IT 技術学習のための
教育用計算機システムの研究

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 技術の進歩は目まぐるしく、学習するためのコストや環境も必要になってくる。また学習のための環境を管理することは手間がかかる。

そのため、学生が IT 技術を学習するための環境を供したい。また提供した環境を使用させるにあたり、管理者への煩雑な手続きを無くすことで管理者の負担を減らさなければならない。管理者側の負担として、VM の作成やアップロード、また VM のセキュリティ管理が挙げられる。

またクラウドサービスと連携し、学生が開発したサービスをオンプレミスの環境から外のサービスへ移行することで、サービスへの高速なアクセスが可能になる。

そこで当研究室では、学生に VM を提供し、資源を管理するための ie-virsh を開発している。 ie-virsh は複数の hypervisor に対応した virsh の wrapper となっている。

また情報工学科では、来年度にシステム更新を行う。現在のシステム構成から得られた知見を活かし、新たな教育用のシステムを構築する必要がある。

本研究では、IT 技術を教育するための環境を学生に提供するために、教育用計算機システムの構築と評価を行った。本研究の教育用計算機システムに ie-virsh と、docker で使われる資源を管理するための ie-docker を開発し、学生に使用させる。またその際に Block device の共有に使う Filesystem 、 GFS2 の評価を行った。

現在のシステムと比較して、本研究で提案する計算機システムが教育用に適正であることを確認した。

Abstract

目 次

図 目 次

表 目 次

第1章　序論

IT 技術は Virtual Machine (以下 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にである。

そのため、学生が IT 技術を学習するための環境を提供したい。また提供した環境を使用させるにあたり、管理者への煩雑な手続きを無くすことで管理者の負担を減らさなければならない。管理者側の負担として、VM の作成や VM イメージのアップロード、VM のセキュリティ管理が挙げられる。

情報工学科では定期的にシステム構成の変更を行う。クラウドサービスと連携し、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、サービスへの高速なアクセスが可能になる。

そこで当研究室では、学生に VM を提供し、資源を管理するための ie-virsh を開発している。 ie-virsh は複数の hypervisor に対応した virsh のラッパーとなっている。

本研究では、IT 技術を教育するための環境を学生に提供するために、教育用計算機システムの構築と評価を行った。本研究では教育用計算機システムに、ie-virsh と、docker で使われる資源を管理するための ie-docker を開発し、学生に使用させる。またその際に Block device の共有に使う Filesystem 、 GFS2 の評価を行った。現在のシステムと比較して、本研究で提案する計算機システムが教育用に適正であることを確認した。

1.1　本論文の構成

第2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。

2.1　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。

2.2　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 で管理している。図 2.1 のように、VM の所有者はメールでシステム管理者へ VM 作成依頼を行い、本学科の提供する Web サービスで VM の起動・停止などの操作を行う。

2.3　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。

2.4　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

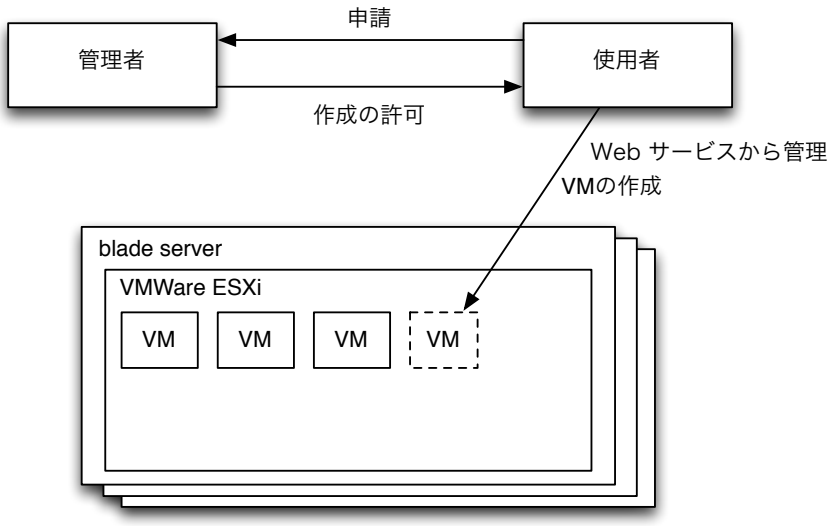

図 2.1: 本学科のシステム

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。

2.5 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 2.2 のようになっている。

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 管理は、主に VMWare ESXi が使われている。

2.6 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。

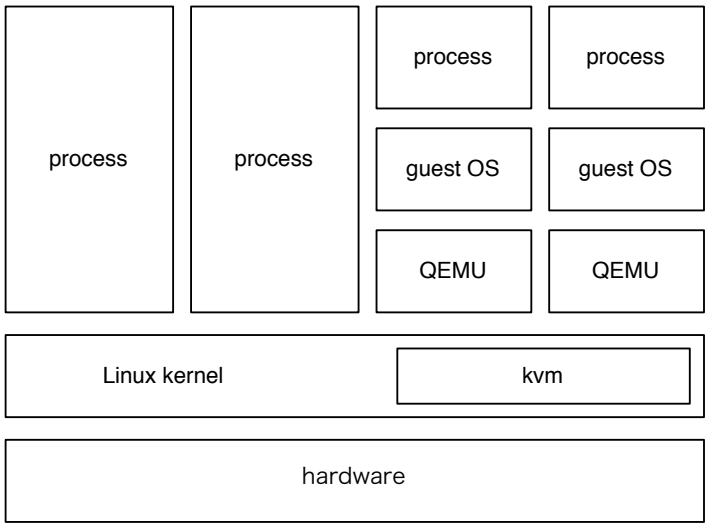

図 2.2: KVM architecture

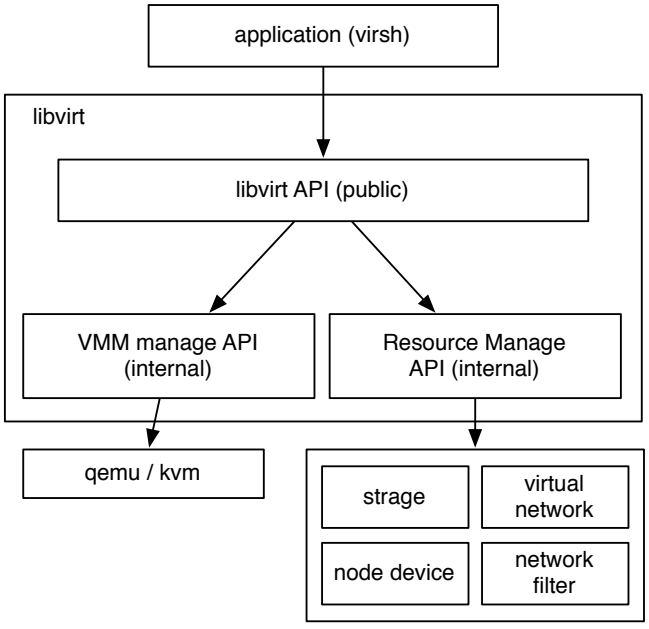

図 2.3: libvirt architecture

図 2.3 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。

2.7　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。

2.8　ie-virsh

virsh はグループの設定を行わない限り、root ユーザ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。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 2.1 が ie-virsh の機能である。

表 2.1: 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使用者にはネットワークやストレージの設定ができないよう実装している。

また ie-virsh には virsh で使用できる、ネットワークの構成の操作などの管理者側がするべき操作はない。管理者でないアカウントでは操作できない。

2.9　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。

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3章　Shien システム

本章では現在のシステムの問題点をあげ、Shien システムでその問題点を改善する方法を示す。

3.1　Multi User Support

情報工学科では一学年 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、複数の相手に同じ権限を、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操作は行うこともできない。

3.2　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。

Shien システムではユーザが使う資源を管理者が応対すること無く自動的に割り当て、制限できるよう構成する。

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。

3.3 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大きなデータ容量を配布できる。次期システムでは iSCSI で複数の計算機と記憶媒体と接続する。

そうすることで複数台の計算機で、VM イメージやユーザのデータを共有する。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。

そのため 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ファイルシステムを使用する必要がある。

3.4 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。情報工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。

Shien システムでは次期システム構成へ対応するため、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。

第4章　Shien システムの構成

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 システムは情報工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際に安定になっているもの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 Ret 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。

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。

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 The corosync Cluster Engine (corosync)

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。

4.5 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。

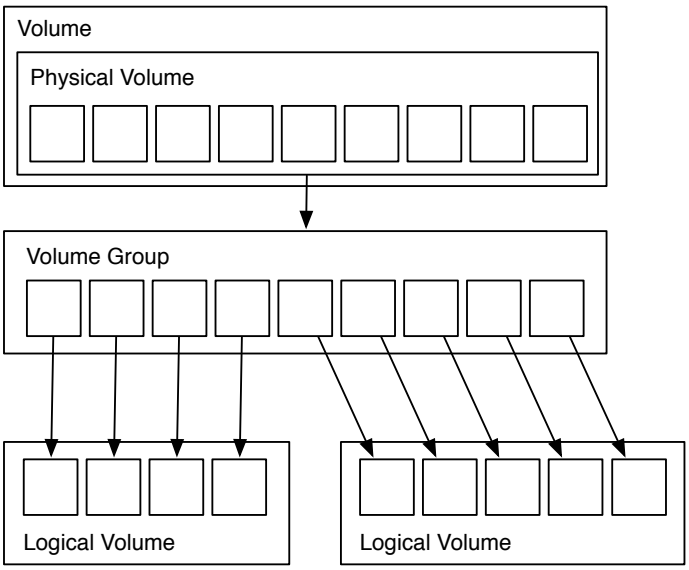

図 4.1: LVM

4.6 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。

クラスタで Volume Group を使用する場合は、Volume Group にクラスタに対応するフラグの設定を行わなければならない。

4.7 構成

複数のブレードサーバでクラスタを構成する。クラスタの構成には corosync を使う。

図 4.2 がブレードサーバ一台の構成である。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、Docker の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。

4.8 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。

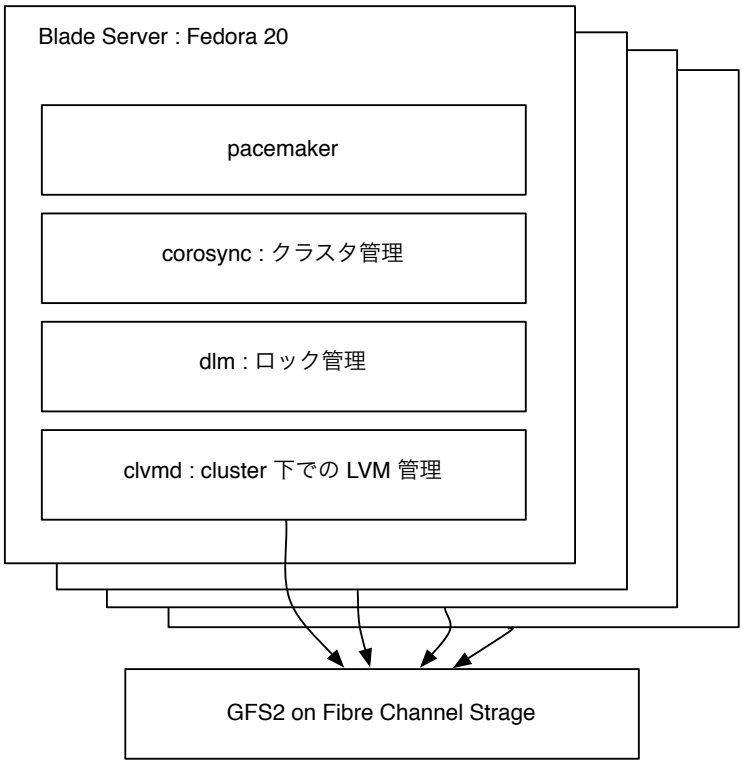

図 4.2: GFS2 Cluster

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。

4.9 ie-virsh debug

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。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。

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。そこで本研究のシステムに、gdb で Linux kernel を追う機能を追加する。

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 4.3 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。

4.10 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 4.4 は Docker のアーキテクチャである。

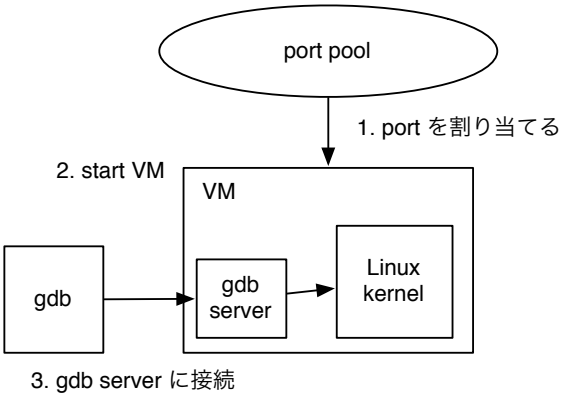

図 4.3: ie-virsh debug

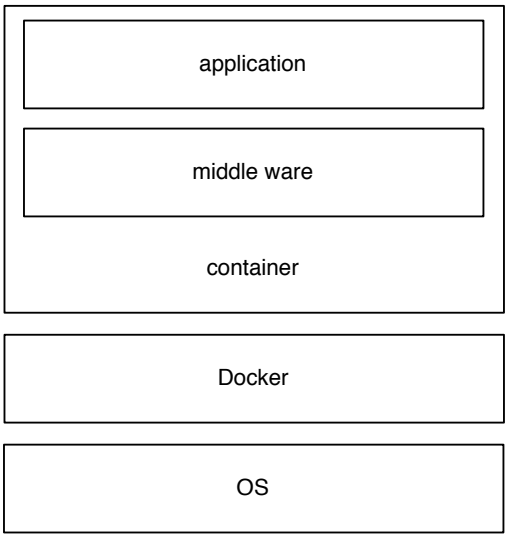

図 4.4: Container architecture

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。

4.11 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。

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

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 4.5 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。

4.12 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。

4.13 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。 ie-virsh 、ie-docker は LDAP を参照して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

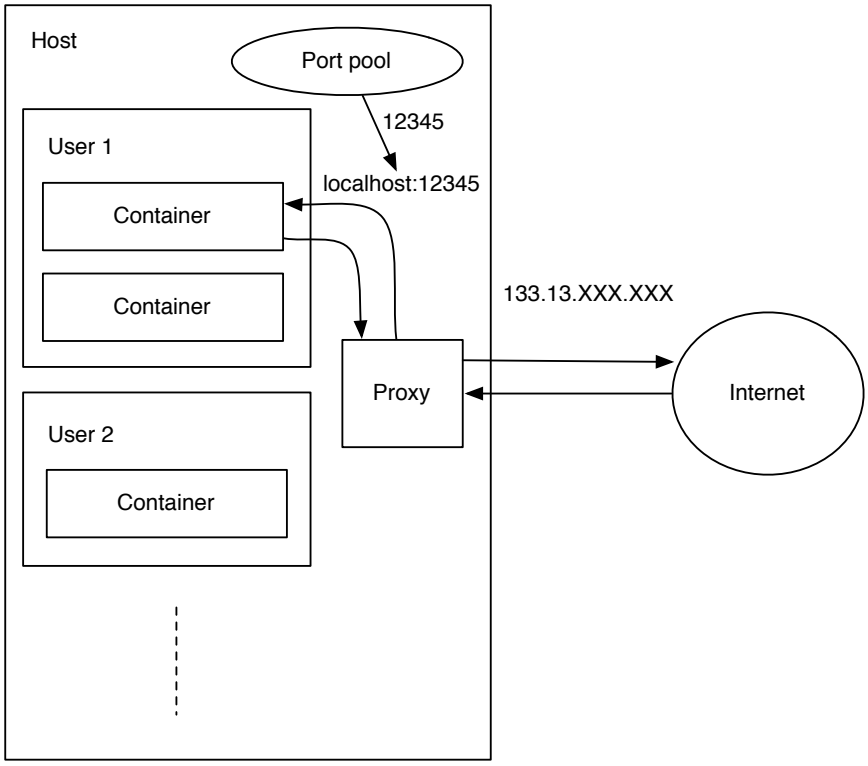

図 4.5: ie-docker による Port の付与

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。しかし各利用者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、権限は管理者と利用者のみに分けられる。そのため権限の配布は同一の権限を利用者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利用者アカウントを 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 Fibre Channel 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。ディレクトリに各利用者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することが可能である。

5.2　ie-virsh で使用する VM イメージのアップロード

利用者は Fibre Channel 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 Fibre Channel ストレージにアップロードする。アップロード先は Fibre Channel ストレージ上にある所有者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しなければならない。

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、そのコマンドを利用する。

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。

5.3 VM の起動

ie-virsh を使用して、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 virsh と同じである。

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

virsh の destroy コマンドをラップしているため、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

Kernel debug をするためには、下記のコマンドを実行する。

```
% ie-virsh debug
```

実行すると Port の Pool から Port を取得する。取得した Port を使用し Kernel debug のための VM が起動する。 VM の起動後に自動的に gdb が立ち上がる。取得した Port を使い、gdb から Kernel debug のための VM に接続する。

5.9 Docker の起動

本研究で新たに実装を行った ie-docker は Docker のラッパーとなっている。任意の Process name を下記のコマンドを使用して割り当てる。

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

Process name は任意に割り当てることができる。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、他のアカウントと被ることはない。

5.10 Docker からの iSCSI ストレージの使用

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 前にコンテナ上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 -it -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リポジトリを配置し、コンテナでそのリポジトリ上のアプリケーションを動作させる。

第6章　Shien システムの評価

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ものである。評価には現在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

また現在の、本学科の計算機システム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 の計測を行った。

Benchmark 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を用いて計測を行った。

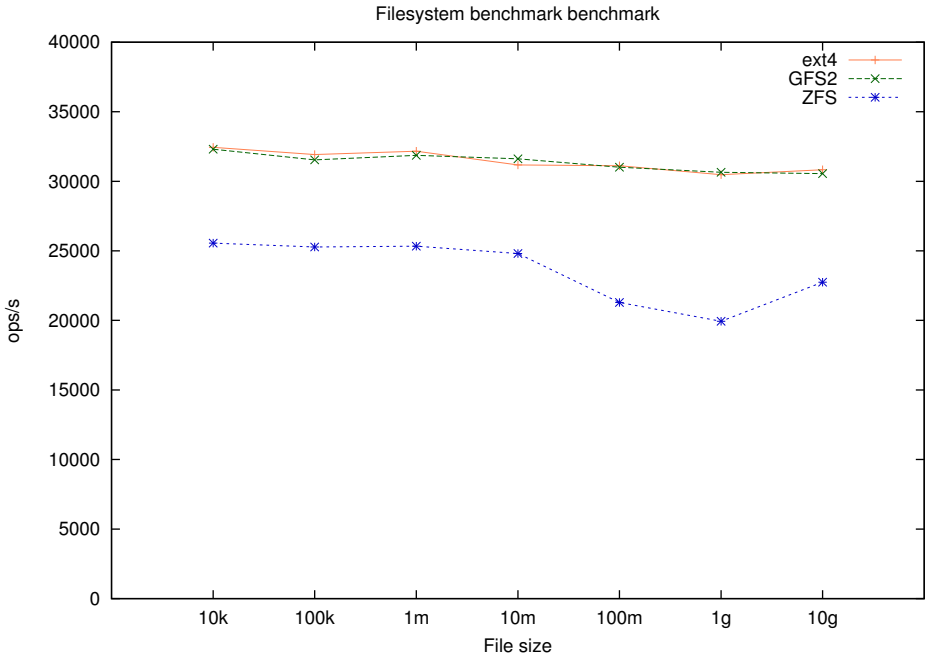

図 6.1: Filesystem ごとの Random read 性能比較

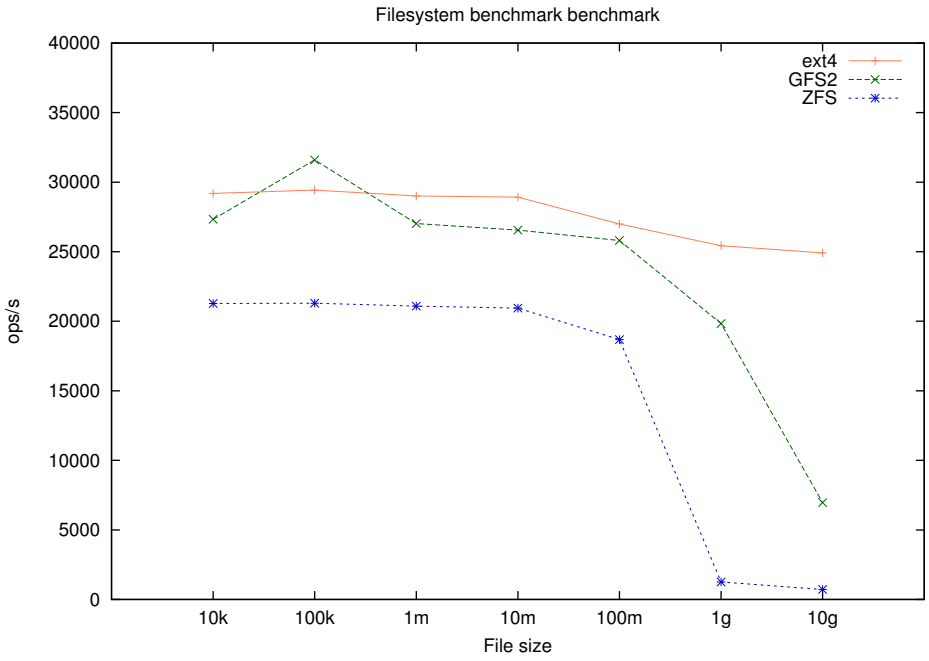

図 6.2: Filesystem ごとの Random write 性能比較

6.2 Filesystem の速度比較

6.2.1 考察

6.3 GFS2 上の複数ノードの計測

6.3.1 考察

6.4 GFS2 上の VM イメージを使った複数ノードの計測

6.4.1 考察

6.5 VM とコンテナとの比較

6.5.1 考察

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 Cloud 上での使用

6.12 VM イメージの転送速度

6.13 コンテナの使用

第7章　結論

7.1　まとめ

7.2　今後の課題

謝辞

参考文献

[1] Messagepack. http://msgpack.org/.

[2] 大城信康, 河野真治. Data segment の分散データベースへの応用. 日本ソフトウェア科学会, September 2013.

[3] 玉城将士, 河野真治. Cassandra を使ったスケーラビリティのある cms の設計. 情報処理学会, March 2011.

[4] 玉城将士, 河野真治. Cassandra と非破壊的構造を用いた cms のスケーラビリティ検証環境の構築. 日本ソフトウェア科学会, August 2011.

[5] Avinash Lakshman and Prashant Malik. Cassandra - a decentralized structured storage system. *LADIS*, Mar 2003.

[6] Fay Changand Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable : A distributed storege system for structured data.

[7] Nancy Lynch and Seth Gilbert. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 2002.

[8] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon's highly avaliable key-value store.

[9] 所眞理雄. DEOS プロジェクト研究成果集 Dependability Engineering for Open Systems, 2013.

[10] 永山 辰巳, 横手 靖彦. オープンシステムディペンダビリティと D-Case を繋ぐリポジトリ, 2013.

発表履歴

-