IT 技術学習のための
教育用計算機システムの研究

Study of Educational Computer System for Learning IT

平成26年度 学位論文(修士)

琉球大学大学院 理工学研究科
情報工学専攻

平良 太貴

要旨

IT 技術学ぶ時に、学習環境として実際の計算機上でエミュレートされた仮想の計算機である Virtual Machine (以下 VM) と、OS 上の隔離された環境を構築する技術であるコンテナが重要である。これらはローカルに設置された計算機、あるいはクラウド上に作られる。学生の使用する VM やコンテナを、学生側あるいは教員側から適切に管理するシステムが必要となっている。管理システムには、マルチユーザで動作するのは当然として、入門者や管理者に適した UI、実験的な VM に必然的に含まれるセキュリティの問題への対処などが含まれる。学生の演習には、VM の作成やアップロード、セキュリティ管理、Web サービスの実装などがある。オンプレミスな環境でのテストから、自動的にクラウドにデプロイする継続的開発も重要である。

本研究では、クラウドと VM 、コンテナを管理する Shien システムの構築と評価を行った。複数のユーザに対応するための virsh の wrapper である ie-virsh、コンテナ管理ソフトウエアである Docker の wrapper である ie-docker を作成した。またその際に計算機同士で共有する iSCSI ディスクに使う Filesystem と、コンテナや VM の IO 速度の評価を行った。

Abstract

To study Information technology, Container or Virtual Machine (VM) environment is important. VM emulates Computer to control and to share the resources, and Container isolate resources in an operating system. These are built on premise blade machine or on a cloud service. Management these VM and Container from students or stuff is necessary. Management system includes multi-user support at first, suitable UI, and security issues which inevitably arises from experimental VM. Student may create VM, upload it to the cloud, manage security and implement Web services on the system. Continuous Integration, that is, automatically test on a local system and deploying is also important study.

In this study, we built and evaluate a system called Shien, which manage cloud, VM and Container. To make system multi-user, we developed ie-virsh (virsh wrapper) and ie-docker (docker wrapper). We presents the evaluations of Shien system and we also evaluate various file systems such as GFS or ZFS.

目 次

図 目 次

表 目 次

第1章　IT 教育向けの計算機管理

IT 技術は VM やコンテナの普及により、より手軽に開発し試せる環境が整ってきている。手元の PC 上で VM やコンテナを立ち上げ、ウェブサービスやシステムの開発を行うことができる。しかし VM やコンテナを使用して IT 技術を学習するためには、高性能 PC の購入や有料のクラウドサービスを使用しなければならないため、大きなコストがかかる。これらの負担を IT 技術を学ぶ学生に負わせない、新たな仕組みが必要である。

また本学科では定期的にシステムの更新を行う。次期システムではクラウドサービスとの連携を予定している。クラウドサービスへ学生が開発したサービスをオンプレミスの環境から外部のサービスへ移行することで、学生の開発した Web サービスへの、遠隔地からの高速なアクセスが可能になる。更にクラウドサービスを使用した Web サービスの開発のワークフローを学ぶことができる。

本研究では学生の VM やコンテナの学習環境の提供と、クラウドサービスへの連携することのできる Shien システムを提案する。 Shien システムは複数の計算機と、その計算機の間で iSCSI ディスクを共有すること想定している。共有する iSCSI ディスクは、複数の計算機からの並列なアクセスに耐えられる Filesystem が必要である。その Filesystem には GFS2 を採用した。

Shien システムでは本研究室で開発している ie-virsh と ie-docker を使用する。ie-virsh は VM の管理に、ie-docker はコンテナの管理に使用される。Shien システムの評価では、使用する Filesystem の GFS2 の計測・評価を行った。本学科で行われている授業 Operating System で使用した際の利点を挙げ評価を行った。

1.1　本論文の構成

本論文では、始めにこれまでの OS 管理システムに必要な要素を洗い出す。次に本学科の 2 つのこれまでの OS 管理システムについて述べる。VMWare ESXi を用いた OS 管理システムと、ie-virsh を用いた OS 管理システムの特徴や操作方法について説明する。そして始めに挙げた要素を満たしているか懸賞を行う。第 3 章では Shien システムの全体構成について説明し、使われているツールや Filesystem 、サーバ構成について述べる。第 4 章では第 3 章で説明した Shien システムの管理と利用方法について説明する。第 5 章では Shien システムで使われる Filesystem の GFS2 を、他の Filesystem と比較し、また VM やコンテナからの GFS2 を読みだした際の性能を評価する。更に授業で使用した際の使用方法や問題点を挙げる。第 6 章では、本研究におけるまとめと今後の課題について述べる。

第2章　教育用計算機環境

本章では教育用システムの要件を挙げる。

2.1　Multi User Support

本学科では一学年 60 名、学科全体で 240 名の学生が在籍している。そのすべての学生に対応するため、複数のユーザを管理できる必要がある。そのためには、複数のユーザに同じ権限を容易に配布するシステムでなければならない。

現在の本学科の VM 管理システムでは、複数のユーザに対応するための権限の配布は VMWare vSphere Client で行われている。VMWare vShpere Client は豊富な機能を持っており、権限も細かく配布することができる。しかし多くのユーザに権限を配布するには手間がかかってしまうため、その手間を減らさなければならない。

配布するべき権限は二種類ある。管理者とユーザである。管理者はシステムそのものを管理する。計算機や hypervisor の設定、全ての VM やコンテナの操作をすることができる。

ユーザはユーザ自身の資源のみを操作することができる。またシステム全体に関わるネットワークや他ユーザの VM などの操作は行うことができない。

2.2　コンテナへの対応

OS の環境を複数のグループで分割し、別のサーバのように利用することのできる技術がコンテナである。サーバ運用でコンテナを使用する例が増えてきている。一台のサーバに対して複数のサービスを、VM に比べて少ないオーバーヘッドで使用することができるためである。これからコンテナでの運用が主流になっていくので、教育用システムでもコンテナを使用可能にしなければならない。そのためには Linux をサーバに用いる必要がある。

2.3　資源の制限

計算機の資源は、主にストレージ、CPU 、メモリである。ユーザに提供できる計算機資源は限られている。ユーザ１つに対して多くの資源を与えてしまうと、他のユーザへ資源の配布が困難になる。

また明示的に資源を与えなくても、ユーザが管理者の許可無く大量の資源を確保することを防ぐ必要がある。

現在のシステムではユーザは VM という形で資源を切り出している。 VM を新たに作成するにはメールで申請する。メールを受け取った管理者は VMWare ESXi 上に VM を作成し、VM 内に作成したユーザを申請者にメールで返信する。学生は最大 240 名いるため、一人ひとりに VM 申請作成を行うのは困難であり、全員に均等に VM を使って学習する機会を与えられない。

制限可能な資源は、VM の場合は以下のようになる。

- 使用 CPU core 数
- メモリ容量
- 作成可能な VM の数

またコンテナの場合は以下のようになる。

- 使用 CPU core 数
- メモリ容量
- 作成可能なコンテナの数

これらを制限することができれば、管理者がユーザの資源を固定できる。

2.4　iSCSI ファイルシステムへ対応

計算機の外部に大容量の記憶媒体を置くことで、計算機そのものに記憶媒体を搭載しなくても多くのユーザにデータ容量を配布できる。図 2.1 のように、次期システムでは複数の計算機で大容量 iSCSI ディスクを共有する。

複数台の計算機で iSCSI ディスクを使用することで、 VM イメージやユーザのデータを共有できる。またそれによって VM をライブマイグレーション可能にする。計算機がメンテナンスをしなければならなかったり、計算機を止める必要のある際に VM を止めずに他の計算機へ移動することができる。

iSCSI で複数台の計算機から接続された場合に、高速で整合性を保つことのできる Filesystem を使用する必要がある。

2.5　クラウドへの対応

クラウドサービスとは外部のサービス上でウェブサービスなどのシステムを稼働させる事のできるサービスである。現在のシステムでは、クラウドサービスへの対応を行ってい

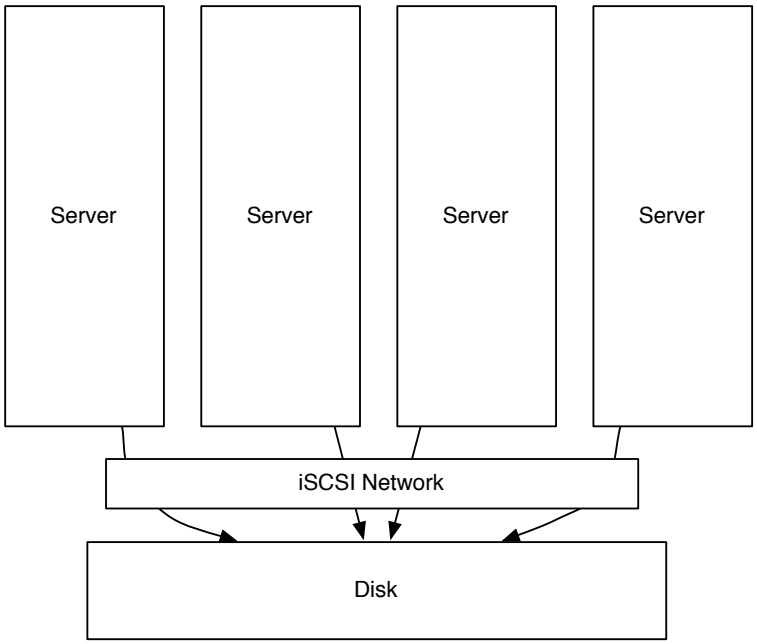

図 2.1: iSCSI ディスクを使用した構成

ない。学生が外部のクラウドサービスを使用するためには、自主的にコストを支払い一から構成する。

次期システムではクラウド上の資源を確保する。本学科のシステムのバックアップが行え、クラウドサービスを学生が使えるようになる。

次期システムへ対応するため、クラウドサービスへ学生の開発したサービスやシステムのデプロイが行え、また管理者の負担を抑えて管理できるようにしなければならない。

2.6　Linux kernel debug

Linux Kernel のソースコードを読む方法は 2 つ挙げられる。Linux kernel をダウンロードし、そのソースコードをそのまま読むという手法と、gdb で逐次ソースコード追って読むという手法である。教育用の環境の機能の一つとして、Linux kernel を debug する、という機能が必要である。gdb で Linux kernel のソースコードを追う準備には、手間がかかってしまう。 また授業で Linux kernel を題材に出す際、gdb で追うことができると課題の幅が広がる。

2.7　セキュリティ管理

教育用の環境で VM を使用する場合、ユーザの VM へのセキュリティ管理が重要になる。具体的には脆弱なパスワード設定や無駄なポートの開放である。それをユーザに通知

する機能が必要になる。またユーザの VM が外部を攻撃したりすることを防がなければならない。

第3章　これまでの学生向け VM 管理システム

本章では 2015 年現在使用されている、琉球大学情報工学科 (以下本学科) の運用している 2 つの VM 管理システムについて述べる。

3.1　VMWare ESXi

VMWare が無償で配布している Hypervisor であり、計算機に直接インストールし使用することができる。現在本学科では VMWare ESXi を Hypervisor として VM を管理している。

3.2　VMWare ESXi による VM 管理システム

オンプレミスのブレードサーバで構成されており、VMWare ESXi で動作している VM を VMWare vSphere Client と Web サービスで管理している。図 3.1 のように、VM の所有者はメールでシステム管理者へ使用 VM を増やす依頼を行い、ユーザは本学科の提供する Web サービスから VM を作成する。資源の制限はシステム管理者が申請を受諾することで行っていた。複数の学生で使用するため、Multi User Support に対応している。またコンテナには対応していない。

3.3　本学科の提供する Web サービス

VM を操作するインターフェイスとして、VMWare の API を使った Web サービスを使用できる。

本学科のシステムでは、VM の操作は Web サービス実装を通して行う。VM の作成はメールなどの連絡手段を使って、管理者を通して行う。既成の VM をブレードサーバへアップロードするにも、管理者と連絡し手続きを取る。

Web サービスのポータル画面は図 3.2 のようになる。本学科では IP アドレスもこの画面で配布している。

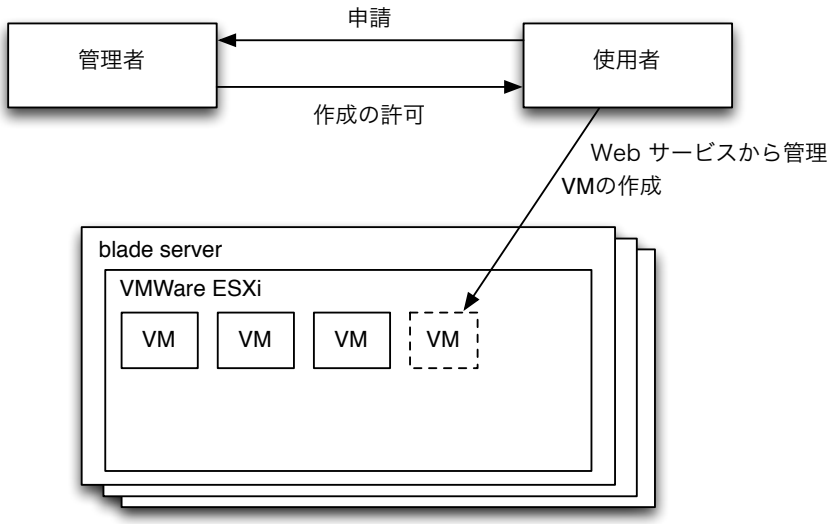

図 3.1: 本学科のシステム

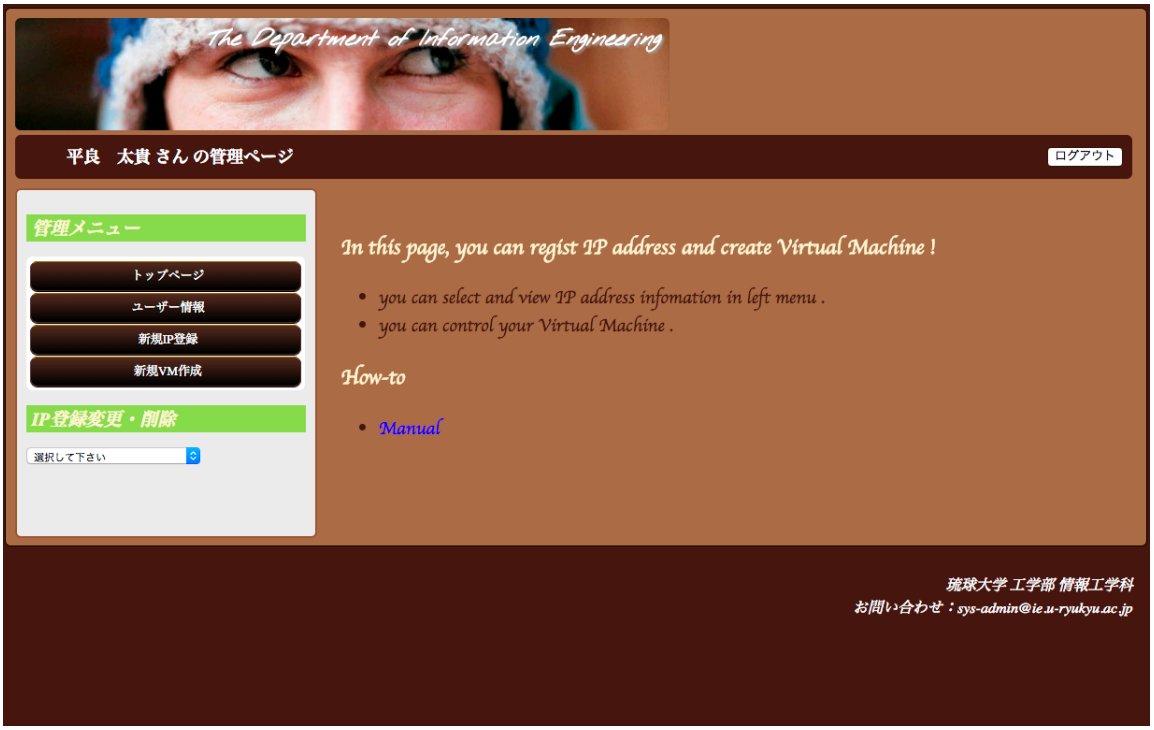

図 3.2: 本学科の Web サービス

VM を作成するためには、図 3.2 の VM 新規作成をクリックする。クリックすると図 3.3 の画面に移る。図 3.3 で必要な情報をすべて入力し、「ok」をクリックすることによって VM を作成することができる。

図 3.3: Web サービスの VM 作成画面

VM 作成後は VM を選択すると、図 3.4 へ移り、VM を管理することができる。本学科の Web サービスの管理画面で可能なことは、下記になる。

- 起動
- 停止
- サスペンド
- 再起動
- スタンバイ

3.4　VMWare vSphere Client

GUI で VM を操作することができ、豊富な機能と高度な操作が可能となっている。管理者は VMWare vSphere Client での管理を行っている。VM などの資源に対する操作の

図 3.4: Web サービスの VM 管理画面

権限を細かく扱うことができるため、利用者に対して権限を配布することが可能である。しかし GUI が複雑なため、操作に習熟する必要がある。

また使用する OS が限定されており、主に Windows でのみ使用可能なため、Mac OS X を推奨している本学科では使用が困難である。

3.5 Kernel based Virtual Machine (KVM)

Linux 自体を VM の実行基盤として機能させるソフトウエアである。無償で利用可能なオープンソースとなっている。CPU の仮想化支援機能を必要とし、完全仮想化により仮想化環境を提供する Hypervisor である。

KVM は Linux のカーネルモジュールとして実装されており、OS が持つメモリ管理プロセスやスケジューリング機能を利用している。アーキテクチャは図 3.5 のようになっている。

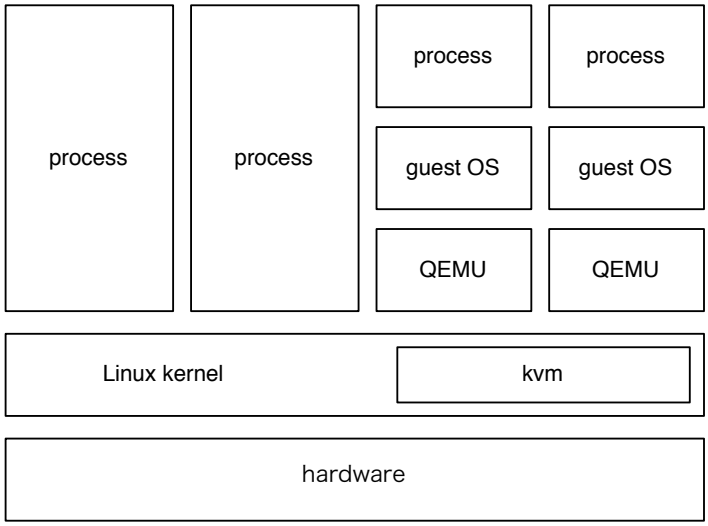

図 3.5: KVM architecture

Intel-VT や AMD-V などの仮想化支援機能を持つプロセッサや BIOS の載っている PC 上で動作する。

本学科のシステムは、KVM と VMWare ESXi の 2 つの Hypervisor を利用している。本学科全体の VM 管理には、主に VMWare vSphere Client が使われている。

3.6 libvirt

VM 管理ツールである virsh を含む、 VM の制御を抽象化したライブラリである。 VM の情報を習得・操作することが可能な API 群である。

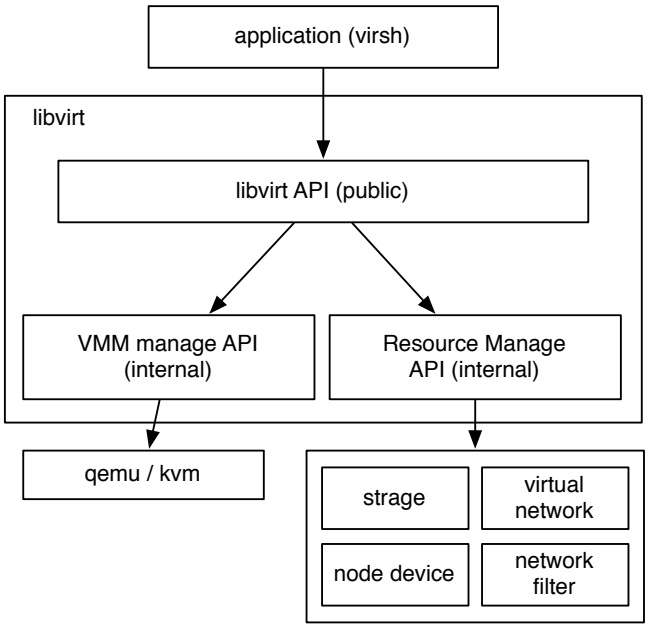

図 3.6: libvirt architecture

図 3.6 のように、アプリケーションから libvirt API を呼び出すと、API に従って内部の VMM API もしくは資源管理 API を呼び出し、制御を行う。libvirt は VM の管理だけではなく、仮想ネットワーク、仮想ストレージも管理することができる。もともとは Xen に対して API を提供していたが、現在は多くの Hypervisor に対応している。

3.7　virsh

virsh は libvirt に付属する VM 管理コンソールである。libvirt の API で VM を制御することができる。VM の起動や停止、情報の表示、ゲストが接続しているネットワークやデバイスの管理をすることができる。

virsh をそのまま使用して複数のユーザを管理するためには、ユーザやグループの設定が必要である。またユーザやグループの設定を行ったとしても、ネットワークなどの操作をを制限することができない。

3.8　ie-virsh

virsh はグループの設定を行わない限り、root ユーザの権限でのみ使用可能である。virsh を使う管理者以外のユーザ全てに root ユーザの権限を使わせてしまうと、他のユーザに対しての不正な操作や多量の資源取得、ネットワーク等の管理者がするべき操作を許してしまう。

そのため ie-virsh の開発を行った [1]。ie-virsh は virsh をラップすることで作成された VM 管理用のツールである。ie-virsh を使用することにより、他のアカウントの VM を操作させずに VM 所持者に操作させられる。

表 3.1 が ie-virsh の機能である。

表 3.1: ie-virsh のコマンド

define	XML の template を元に domain を作成
undefine	define で作成した domain を削除
list	define で作成した domain を一覧表示
start	指定した domain 名の VM を起動
destroy	指定した domain 名の VM を停止
dumpxml	domain の XML を参照
debug	linux kernel のコードを gdb で読む

virsh ではネットワークやストレージの設定を行うことも可能である。しかし ie-virsh では管理者ではないユーザにはネットワークやストレージの操作・設定ができないよう制限している。

3.9　ie-virsh による資源の制限

複数のアカウントで計算機資源を共有する場合、管理者による資源の制限が必須である。VM の使用が主な ie-virsh では過剰なディスク容量やメモリ、CPU の確保を防がなければならない。

ie-virsh がラップしている virsh は、XML ファイルを使って VM を管理している。XML ファイルには VM のパラメータが記述されている。

ie-virsh では学生が使用する VM が使用する資源を制限するために、予めこの XML ファイルのテンプレートを作成し利用している。

XML テンプレートで予め制限されている設定は、以下のようになる。

- ネットワークの設定
- I/O 設定
- VM イメージのフォーマット
- CPU 数
- メモリ容量

これによって学生が使用する背資源を制限し、過剰なメモリや CPU の確保を防ぐ。また学生が VM を多く作成するという形で資源を利用してしまうことを防ぐために、作成し操作できる VM の数を 4 台に制限した。

3.10　ie-virsh による OS 管理システム

ie-virsh によって複数のユーザの VM 管理を行っている。ユーザに Fibre Channel ストレージに VM をアップロードさせ、本学科のブレードサーバ上で Web サービスの開発や実験のために VM を使用させる。

Fibre Channel ストレージにより VM イメージを複数のブレードサーバから参照できる。Fibre Channel ストレージは複数の計算機から読み書きすることのできる Filesystem が使用されている。

また本学科では Operating System という授業を提供している。この授業では OS について学ぶ一環として VM について学習し、課題を提出させる。課題では VM の環境を学生が設定し、本学科の持つブレードサーバ上に VM をアップロードし、プログラムの実装や計測を行う。ie-virsh による OS 管理システムを使用し、 VM のアップロードから計測までの課題を行わせた。

第4章　Shien システム

本章では Shien システムの構成について述べる。

Shien システムは、複数のアカウントの計算機資源の使用を管理できるよう補助するシステムである。アカウント所有者が開発したアプリケーションのデプロイや、コンテナ・VM を使った実験を行える環境を提供する。

4.1　Fedora

Shien システムは本学科の次期システムに合わせて設計する。そのためには最新のソフトウエアを取り入れ、次期システムが稼働する頃に安定な構成を検証する必要がある。

Fedora は最新の技術を積極的に取り込んでおり、その成果を Red Hat Enterprise Linux に取り込まれるといった検証目的の位置づけになっている。また本学科の基幹システムでは主に CentOS が使われている。CentOS は Red Hat Enterprise Linux と完全互換を目指した Linux ディストリビューションである。

つまり最新の Fedora を使用することによって、教育用 OS 管理システムで使われる OS やソフトウエアを予測し、また検証することが可能である。そのため今回のシステムでは Fedora を利用し、次期システムが稼働する際の環境を構築し、検証を行った。

4.2　Global File System 2 (GFS2)

iSCSI に対応しており、複数のノードからのアクセスに対して整合性のある読み書きをするファイルシステムとして、GFS2 を選択する。

GFS2 は、Linux カーネルファイルシステムインターフェィスに直接的に対応するカーネルファイルシステムである。単独システム内、またはクラスタ設定の一部として使用することができる。GFS2 では、すべてのノードから同じ共有ストレージへアクセスできる。クラスタファイルシステムの一部として使用された場合、GFS2 は分散型メタデータと複数ジャーナルを運用する。

GFS2 でフォーマットされたディスクへアクセスするノードはクラスタ構成になっていなければならない。

4.3 Distributed Lock Manager (DLM)

GFS2 を使用する Red Hat のクラスタでは、ロック機構として DLM が使用される。DLM は GFS2 ファイルシステムへのアクセスなどの、クラスタ内のリソースへのアクセスを制御する。DLM がない場合、共有ストレージへのアクセス制御がなくなり、クラスタ内のノードが相互のデータを破損させる可能性がある。

DLM は各クラスタノードで実行され、またロック管理はクラスタ内のすべてのノードを対象として行われる。

Quorum はノードがクラスタのメンバとして、過半数と通信できるノードがクラスタとして動作できる権利である。二台の場合は過半数が無いため無効化される。

DLM をクラスタで動作させるには、以下のように設定する必要がある。

- ノードをクラスタの一部として設定する
- すべてのノードがクラスタのメンバーであり、Quorum を所持している
- ノード同士が IP アドレスで通信できる必要がある。

4.4 Pacemaker

高可用性クラスタソフトウエアであり、クラスタを構成する。クラスタの状態を伝搬、クラスタへのノードの参加判断、クラスタノード感における情報の同期、一定間隔で相手ノードと通信して生死を確認、などのクラスタ制御を行うことができる。

4.5 The corosync Cluster Engine (corosync)

Pacemaker でクラスタを構成するためには、クラスタの基盤が必要である。corosync は、高可用性を実現するクラスタ基盤ソフトウエアである。

Extended virtual syncrony と呼ばれる形式で、マルチキャストやブロードキャストを使ったノード間のメッセージング機能を提供する。また、クラスタ全体でプロセスグループを管理するための管理機能や、基礎的なアプリケーションの監視などの機能も提供する。

今回使用する Fedora では過去にクラスタを管理するために使用されていた Cluster Manager (CMAN) が削除されているため、今後使用できない。Shien システムの設計をするためには CMAN を外す必要がある。

4.6 Logical Volume Manager (LVM)

LVM とは、複数のハードディスクやパーティションにまたがった記憶領域を一つの論理的なディスクとして扱うことのできるディスク管理機能である。

図 4.1 を用いて説明を行う。LVM はハードディスク内に、Physical Volume と呼ばれる LVM 用のパーティションを用意する。その Physical Volume を初期化し、Physical Extent と呼ばれる小さな領域に分割する。複数の Physical Volume をまとめて新たに作られる管理単位が、Volume Group になる。そして実際のパーティション同様に利用できる Volume 、Logical Volume を作成する。

Logical Volume は物理的なディスクのパーティションと同様に利用することができる。この Logical Volume を GFS2 でフォーマットする。

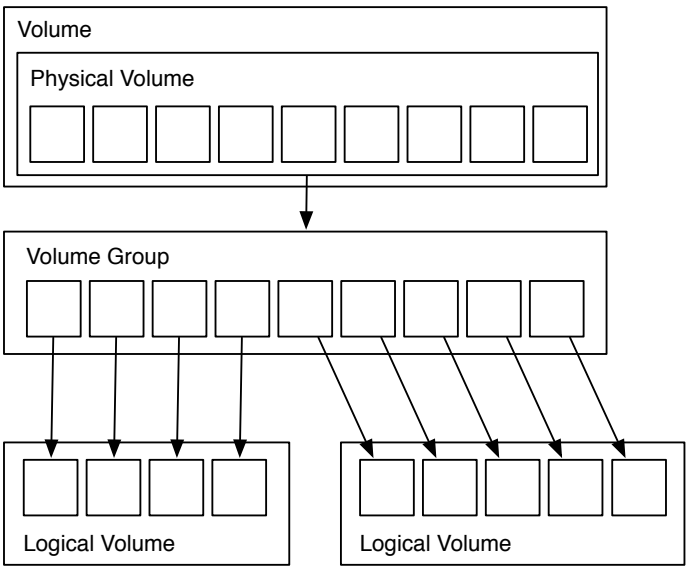

図 4.1: LVM

4.7 Clustered Logical Volume Manager (CLVM)

クラスタを構成するノード間で LVM を使用する場合は、CLVM を使う。

LVM のクラスタリング拡張機能のセットである。一部のホストが変更した LVM 情報を他のホストに通知する。クラスタは LVM を使用した共有ストレージを管理できるようになる。GFS2 は CLVM を利用して複数ノードからのアクセスに対応する。

4.8 構成

図 4.2 がブレードサーバの構成である。サーバ全てに Fedora を導入し、クラスタを構成する。クラスタの構成には corosync を使う。クラスタのノード同士は corosync を使用し、互いに通信を行う。

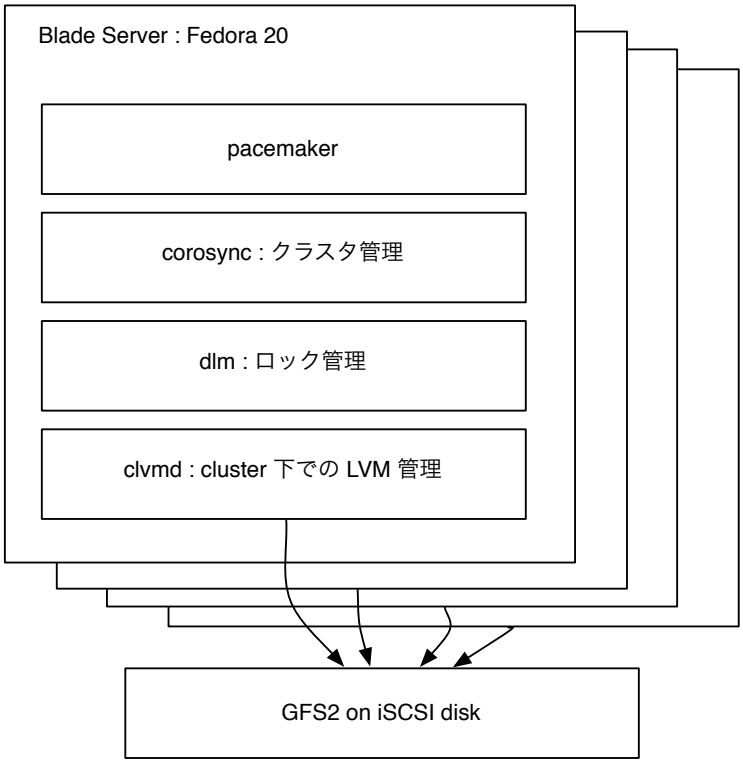

図 4.2: GFS2 Cluster

iSCSI ディスクに LVM で使用できるパーティションを割り当てる。パーティションを一つにする場合は、Physical Volume と Volume Group、Logical Volume は全て一つである。Logical Volume を GFS2 でフォーマットする。フォーマット時に journaling を指定しなければならない。journaling 数は、GFS2 を参照する計算機の数である。

GFS2 でフォーマットされた一つの iSCSI ストレージを共有し、iSCSI ストレージに VM のディスクイメージ、コンテナで動かすアプリケーションを保存する。これにより複数のブレードサーバ間で VM のイメージの共有や移動、コンテナイメージの移動を簡単に行える。

4.9 Docker

Docker とは Docker 社が開発してるオープンソースのコンテナ型仮想化ソフトウエアである。Linux 上で Linux コンテナ (LXC) を活用し、コンテナ型の仮想環境を作成する。

図 4.3 は Docker のアーキテクチャである。

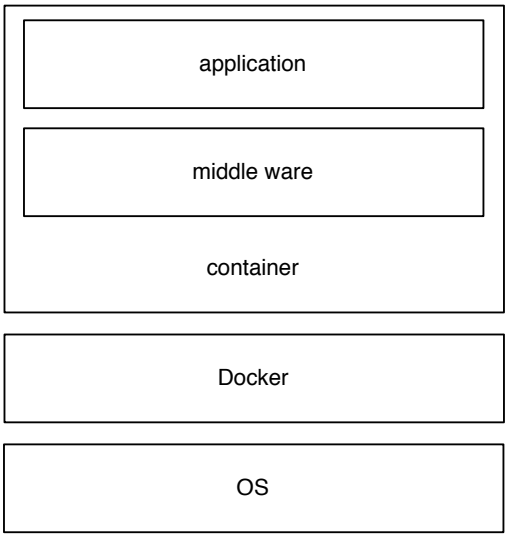

図 4.3: Container architecture

コンテナは OS 環境を複数のグループに区切って別のサーバのように利用する技術である。KVM などの hypervisor 型の仮想化とは異なり、ホスト OS がまとめて管理する。Docker は、コンテナでアプリケーションを実行するため、コンピュータリソースの隔離・制限や、他のホスト、他のコンテナとのネットワーク構成や、ファイル・ディレクトリの世代と差分の管理などの機能を持つ。また容易に配布して実行することもできる。

Docker では新しい独自技術をほとんど用いていない。しかし新たなサーバ資源の運用方法として注目されており、学生が学ぶのに必須な技術の一つである。

4.10 ie-docker

ie-docker は本研究で新たに開発を行った、Docker をラップし複数のユーザで利用することのできるコンテナ管理ツールである。他のアカウントのコンテナを操作させない。またアカウントの使える docker の機能を制限する。表 4.1 が ie-docker の機能である。

またウェブサービスを ie-docker で動作させる際、複数のコンテナが外部に向けて同一の Port を使用することはできない。そのため図 4.4 ie-docker では、docker run を実行すると自動的に Port を取得する。

Port を取得後は、リバースプロキシを使用して外部に公開する IP アドレスやドメイン名に Port を割り当て、外部からウェブサービスが参照できるようにする。

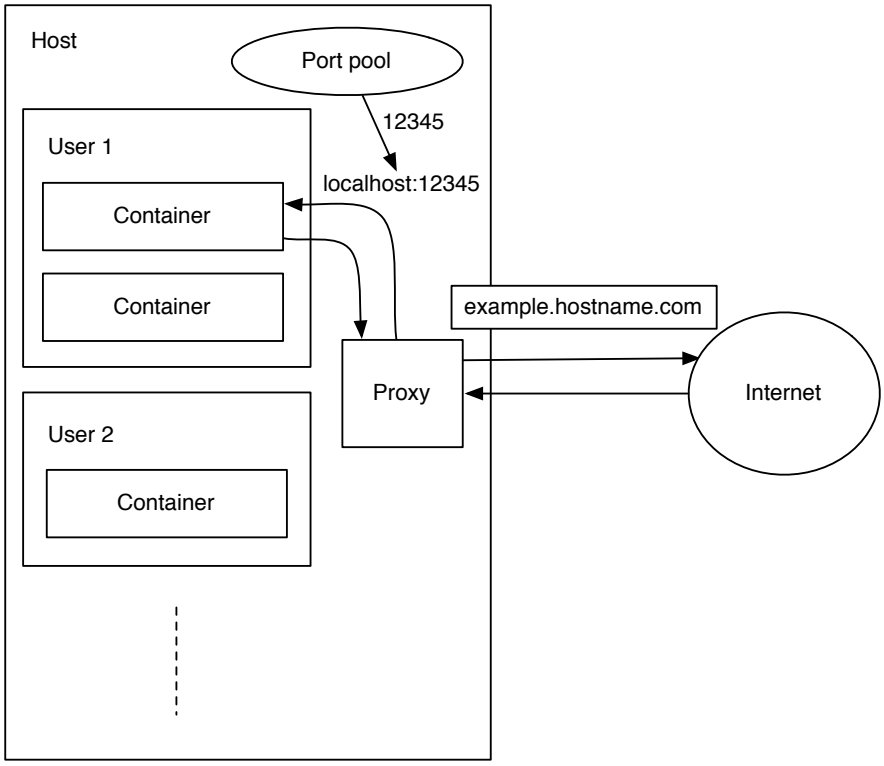

図 4.4: ie-docker による Port の付与

表 4.1: ie-docker のコマンド

run	XML の template を元に domain を作成
build	define で作成した domain を削除
attach	動作しているコンテナに attach する
images	docker image を一覧表示する
rmi	docker image を削除する
rm	docker process を削除する
start	docker process を起動する
stop	docker process を停止する

4.11 ie-docker による資源の制限

Docker はコンテナごとに使用するメモリの上限を指定することができる。ie-docker では、コンテナのメモリ使用量の制限をユーザ全てに管理者が付与できるよう実装した。docker run で Docker のコンテナを起動する。起動時に -m (memory) を指定することでメモリの指定が可能だが、今回は ie-docker でラップし -m (memory) を自動的に付与するよう実装を行った。

また ie-virsh と同様に、1 ユーザが使用可能なコンテナの数を 8 つに制限した。

4.12 クラウド上での使用

クラウドサービスではコンテナをユーザに使用させる。コンテナが VM と比べて高速なためであり、またクラウドサービス上へ VM イメージを送信するよりもデプロイしやすいためである。ie-docker は LDAP を参照してユーザに与える資源を制限しているため、クラウド上でも本学科の LDAP を利用する必要がある。

第5章　Shien システムでの管理方法

本章では、Shien システムを用いたシステム管理の方法と、 VM やコンテナ操作方法を述べる。

5.1　LDAP による権限管理

VMWare vSphere Client では VM やシステムに対する権限を細かく設定できる。しかし各ユーザに対して適切な権限を振るのは多くの時間を要するため、自動的に権限を配布する必要がある。

教育用の計算機システムでは、権限は管理者とユーザのみに分けられる。そのため権限の配布は同一の権限をユーザに一度に振るのがよい。

本研究で提案する計算機システムでは、LDAP から取得したアカウントによって権限を配布する。情報工学科では学生や教師などのユーザアカウントを LDAP で管理しており、そのアカウントをそのまま権限配布に使用することができるためである。

またデータや VM のイメージを保存する iSCSI ストレージに各利用者へディレクトリを自動的に作成し、そのディレクトリにそれらを保存する。ディレクトリに各ユーザと管理者だけがアクセスできるように権限を設定する。

Shien システムの利用者は他の利用者の VM やコンテナに対して操作を行うことができず、またデータや VM を操作・改編することができない。

そのように、LDAP を使用して権限を管理できる。

5.2　ie-virsh で使用する VM イメージのアップロード

ユーザは iSCSI ストレージに VM イメージを保存する。この計算機システムでは各 VM 所有者ごとにディレクトリを分けて使う。

VM 使用者は手元の PC で VM イメージを作成し、実験・開発環境を作成する。次に VM イメージを iSCSI ストレージにアップロードする。アップロード先は iSCSI ストレージ上にあるユーザ自身のディレクトリである。ie-virsh は VM を設定する XML ファイルをテンプレートから作成するため、VM イメージの形式は固定である。そのため、VM イメージをテンプレートに合わせた形式に変換する。

アップロードした後は、ie-virsh で XML テンプレートを使用したドメインを作成する。ie-virsh は virsh のドメインを自動的に定義することができる define コマンドを持っており、そのコマンドでドメインを作成する。

ドメインの定義は下記のように行う。

```
% ie-virsh define [01 - 04]
```

ドメインは 01 から 04 までの名前をつけることができる。これは一人のユーザに対し、VM を４台に制限しているためである。

5.3 VM の起動

ie-virsh を使用して、作成したドメインの VM を起動する。起動するには、下記の操作を行う。

```
% ie-virsh start [01 - 04]
```

また、VM に適切な設定を行うことによって、console でログインすることができる。console を使用するには下記の操作を行う。

```
% ie-virsh console [01 - 04]
```

各 VM 所有者自身で console にアクセスすることができるため、間違った VM の設定を適用したとしても管理者に連絡する必要はない。console でアクセスし、正しい設定に変更することができる。

5.4 VM のリストの取得

VM 所有者は ie-virsh を使用して自身の持つ VM の一覧を見ることができる。下記の操作を行う。

```
% ie-virsh list
```

virsh はオプションで –all をつけることによって停止した VM も一覧することができるが、ie-virsh はサーバ用途以外の実験にも使用されるため、VM を停止する場合も多い。そのため ie-virsh の list コマンドではオプションを付けなくても停止している VM が一覧に含まれる。

5.5 VM の停止

VM の停止は下記のコマンドで行う。

```
% ie-virsh destroy [01 - 04]
```

virsh の destroy コマンドをため、VM を強制的に停止させる。

5.6 VM への console ログイン

VM へ console でログインするには、下記の操作を行う。VM が console から見れるよう、正しく設定しておく必要がある。

```
% ie-virsh console [01 - 04]
```

VM へリモートログインできなくなった時に、VM の設定を修正するために使用させる。管理者は console ログインするための連絡を使用者とやりとりする必要はない。

5.7 VM のブレードサーバ間の移動

ブレードサーバは GFS2 でフォーマットされた Fibre Channel ストレージを共有している。Shien システムではそのストレージに VM イメージを配置する。VM イメージはどのブレードサーバからでも起動することができる。

VM を他のブレードサーバへ移動するには、VM の XML 設定ファイルを移動先のブレードサーバへコピーし、virsh のコマンドで define する。

Fibre Channel ストレージにある VM イメージのパスが同じであれば、そのまま起動することができる。

また VM イメージは共有しているため、移動元の起動中の VM の VM イメージを使って移動先のブレードサーバで VM を起動することも可能となっている。

virsh の機能として、ライブマイグレーション機能がある。GFS2 はライブマイグレーションに対応している。ライブマイグレーション機能を利用することで、他のブレードサーバへ起動中の VM を起動したまま移動させることが可能である。

5.8 Kernel debug 方法

本研究では、ie-virsh の新しい機能として debug コマンドの実装を行った。本学科には Operating System という授業がある。その授業では OS について学習するため、Linux kernel を読む課題を出す。ie-virsh の開発はその授業 Operating System の課題に対応するためである。

KVM には gdb で接続するための Port を指定し、接続すると gdb から Linux kernel のデバッグへ入ることができる。更に libvirt の XML 設定ファイルに、KVM へ gdb で接続するために開ける VM の Port を記述することで、VM の Port を開くことができる。

今回は図 5.1 のように、Port の Pool から Port 番号を取得し、デバッグ対象の VM を起動し、その VM へ gdb で接続するという方法で実装した。Port は、libvirt の XML 設定ファイルに Port 番号を書き込み、VM を起動することで割り当てられる。

Kernel debug をするためには、下記のコマンドを実行する。

```
% ie-virsh debug
```

実行後は gdb に入る。

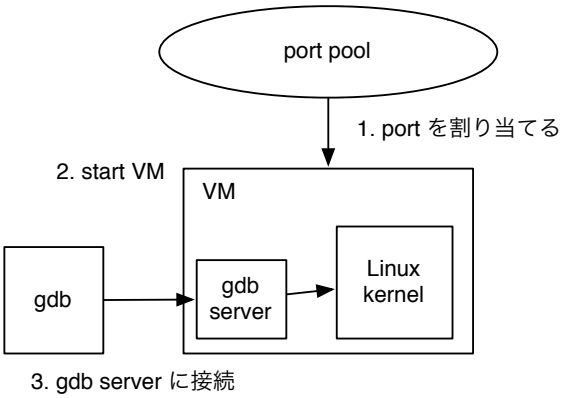

図 5.1: ie-virsh debug

5.9　ie-docker によるコンテナの起動

任意の Process name を下記のコマンドを使用して割り当て、コンテナを起動する。

```
% ie-docker run -i -t --name [process name] [image name] [exec command]
```

Process name は任意に割り当てることができる。同名の Process name は使用できない。また process name はアカウント名が ie-docker によって補完されるため、他のユーザが作成した p Process name と被ることはない。

一度 run を行ったコンテナに対しては、run を打たなくても下記のように起動することができる。

```
% ie-docker start [process name]
```

5.10　ie-docker によるコンテナの停止

コンテナの停止は下記のコマンドを使用する。

```
% ie-docker stop [process name]
```

5.11　Docker からの iSCSI ストレージの使用

最初に iSCSI ストレージ上に Repository を作成する。 下記のコマンドを実行することによって管理者が指定した iSCSI ストレージ上のディレクトリに Repository が作成される。また Repository ごとの Port 割り当ても下記のコマンド実行時に行われる。

```
% ie-docker create [repository name]
```

次に Repository name を使用してコンテナを起動する。またコンテナ内部から外へ出す Port を指定する。これによってホストの Port とコンテナの Port を接続し、コンテナの外部と通信することができる。

Docker の特徴として、Commit を行なわずにコンテナを止めてしまうと、コンテナ内部に保存していたデータが消える。つまり Commit 前にコンテナに追加されたデータを保つためには、コンテナの外部にデータを保存しておかなければならない。

Shien システムでは、iSCSI で接続されたストレージにコンテナのデータを保存する。そうすることによってデータを保護することができ、またコンテナのイメージ内に動作に必要でないデータを配置しなくてよい。

Docker は、ホストのディレクトリをコンテナ内部のディレクトリにマッピングすることができる。ie-docker もその機能を持つ。-v オプションを使い、下記のように Docker コンテナを実行する。

```
% ie-docker run -i -t -p [port number] -v [directory:container directory] \
    --name [process name] fedora:20 /bin/bash
```

実行されたコンテナはホストのディレクトリにマッピングされた、任意の名前のディレクトリを参照することができる。iSCSI ストレージ上にアプリケーションの Repository を配置し、コンテナでその Repository 上のアプリケーションを動作させる。

また ie-docker は指定したディレクトリにまとめてユーザの Repository を作成する。管理者は指定したホストのディレクトリに対して容量の制限をかければよい。

起動中のコンテナに対しては、下記のコマンドで接続し操作することができる。

```
% ie-docker attach [process name]
```

attatch 後は Ctrl-p Ctrl-q でコンテナを止めずに切断することができる。

第6章　Shien システムの評価

本章では Shien システムの評価を行う。新たに使用した GFS2 との比較と、授業で使用した際の評価を述べる。

6.1　実験環境

Shien システムの検証目的の一つは、本学科の次期システムでの使用を検討するためである。評価には本学科でのブレードサーバ環境を使用した。ブレードサーバの仕様は表 6.1 となっている。

表 6.1: ブレードサーバの仕様

名前	概要
CPU	Intel(R) Xeon(R) CPU X5650@2.67GHz
物理コア数	12
論理コア数	24
Memory	132GB
OS	Fedora 20

現在の本学科のブレードサーバと接続されているストレージは Fibre Channel ストレージである。そのため Fibre Channel ストレージを用いて GFS2 の性能計測を行った。

6.2　実験概要

ベンチマークは Filebench というベンチマークツールを用いて行った。Filebench はファイルシステムパフォーマンスの測定と比較のためにファイルシステムのワークロードをフレームワーク化したものである。シンプルなワークロード構築用の言語 .f 言語が搭載されている。

Filebench は .f 言語で書かれた基本的なワークロードを持っている。今回はそのワークロード randomread.f と randomwrite.f を用いて計測を行った。

6.3 Filesystem の速度比較

図 6.1 と図 6.2 は ext4 、ZFS 、GFS2 における読み書きを行うファイルのサイズに対する速度である。

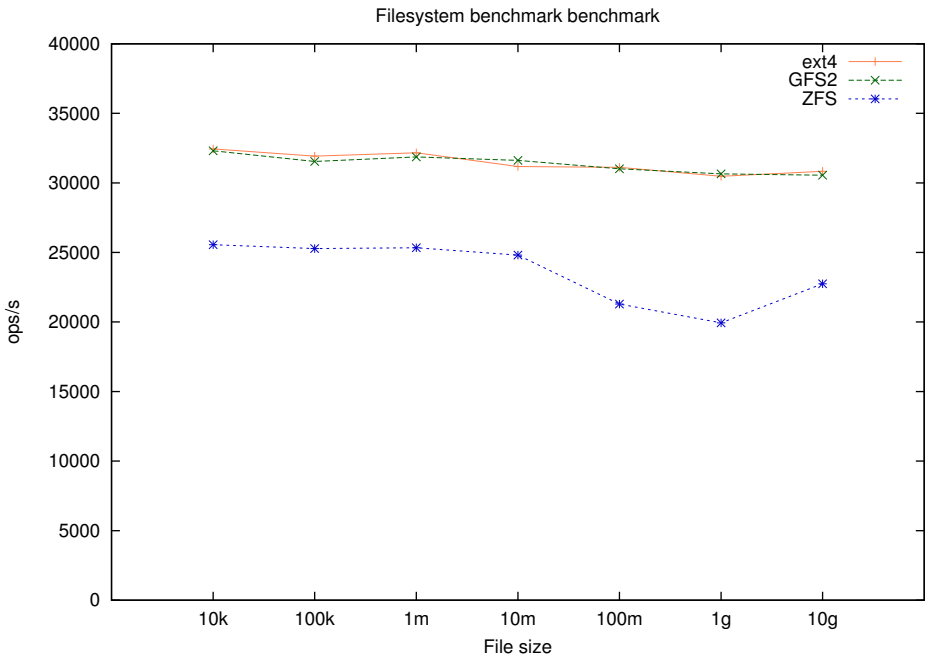

図 6.1: Filesystem ごとの Random read 性能比較

6.3.1 考察

Linux の標準的に使用されている Filesystem の ext4 は、Linux から直接アクセスされるため高速である。しかし今回の計測結果では、GFS2 の読み込み速度は ext4 と殆ど変わらない。読み込み速度に関しては、Linux のローカルファイルと同等の速度で読み込める。

そのためアプリケーションや VM が GFS2 上にあったとしても、読み込み速度はローカルディスクと変わらないため GFS2 が原因で速度が低下することはないと考えられる。

また Write に関してもファイルサイズが 100 MB の場合までは ext4 と比べ殆ど変わらない速度となっている。しかし 1GB 以上の大きなファイルへの読み込み速度は ZFS とともに極端に落ちてしまう。

VM は内部で VM イメージに対して停止した時に書き込むため、VM が停止した際の書き込み量によっては極端に遅くなると考えられる。

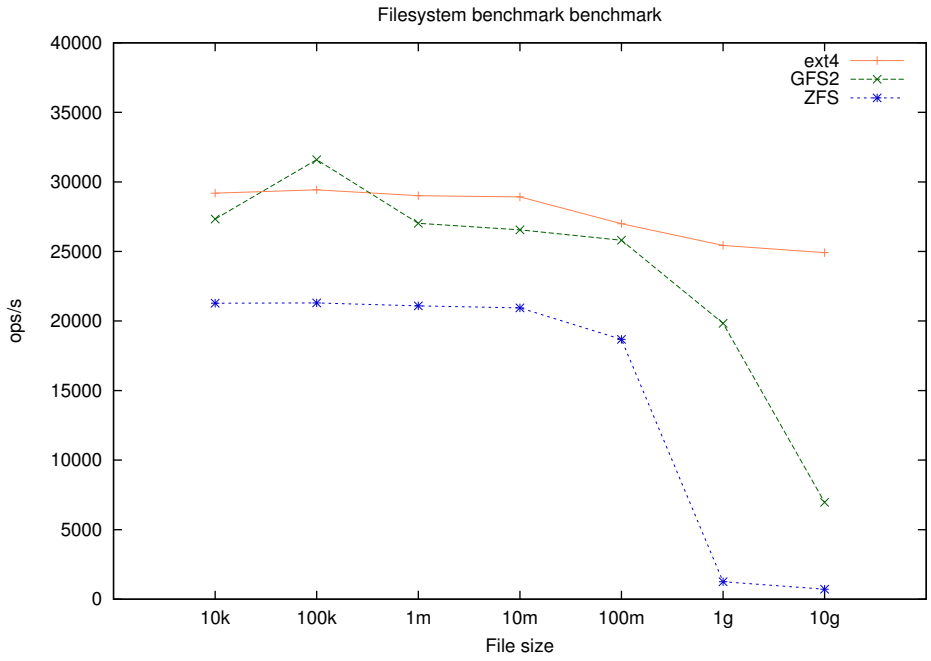

図 6.2: Filesystem ごとの Random write 性能比較

6.4 GFS2 上の複数ノードの計測

図 6.3 と図 6.4 は GFS2 の Fibre Channel ストレージを、同時に複数のブレードサーバから読み書きしたものである。読み書きしたファイルは別々のディレクトリにある。

6.4.1 考察

GFS2 のロック機構、DLM は inode ごとにロックを管理している。そのため全く同一のファイルに対して読み書きを行わなければ、一定の速度で読み書きすることができる。今回計測を行った 3 ノードからのアクセスでも殆ど変わらない結果が見て取れる。

6.5 VM とコンテナとの比較

図 6.5 と図 6.6 はベンチマークをホストから GFS2 を計測した場合と、VM とコンテナ上から計測した場合のグラフである。VM は Fibre Channel ストレージ上に VM イメージを置き、KVM を Hypervisor として起動した VM 内から Filebench を使用しベンチマークを計測した。

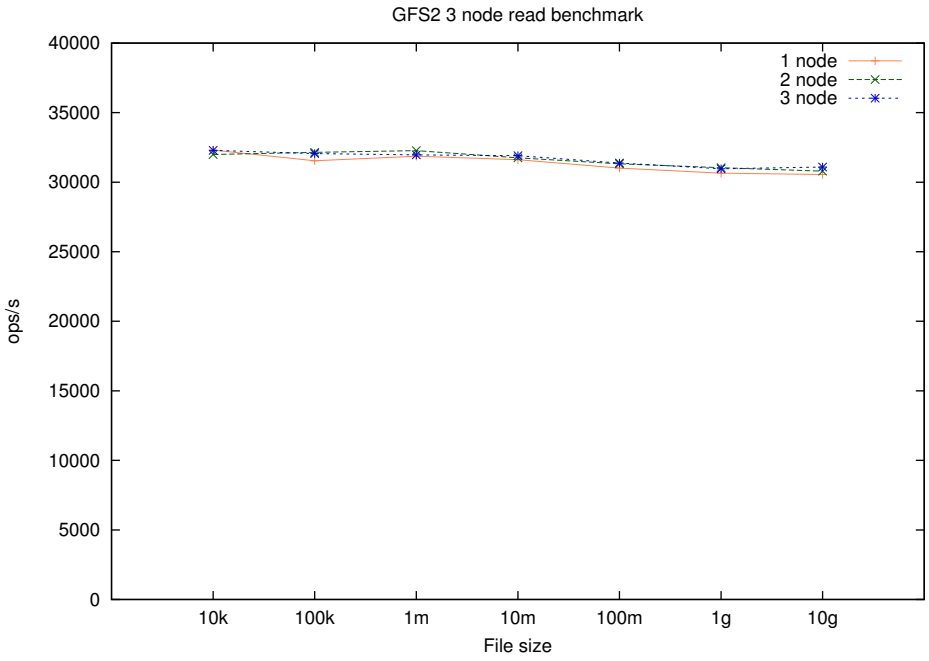

図 6.3: GFS2 から参照したノード数ごとの Random read 性能比較

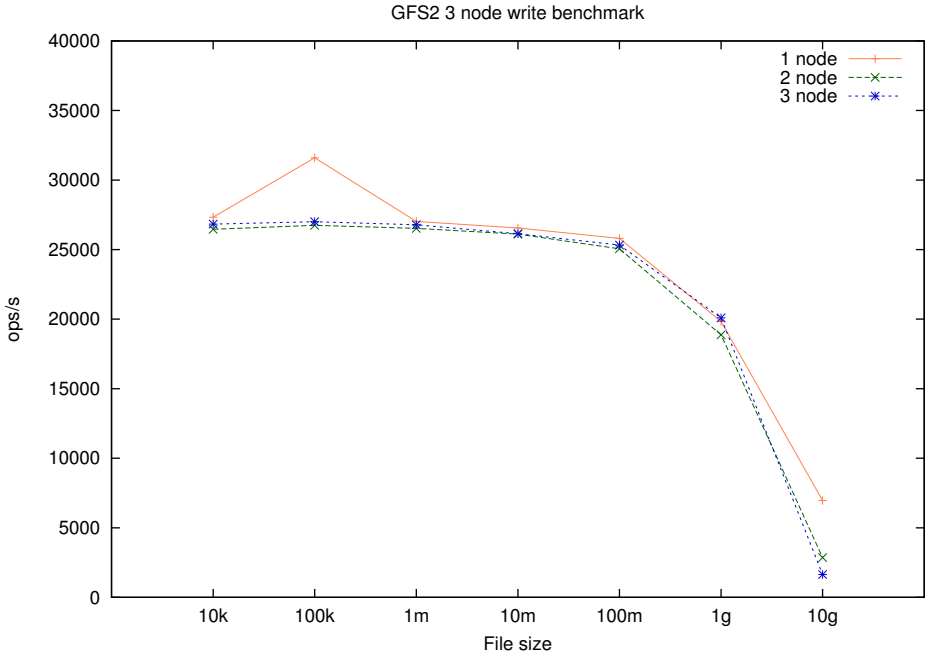

図 6.4: GFS2 から参照したノード数ごとの Random write 性能比較

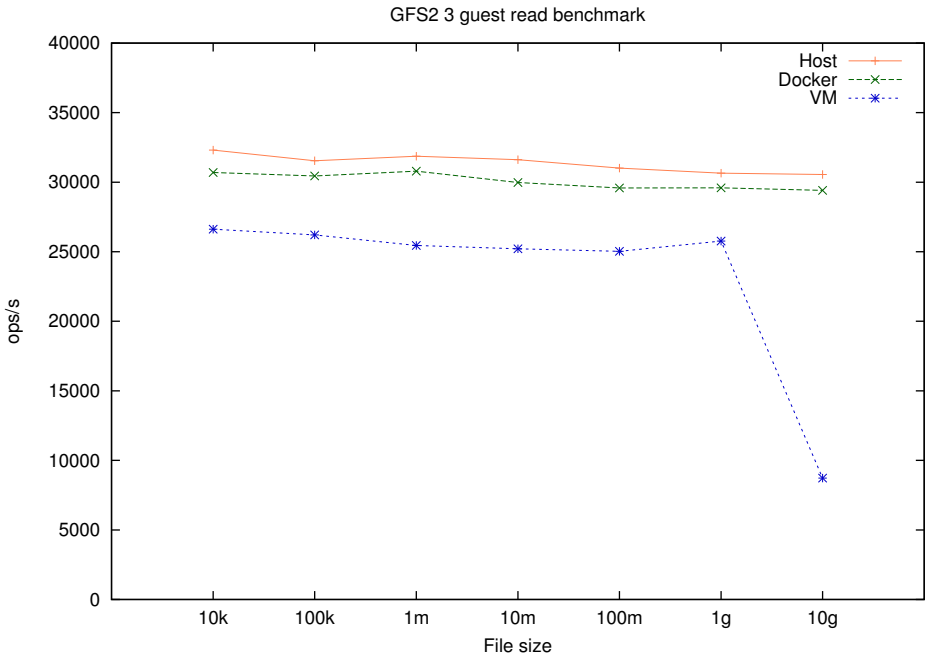

図 6.5: VM 、コンテナの Random read 性能比較

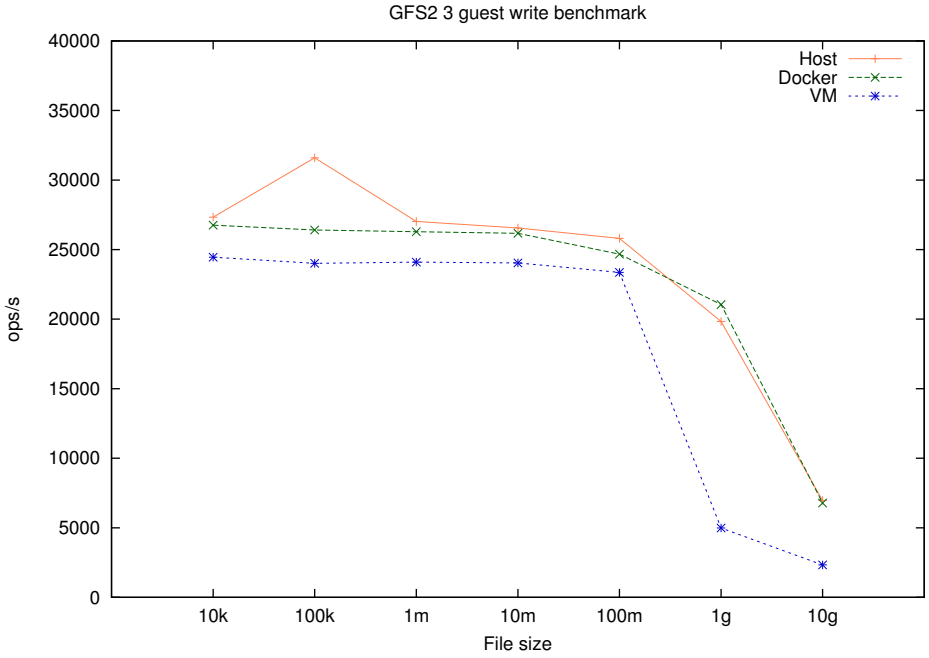

図 6.6: VM、コンテナの Random write 性能比較

6.5.1 考察

コンテナとホストからの計測は、Read と Write の性能ともに殆ど変わらないため、コンテナは直接の計算機からの Read や Write と変わらず使用することができる。

しかし VM に関しては、Read や Write ともに性能が低下している。Docker はホストと Linux kernel を共有しているため、IO が高速である。Docker と異なり VM はハードウエアもエミュレートしているため、速度が低下する。

Read と Write の多いアプリケーションを実装する場合はコンテナを使用するのがよい。VM は Kernel も独立に動作するため、Linux 以外の OS を動作させる場合に使用する。

6.6 授業 Operating System での使用

情報工学科では Operating System という授業が行われている。授業 Operating System で実際に ie-virsh を使用した。複数人の学生による VM の使用を、管理側の負担を抑えて管理することができた。

授業ではまず、受講生が自身の PC で VM を install し環境設定する。今回は Vagrant を使用した。Vagrant は Virtual Box を利用するため、VM image の形式も Virtual Box に則した ovf 形式である。設定した ovf 形式の VM image を Fibre Channel ストレージ上にアップロードし、KVM で起動することのできる形式 qcow2 へ変換する。

その VM image を元に ie-virsh で VM を起動する。

6.7 外部へ公開

本学科では学生に向けてグローバル IP アドレスを配布しており、ie-virsh で管理される VM はグローバル IP アドレスを割り当て、公開していた。

しかし授業 Operating System での使用の際、全ての VM にグローバル IP アドレスを割り当てていたため、学生に VM のセキュリティ設定が委ねられていた。

授業の課題のために起動し放置していた、外部からの攻撃に対して脆弱な VM があった場合は VM の停止を呼びかけるか、VM の Port やユーザ名、パスワードが脆弱でないか調査する仕組みが必要である。

6.8 Vagrant

Vagrant は異なる環境に移行可能な開発環境を構築・管理・配布することができる開発環境作成ツールである。手軽にテスト環境を導入・破棄することができ、変更が加わっても開発環境・本番環境に自動的に適用できる。

VirtaulBox などのプロバイダを使って、VM を Vagrant 経由で立ち上げる。手軽に起動・停止・ssh ログインできるため、Web サービスの開発や開発環境の配布などに利用される。

Vagrant は KVM をプロバイダとするプラグインを持っており、 KVM を VirtualBox のようにプロバイダとして動作させることができる。KVM 上の VM を Vagrant の操作と同じように起動・停止・設定することが可能となっている。

6.9 Vagrant Box

Vagrant で VM を利用する際に、VM のベースとなるイメージファイルが Vagrant Box である。 Vagrant で Vagrant Box を VM イメージとして起動し、設定し、開発環境を配布することができる。また配布されている Vagrant Box を取得して起動し、使用することができる。

6.10 Vagrant Box について

授業 Operating System では、ie-virsh の VM イメージに Vagrant Box を使用した。受講者の PC 上で Vagrant を使って開発環境を作成させ、その VM イメージをブレードサーバにアップロードして変換し、ie-virsh で起動する。そうすることで Vagrant で作成した開発環境を ie-virsh からそのまま使用することができる。

Vagrant Box イメージは簡易なパスワードとユーザ名で Vagrant から管理されていたため、そのままブレードサーバへアップロードしサーバ用途に使用してしまうと、簡単に外部から侵入され乗っ取られてしまう。そのため Vagrant Box イメージを使用する際は、外部から攻撃されないような設定か確認し、脆弱な設定なら設定し直す必要がある。

6.11 さくらクラウド

さくらインターネット株式会社 [2] の運営する、高性能なサーバと拡張性の高いネットワークをクラウド上に自在に構築できるパブリッククラウドサービスである。次期システムでクラウドサービスとして使用する予定となっている。

6.12 さくらクラウドへの VM イメージ送信

クラウドサービスへのサービスのデプロイ方法の一つとして、VM イメージを直接クラウドサービスにアップロードするという方法がある。

本学科のブレードサーバから、VM イメージをさくらインターネットの石狩リージョンへ向けて送信した。さくらクラウドへ VM イメージを送信する場合、まずアーカイブを作成する。作成後に FTPS の URL とホスト名・パスワードが表示される。その情報を使用して curl を使い FTPS 接続し送信する。

```
% curl --ftp-ssl -T fedora20.img ftp://username:password@hostname:21
```

表 6.2: VM イメージ送信

VM イメージサイズ	10.0 GB
VM イメージ形式	raw
送信速度	34.9 M
送信時間	00:04:53

送信時間は表 6.2 のようになる。

授業 Operating System で使用すると、一回で 60 名の受講生が VM をさくらクラウドへ向けて送信するため、現実的ではない。そのため VM 以外の方法を使う必要がある。

第7章　結論

本章では、本論文で述べたことをまとめ、また今後の課題を述べる。

7.1　まとめ

本研究ではまず始めに次期システムに対応し、さらに複数のユーザの資源を管理する教育用システムの要件を挙げた。次に本学科のシステム構成と現在使用されている OS 管理システムの使用方法の説明を行った。使われていたシステムは VMWare ESXi を用いた OS 管理システムと、ie-virsh を用いた OS 管理システムである。挙げた 2 つのシステムが始めに挙げた要件を満たしているか検証を行った。

Shien システムは iSCSI ディスク上の GFS2 を計算機が共有するという構成になっている。これまでの本学科のシステムと比べ、一定の資源を一度に配ることができ、またコンテナに対応している。ie-docker の実装を新たに行ったためである。次期システムのクラウドサービスをユーザに使用させるため、クラウドサービス上の資源をユーザにコンテナとして配布するという仕組みも述べた。

また Shien システムの操作方法や Shien システムを使った資源の管理方法を述べ、これまでのシステムとの違いやコンテナを使ったクラウドサービスへの対応についての説明を行った。

最後に GFS2 を他のファイルシステムと比較し、またホストサーバや VM 、コンテナから GFS2 のベンチマークを計測した。GFS2 を使用することで複数の計算機と VM イメージやデータの共有をシングルノード用の Filesystem と変わらず使用することができることを確認した。またコンテナを使用することで VM と比べ速い IO でアプリケーションを動作させられることを確認した。更に授業 Operating System で ie-virsh を使用した際の問題点の抽出と、評価を行った。

Shien システムを使用することで次期システムに対応することができ、クラウドサービスもユーザに使用させられることを確認した。

7.2　推奨するシステム

これまでの検証から、本研究で提案する Shien システムは次のようになる。Shien システムはオンプレミスとクラウドサービス上の使用に分かれている。

オンプレミスは、複数の計算機と iSCSI ディスクで構成される。iSCSI ディスクを GFS2 でフォーマットし計算機同士で共有する。共有した iSCSI ディスクには、VM イメージや

コンテナで動作させるアプリケーションを配置する。ie-virsh で VM の管理を、ie-docker でコンテナの管理を行う。ie-virsh と ie-docker は複数の計算機に対応していないため、VM 専用の計算機に ie-virsh を、他の計算機に ie-docker を導入する。

GFS2 でフォーマットされた iSCSI ディスクの共有は、本研究で検証を行った。その検証結果では複数の計算機から同時にアクセスしても別のファイルやディレクトリであれば、ホストのディスクにアクセスする速度と変わらない速度でアクセスすることができることが分かった。そのため ie-virsh と ie-docker をホスト上で直接動作させることと同様に使用できる。

ie-virsh では、VM にグローバル IP アドレスを使用することで VM 上に構築したサービスを公開することができる。また ie-docker で公開するためには、ie-docker が割り当てる Port を使用する。Proxy を介することで、Port とホストの IP アドレスの組み合わせに名前を割り当てることで外部に公開することができる。

クラウドサービスでは、ie-docker を使用する。オンプレミスで使用しているコンテナを、クラウドサービスへ移行する。Docker は Docker イメージを Build し、そのイメージをベースにコンテナを起動する。そのため Docker イメージを Build することで、VM イメージと比べて軽量なイメージをクラウドサービスとやり取りすることができる。またアプリケーションと Docker のイメージの Build に使われる Dockerfile を Repository へ登録し、クラウドサービスへアプリケーションを Commit し Push することで、そのアプリケーションをクラウドサービス上のコンテナで動作させる。

7.3 今後の課題

まず起動している VM やコンテナに対するセキュリティチェックの必要性があげられる。授業 Operating System で Vagrant Box を使用した例からも、脆弱なユーザ名やパスワードを使用するユーザは多い。そのため管理者側が定期的にユーザ名やパスワード、開いているポートをチェックし攻撃されてしまうような設定ならば通知を行う。このチェックは自動的に行えると良い。ssh を使ったユーザ名とパスワードのチェックをする場合、パスワードを 3000 個使用すると、1 つのユーザ名に関して二時間以上かかってしまうため、パスワード候補は更に少ない数を選択しなければならない。

次にコンテナと VM の使い分けを、複数の計算機上でどう行えばよいかを考える必要がある。Shien システムの ie-virsh や ie-docker の構築は、今回一つの計算機上で行った。しかし iSCSI ディスクを共有している計算機は複数あるため、複数の計算機上ではどういった使用が適切か計測し、使用方法を検討する必要がある。

また複数の計算機で ie-virsh を使用する際に、VM をどの計算機に配置するかを決定する機能が必要である。ie-virsh は現在一つの計算機で動くことを想定している。しかしユーザに配る VM を配置する計算機が複数台ある場合は、計算機のメモリ容量やディスク使用率を参照し VM を配置する計算機を選択する必要がある。

ie-docker をクラウドサービス上で使用させる場合、継続的インテグレーションに対応するとユーザの開発を補助することができる。具体的には Jenkins を各ユーザに ie-docker

と連携し使用可能にする。Jenkins は Jenkins ユーザを全ての Build に割り当てるため、ie-docker との協調方法を考えなければならない。

またクラウドサービスとオンプレミス環境の連携を考える必要がある。Shien システムではサービスのデプロイは可能だが、オンプレミス環境とクラウドサービス上のサービスを切り替えることはできない。アプリケーションのデータの同期手法を実装する必要がある。

謝辞

本研究を行うにあたりご多忙にも関わらず日頃より多くの助言、ご指導をいただきました河野真治准教授に心より感謝いたします。

研究を行うにあたり、環境の調整、意見、実装に協力いただいた並列信頼研究室の全てのメンバーに感謝いたします。

ブレードサーバ環境について相談に乗っていただいた長田智和助教授に感謝いたします。

最後に、大学の修士まで支えてくれた家族に深く感謝します。

参考文献

[1] 平良太貴, 河野真治. OS 授業向けマルチユーザ VM 環境の構築. 情報処理学会システムソフトウエアとオペレーティングシステム研究会 (OS), may 2014.

[2] さくらインターネット. http://www.sakura.ad.jp/.

発表履歴

- OS 授業向けマルチユーザ VM 環境の構築,
  平良太貴、河野真治 (琉球大学)
  情報処理学会システムソフトウェアとオペレーティングシステム研究会 (OS), May, 2014