コンピュータ仮想化と Docker

那須 大晃，森村 周平
Hiroaki NASU，Shuhei MORIMURA

1　はじめに

開発環境で動作していたプログラムが，本番系のシステムに移行し，実行すると環境変数が違っていたり，ライブラリのバージョンが適切でないといったことが原因で動作しないということはよく発生する．このため，開発者にとって実行環境のチューニングが足かせになる．このような，開発者の悩みを背景として，Docker のようなアプリケーションの開発環境の作成と廃棄が容易に行える仕組みが発展した．

2　仮想化

2.1　仮想化とは

仮想化とは，ハードウェアを利用するときの論理的な構成を，利用者やアプリケーションに使いやすい形に変換して提供することである [1)]．

OS はハードウェア資源の仮想化を行うものである．また，複数台のマシンを一つにまとめるクラスタリングを行うことも仮想化の一つである．このように，コンピュータシステムにおいて仮想化は様々なレベルで行われる．

ところで，CPU の動作モードにはユーザモードとカーネルモードがある．これらの動作モードと OS の関係を Fig. 1 に示す．

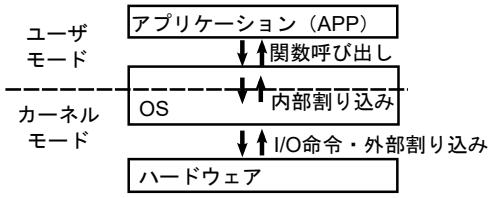

Fig.1　ユーザモードとカーネルモード

メモリ管理や I/O 命令はカーネルモードでしか実行できない．そのため，アプリケーションはメモリの確保や，I/O をカーネルモードで動作している OS へ依頼することによってアプリケーションの処理を行っている．

仮想マシンは複数の OS を一つのコンピュータで動作させる仕組みである．カーネルモードが利用できるのは一つの OS に限られ，仮想マシンはカーネルモードを疑似的に作り出すことによって複数の OS を動作させている．このため，仮想マシン上で動作する OS はオーバヘッドが大きく動作が緩慢となるという問題があった．しかし，2000 年代に入ると，Intel の VT-x など，CPU が仮想化支援機構を搭載するようになる．VT-x とは，CPU に仮想マシンのための動作モードを追加したものであり，これによって大幅にオーバヘッドが削減され，サーバ仮想化は実用レベルの技術となった．

2.2　サーバ仮想化の目的

サーバ仮想化の利用目的には以下のものがある．

- 複数台のサーバをまとめて物理的なサーバ台数を減らすサーバ統合
- サーバ上で動作しているサービスを別のシステムへ移すサーバ移行
- 本番環境と同等の開発環境の提供
- 異なる物理サーバ上へ稼働中のシステムを移動させることによる高可用性の実現

2.3　サーバ仮想化の種類

仮想マシンにおけるサーバの仮想化を，OS との関係で分類するとハイパーバイザ型およびホスト型，コンテナ型の 3 種が存在する．仮想化の模式図を Fig. 2 に示す．

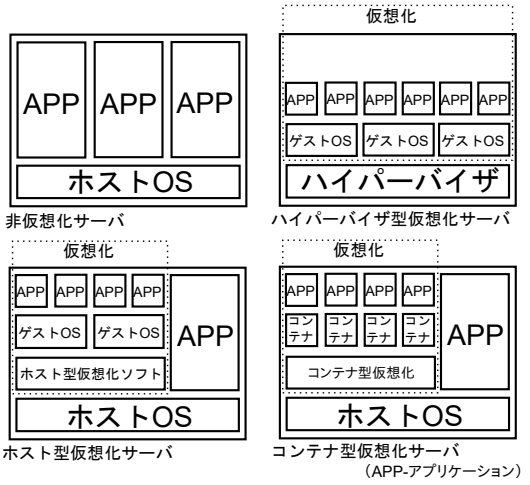

Fig.2　サーバ仮想化の種類

ハイパーバイザ型仮想化とは，サーバ上に独自の仮想化管理ソフトウェアであるハイパーバイザをインストールし，その上で仮想マシンを稼働させる方式のことである．このとき，ハイパーバイザ型仮想化はサーバ全体を仮想化し，仮想マシン上の OS のカーネルモード命令を変換し動作している．

ホスト型仮想化は，ホストの OS 上に仮想化ソフトウェアをインストールし，その上にゲストとなる OS を配置するもので，ゲスト OS の命令をホスト OS が解釈できるように変換する処理を行う．ホスト型仮想化は，Fig. 2 のようにホスト OS が立ち上げたアプリケーションと仮想化上のアプリケーションとの共存ができる．

以上二つの方式は，仮想化を行っていない OS と比べてカーネルモード命令をエミュレーションするためのオー

バーヘッドがあり動作が遅くなる．そのため，このオーバーヘッドをなくす方法の一つとして，コンテナ型の仮想化技術が考え出された．

コンテナ型仮想化は，一つの OS にコンテナという，独立したサーバと同様の振る舞いをする区画を用意し，それを個別のユーザやサービスに割り当てるものである．コンテナ型はホスト型仮想化と構成が似ているが，コンテナ内にゲスト OS を含まず，コンテナ上のアプリケーションがホスト OS のカーネルを利用して動作する．コンテナ型仮想化はゲスト OS が存在しないため，コンテナごとに異なる OS を動作させることはできなくなる．

3 Linux のコンテナ技術

コンテナ型仮想化を実現する要素として Linux namespaces および Linux cgroups，AUFS（Another Unionfs）がある [2]．

Linux Namespaces はコンピューターリソースの隔離を行うものであり，例えば，ネットワークインターフェースを複数あるように見せかけて，それぞれに異なる IP アドレスを割り当てることで単一のコンピュータ上で複数の web サーバを稼働させるといったことができる仕組みである．また，ファイルシステムもプロセス単位で分離することができ，異なるデバイスをマウントすることができる．

Linux cgroups は，プロセスをグループ化して，あるグループに存在するプロセスに対してハードウェア資源の配分を調節する仕組みである．

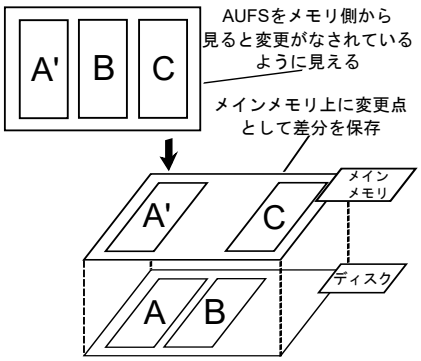

Fig.3　AUFS の図解

Fig. 3 は AUFS を模式的に示したもので，ディスクとメインメモリの 2 層を表している．AUFS 起動時にディスクでは A，B のファイルが存在し，AUFS 起動後のデータ変更が A’，C としてメインメモリ上に保存されている様子を表す．

AUFS は変更が行われた後のデータをメインメモリ上に差分データとして保存する．従って，Fig. 3 の上部のように AUFS をメモリ側から見ると，適切にデータ変更がなされているように見える．このとき，ディスクにデータは書き込まれておらず，一度仮想サーバを停止すると次に立ち上げる仮想サーバには先の変更は全く残っていない．これによって，開発時に行った変更の破棄が容易になる．

以上三つの技術を使うことで，ホストの実環境上に仮想の Linux 環境をコンテナとして動かすことができる．

4 Docker

4.1 Docker の概要

Docker はコンテナ型仮想化管理ソフトの一種である．Docker は，開発者が容易にアプリケーションの動作環境を作るという開発思想の元で作られ，ソフトウェアの開発・配備・展開を支援するという特徴をもつ．

4.2 Docker の利用方法

Dockerfile はアプリケーションを動作させるための基本的な環境や設定も記述するファイルである．Docker は Dockerfile を用いることで，スクリプトを使ってアプリケーションの動作に必要な実行環境を管理することができるようになる．

Docker には，Dockerfile から生成できる Docker イメージというものが存在し，Docker イメージを一般に公開する仕組みを有している．具体的には，Docker レジストリというウェブサービス上で他のユーザと Docker イメージを共有することが可能となる．このような機能を持つことで，Docker イメージに書かれた，他のユーザが作成したソフトウェアとソフトウェアの構築プロセスを自分のサーバ上に容易に再現することが可能となる．

この機能を応用すると，開発者が本番系とローカルの開発系のシステムで開発することを考えたとき，ローカルでの開発環境と実行環境を Docker イメージとしてパッケージ化することで，Docker 上に構築する本番系システム上へ開発済みシステムの配備が容易に行える．

5 まとめ

コンテナ型仮想化はハイパーバイザ型仮想化やホスト型仮想化とは異なり，厳密な意味での仮想マシンではないが，ゲスト OS を持たないためにオーバーヘッドがないという利点を持つ．つまり，コンテナ型仮想化は仮想マシンに求められる実用的な機能を上手く実現しており，着眼点の良い方式であるということができる．Docker はコンテナ型仮想化の管理を行う仕組みであるが，セキュリティに関する機能を持っておらず，脆弱性が指摘されている．

2016 年 6 月よりコンテナ型仮想化の標準化を行う Open Conteiner Project が開始される．ここで，各種の改善が検討され，セキュリティなどの仕組みも導入されることになると考えられる．

Microsoft は Windows Server 上で Docker が利用可能になると発表した [3]．Docker の利用が広がれば，システム開発運用上の知見が蓄積され，Docker の機能向上が促進されると考えられる．

参考文献

1) 前川守: オペレーティングシステム，岩波書店（1989）．
2) ” いまさら聞けない Docker 入門（1），アプリ開発者もインフラ管理者も知っておきたい Docker の基礎知識, http://www.atmarkit.co.jp/ait/articles/1405/16/news032.html.
3) ” docker，Docker and Microsoft, https://www.docker.com/microsoft.