仮想化技術の進展

小畑 拓也，鍵谷 武宏
Takuya KOBATA,Takehiro KAGITANI

1　はじめに

近年，CPU の高性能化により，CPU 稼動率の低さが顕著になってきている．通常，稼働率はピーク時に備えて設定されるため，平均するとほとんど稼動していない．また，IT システムの高度化・複雑化により，サーバの需要が大幅に伸びているが，台数の増加に伴って設置スペースが限界に達してきている．そこで，この 2 つの問題を解決する手法として，複数のサーバを 1 台にまとめることのできる，仮想化技術に注目が集まっている．本稿では仮想化技術をまとめ，今後の展望について述べる．

2　仮想化技術

仮想化技術とは，プロセッサやメモリ，ディスク，通信回線など，コンピュータシステムを構築する資源を物理的構成に拠らず柔軟に分割，および統合する技術である．Fig. 1 に仮想化技術の分類を示す．

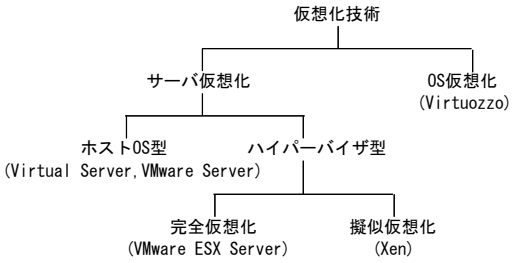

Fig.1　仮想化技術の分類 (出典：自作)

3　サーバ仮想化

サーバ仮想化とは，CPU リソースの利用を最適化する手法である．このサーバ仮想化により，1 台のサーバを複数の仮想的なサーバに分割し各 OS 上でアプリケーションを動作させる，あるいは複数のサーバで 1 つのアプリケーションを動作させることができる．サーバ仮想化は，大きく分けてホスト OS 型とハイパーバイザ型の 2 種類のアーキテクチャに分類することができる．

3.1　ホスト OS 型

ホスト OS 型とは，通常の OS 上に仮想化ソフトウェアをアプリケーションのようにインストールし，その上でいくつかの OS を動作させる方式である．Fig. 2 にホスト OS 型の構造を示す．この方式では，アプリケーションをインストールするだけで仮想マシンを作成できるため，特別なドライバが不要であり非常に導入しやすい．また，ハードウェアによる依存も少ないため，ハードウェアの選択肢の幅が広くなる．しかし，ホスト OS が仮想マシンに特化していない汎用 OS であるため，性能が下がり機能が制限される．

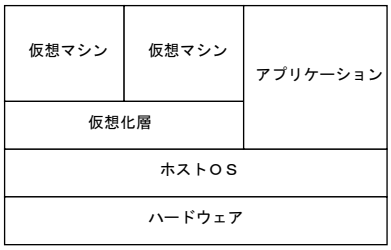

Fig.2　ホスト OS 型の構造 ([1] より参照)

3.2　ハイパーバイザ型

ハイパーバイザ型とは，ハイパーバイザと呼ばれる，仮想マシンの実行に特化した一種の専用カーネル上でいくつかの OS を動作させる方式である．また，ハイパーバイザ型を構成する仕組みは，完全仮想化と擬似仮想化に分類することができる．

3.2.1　完全仮想化

完全仮想化とは，ハードウェアの上にホスト OS を介在させずに仮想化層の仕組みを実装し，仮想化層がハードウェアと直接制御を行う方式である．この方式では，Fig. 3 に示すように，仮想化層がハードウェア上で直接稼動しホスト OS として機能するため，Windows や Linux といった一般の OS を必要としない．また，どんな OS でも修正せずに仮想マシン上にインストールすることができる．インストールされたゲスト OS は，仮想化層が CPU 命令に割り込んで処理を行うため，自らが仮想環境であることを意識せずに稼動することができる．しかし，ゲスト OS は直接ハードウェアの操作を行うことができないため，オーバーヘッドが生じる．

Fig.3　完全仮想化の構造 ([1] より参照)

3.2.2　擬似仮想化

擬似仮想化とは，ゲスト OS に手を加え，仮想化環境を認識させるようにした方式である．Fig. 4 に擬似仮想化の構造を示す．擬似仮想化では OS そのものを書き換えることで，仮想化層で命令を変換せず，ゲスト OS から直接ハードウェアの操作を行うことができる．しかし，OS を修正したため，OS に対するセキュリティ修正プログラムの適用を行うことができず，OS のソースコード修正を統合的に管理できるツールや管理ソリューションはまだ充実していない．また，専用のドライバが必要になるため，対応ハードウェアが限定され，OS そのものの書き換えも必要であるため，オープンソースとしてソースコードが公開されている OS には対応できるが，修正を加えることができない OS には対応できない．

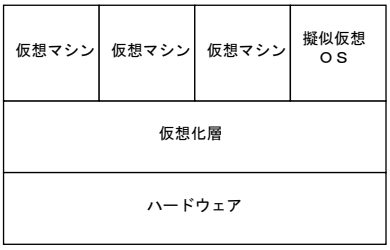

Fig.4　擬似仮想化の構造 ([1] より参照)

4　OS 仮想化

OS 仮想化とは，OS を仮想化することにより，仮想環境を作り出す技術である．1 つのホスト OS を複数の OS にみせかけて使うことで，アプリケーションごとに CPU，メモリ，I/O などのリソースを割り当て，1 台のサーバ上で多数のシステムを稼動することができる．

4.1　OS 仮想化の構成

OS 仮想化は，カーネル 1 つでシステム構成のファイルシステムを複数所持することができる．これは，それぞれのファイルシステムにおいて，仮想環境単位で chroot 機能を利用しているためである．chroot 機能とは，change root 機能のことで，アプリケーションを root ファイルシステムや他のアプリケーションから隔離し，アプリケーションの競合を防ぐ．そのため，仮想サーバごとの再起動も可能である．また，chroot-Barrier 機能という，ルートディレクトリから親にエスケープしないようにするための機能も持っている．

4.2　OS 仮想化の資源管理

OS 仮想化は，Two-level disk quota，CPU scheduler，User Beancounters の 3 つの要素によって資源管理を構成している．Two-level disk quota では，ブロック割当てによって使用することのできるファイルとディレクトリの数を制限し，i ノード割当てによって作成することのできるディスク空間の量を制限し，ディスクの限界を設定する．次に，CPU scheduler では，初期段階で CPU を取得する仮想環境が決定し，第 2 段階で標準 Linux プロセス優先権に基づいて実行するプロセスを選択する．最後に，User Beancounters では，特定の仮想環境のリソース配分を定義する多数のパラメータを決定し，仮想環境に対する制御レベルを指定する．

4.3　OS 仮想化の特徴

OS 仮想化は，資源管理によって，リソースの利用率を設定し，物理リソースを論理的に分割することができる．また，隔離された仮想環境を単一の OS とハードウェア上に作成しているため，ハードウェアやソフトウェアの管理における効率を最大化することができる．さらに，仮想環境における OS はホスト OS と同じ主要部分を共有するため，完全仮想化や擬似仮想化の仮想マシンのサイズよりも大幅に小さくなり，オーバーヘッドも少なくなる．Fig. 5 に OS 仮想化の構造を示す．

Fig.5　OS 仮想化の構造 ([2] より参照)

5　今後の仮想化技術

現在，仮想化技術の中で最も性能の良い擬似仮想化が注目を集めている．擬似仮想化は，専用のドライバを用い OS を書き換えることで，OS 仮想化より劣るものの完全仮想化よりも高速処理を行うことができる．また，仮想マシンを停止させず動作ハードウェアを変更することのできるライブマイグレーションが搭載されており，メンテナンスを容易に行うことができる．しかし，最近ではライブマイグレーションを搭載した OS 仮想化のソフトウェアも開発されるなど，OS 仮想化の技術に進展が見られる．一方で，擬似仮想化を対象に，新たに CPU に仮想化支援技術が搭載されるようになってきたため，擬似仮想化を導入する際の負担が軽減され，性能もさらに向上してきた．つまり，擬似仮想化は OS 仮想化と同等程度の処理速度を発揮できるようになっている．ここで，OS 仮想化はあまり複数の OS を使用せず管理に重点を置くニーズに対して，擬似仮想化は管理よりも複数の OS を使用するニーズに対して存在感を増してくると考えられる．しかし，今後も IT システムが高度化・複雑化を遂げると，擬似仮想化で複数のゲスト OS を持つサーバ群を OS ごとに OS 仮想化でまとめたサーバ群に変更したほうが，性能は変わらず，管理をより容易に行えるだろう．しかし，OS 仮想化が擬似仮想化をまとめるためには，対応する OS の拡大が必要となり，近い将来では用途や規模によって仮想化の使い分けが行われるだろう．

参考文献

1) 宮原徹，SoftwareDesign 2007 5 月号，技術評論社，2007
2) 高橋洋介，仮想化技術 Expert，技術評論社，2007