1.1　仮想化とは

仮想化をひとことで言いますと、「ハードウェア仕様からの解放」と表現できます。ハードウェアは既に組み込まれた基本的に目に見える物であり、ハードウェア仕様はハードウェアを直接扱う上では変更をすることができないものです。例えば、コンピュータで利用するメインメモリがありますが、ハードウェア仕様で2Gバイト搭載したコンピュータは、2Gバイト以上の容量のメモリを利用することは決してできません。即ちメモリ容量というハードウェア仕様で表された通りの利用しかできないのです。仮想化とは、そのハードウェアの仕様を解放して、もっと自由に利用できるようにしようというものです。例えば、仮想メモリという考え方があります。ハードディスクの一部をメモリのように利用して、メインメモリを2Gバイト搭載しても、それ以上のメモリがあるかのごとく動作できるというものです。OSやCPUの機能を使って仮想メモリを実現していますが、まさしくメモリというハードウェアの仕様（2Gバイト）から解放され、ハードディスクの容量が許す限りメモリを利用できる仮想化の一例といえます。（図1.1－1参照）

ハードウェア仕様から解放されることにより、1つのハードウェアが2つや3つに分割して利用できるようになったり、逆に複数あったハードウェアを1つに見せかけたりすることが可能になりました。仮想メモリの例では、メモリとハードディスクを合わせて1つに見せて、アプリケーションソフトウェアはメインメモリを使っているのかハードディスクを使っているのか全く意識する必要がなくなりました。

仮想化はハードウェアを、利用側に見せる内容を変えたり実際のハードウェアを隠蔽したりする特徴があります。

この例での仮想化の特徴をまとめておきます。

- 種類の異なる同様の機能をもったハードウェアを1つのまとまった機能として利用することができます。
- ハードウェアを見る視点によりハードウェア仕様とは違うように見せることが可能です。

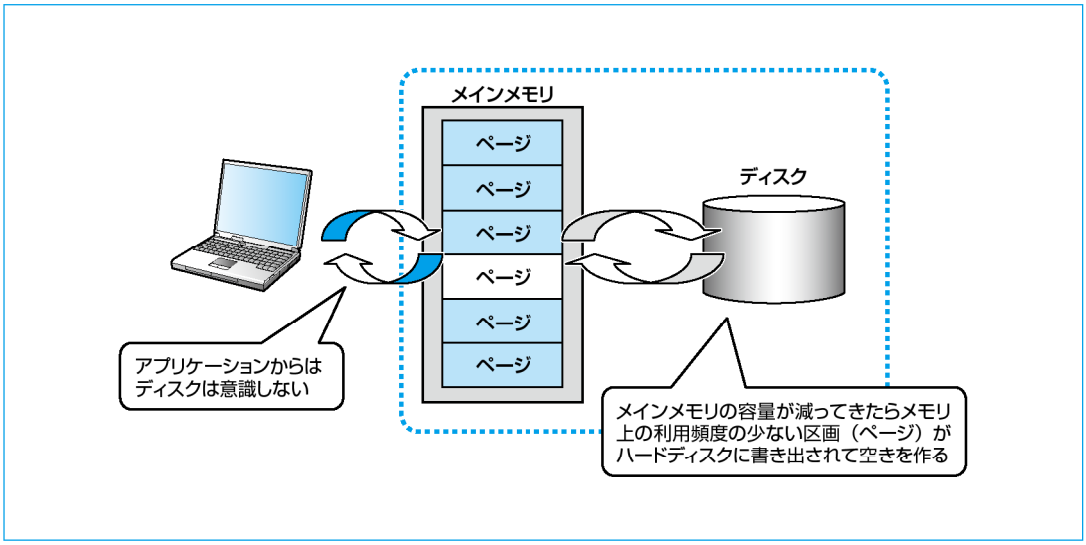

図1.1－1　仮想メモリのしくみ

1.1.1　RAIDの例

別の例を見てみます。RAIDの構成にはいくつかの種類がありますが比較的分かりやすいRAID-1を例にとってみます。RAID-1はハードディスクを2台使い同じデータをその2台に書き込みます。どちらかのディスクが壊れても、もう片方のディスクを使って処理を継続させることができる冗長化を実現する方法であり、一般的にミラーリングと呼んでいる技術です。RAID-1を実現する方法は2つあります。ハードウェア（RAIDコントローラ）を使って構成する方法とOSの機能としてRAID-1を動作させる方法です。

図1.1－2の「ハードウェアでRAID-1を構成」をみますと、ハードウェアでRAID-1の機能を実現しており、OSやアプリケーションはあたかも1台のディスクとしてしか認識されません。ハードウェアがディスクを仮想化しているといえます。それに対して図1.1－2の「ソフトウェアでRAID-1を構成」を見ますと、OSがRAID-1を実現している訳ですからOSは2台のディスクを認識する必要があります。すなわち、ディスクにデータを書き込む場合は2台のディスクに対して書き込みを行い、片方のディスクが壊れた場合は片方からのデータを読み込むなどの仕組みをOSが持つ必要があるわけです。しかし、アプリケーションにはハードウェアで構成していてもOSで実現していても1台のディスクにしか見せないようにしています。このように、仮想化を行う方法は複数あることがわかります。

この例での仮想化の特徴をまとめておきます。

- ハードウェアに付加機能を載せても、利用する側はその機能を意識することなく利用することが可能です。
- 仮想化の実現は、ハードウェアやOSなどのソフトウェアなどを利用して行うことが可能で、複数の実現方法があります。

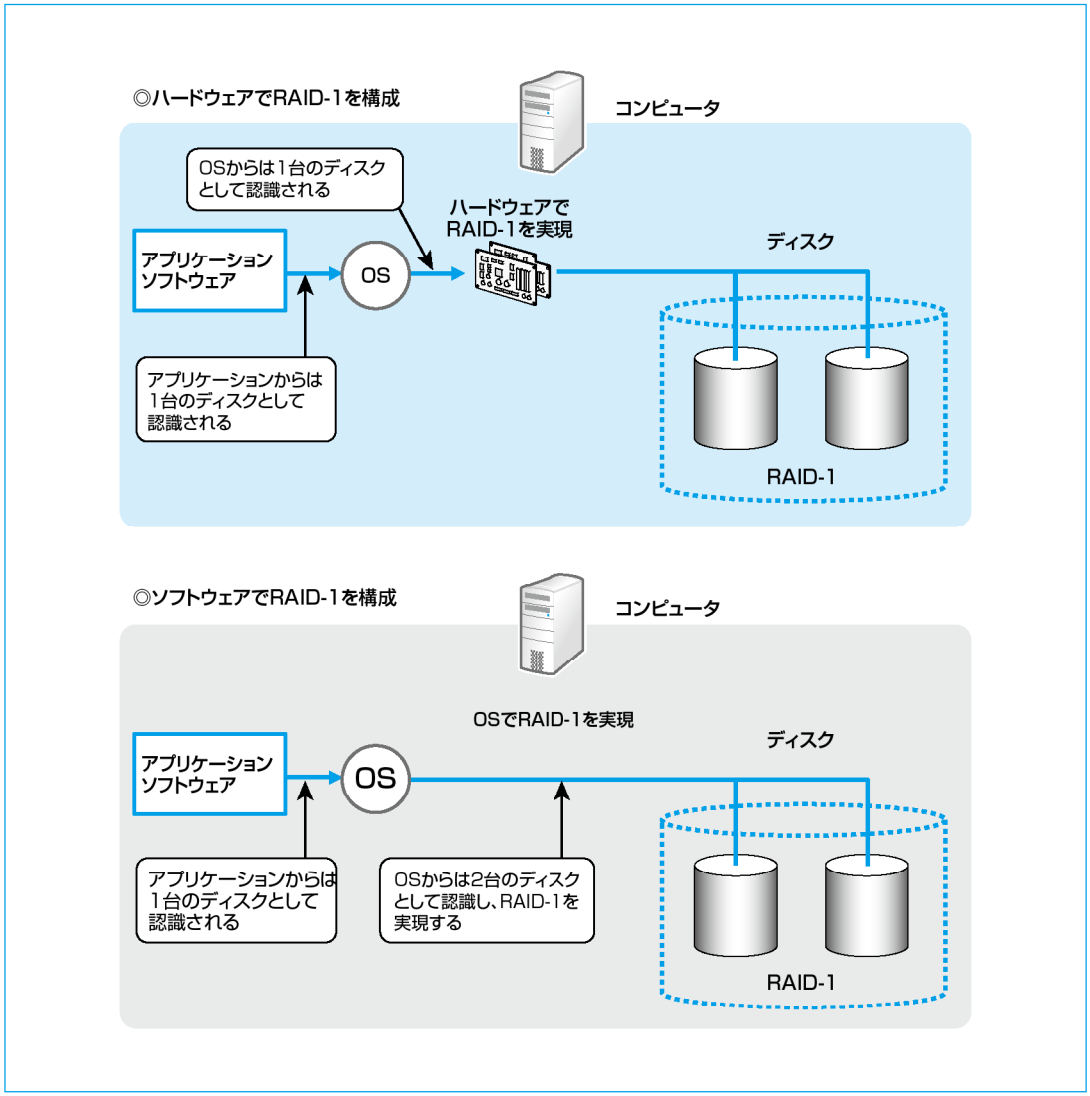

図1.1－2　RAID-1の仕組み

1.1.2 JavaVMの例

Java仮想マシン（JVM：Java Virtual Machine）の例を見ます。JVMはJava言語で開発されたプログラムを動作させるためのプラットフォームです。Java言語で開発されたプログラムの特徴はどのようなOS上でも、どのようなハードウェア（CPU）上でも動作するというものです。ここでは、どのようなハードウェア（CPU）上でも動作する点を見ていきます。C言語のような一般的な開発言語で開発されたプログラムはコンパイルを行うことによりCPUが解釈できる命令に変換されます。この命令はCPUの種類によって変わるもので、異なる種類のCPUでは互換性がありません。それでは、なぜJava言語で開発されたプログラムが異なるCPUで動作できるかといいますと、Javaをコンパイルするとcpu依存の命令に変換されるのではなくバイトコードと呼ばれるCPU非依存の中間コードに変換されるからです。JVMはこのバイトコードを解析してCPUに依存する命令に変換していくという仕組みです。JVMは、CPUというハードウェアを隠蔽してJavaプログラムを動作させており仮想化の1つといえます。

図1.1－3にJVMの仕組みを示します。コンピュータAとコンピュータBは、異なるハードウェア（CPU）で構成されています。両方のコンピュータで動作するJavaのバイトコードは同じですが、JVMはそれぞれのハードウェア用のものを使います。JVMはバイトコードをコンピュータAであればCPU-Aに、コンピュータBであればCPU-Bに解読して実行させます。

Javaのバイトコードは、どのようなハードウェアでもJVMにより仮想化して動作させることが可能です。

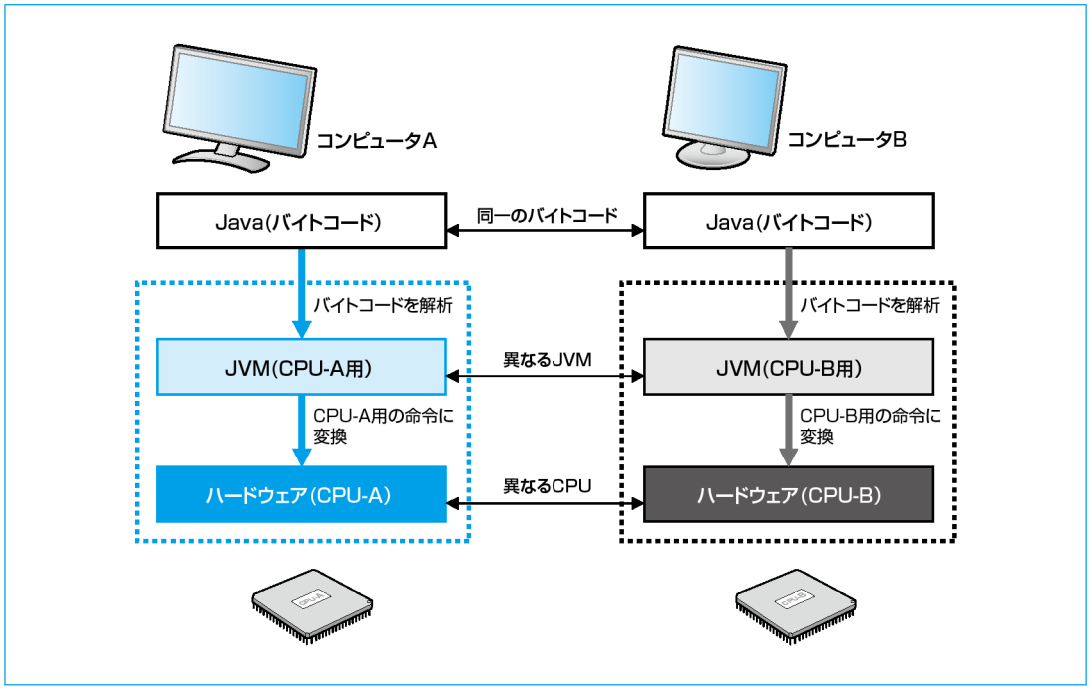

図1.1－3　JVMの仕組み

この仕組みは、インターネット上でゲームなどの動的コンテンツを配信する場合にとても便利な仕組みです。受信するコンピュータはWindows、MACさらにLinuxなどOSやハードウェアの異なるものがあります。全てのコンピュータに対応したプログラムを配信することは不可能ですが、各コンピュータにそのハードウェアに対応するJVMをインストールしておく事で、Javaのプログラムを動作させることができます。

この例での仮想化の特徴をまとめておきます。

- 仮想化により、同じプログラムを異なるOSやハードウェア上で動作させることが可能となります。

これまで出てきた例でもう少し仮想化の特徴をまとめておきます。

- ハードウェアの内容をアプリケーションに見せないようにすることで、ソフトウェアの開発の手間を省くことができます。
- アプリケーションがハードウェアを意識することがなくなることにより、ハードウェアに付加的な機能を付けたり、仕様を変更したりしてもアプリケーションを変更する必要がなくなります。
- 仮想化により、同じプログラムを異なるOSやハードウェア上で動作させることが可能となります。