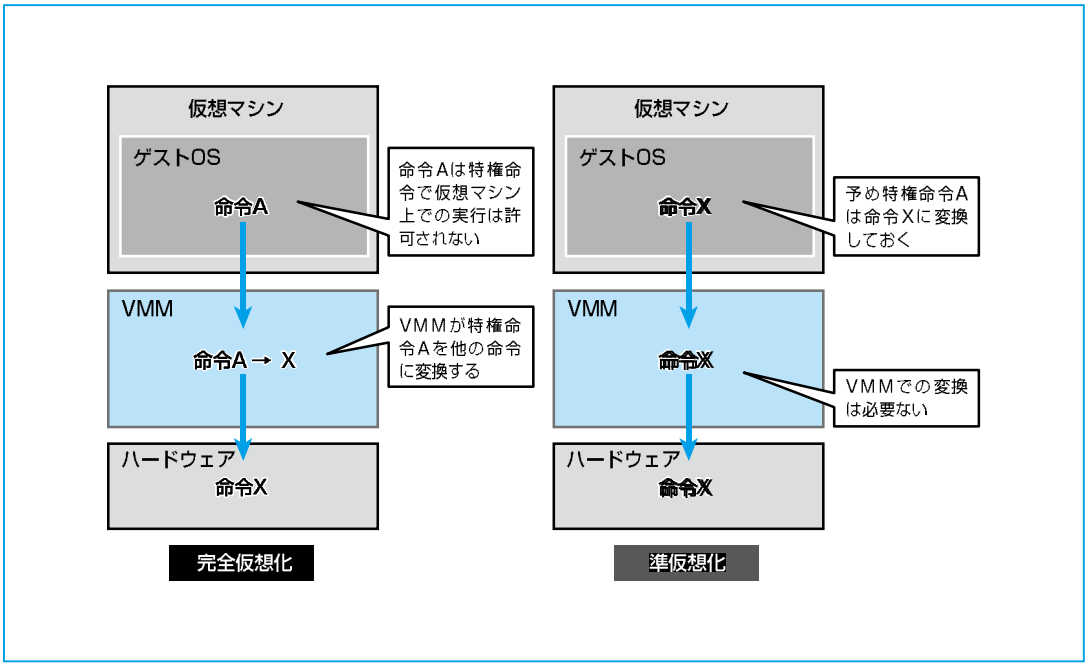

図3.1－6　完全仮想化と準仮想化の処理の流れ