令和2年度 卒業論文

コンテナ技術を用いた教育計算機システムの構築

琉球大学工学部工学科知能情報コース

175733E 氏名 宮平 賢

指導教員：河野 真治

目 次

図 目 次

表 目 次

ソースコード目次

第1章　はじめに

情報通信技術の普及に伴い学生が学ぶ学習環境が必要となる。その学習環境として VM や コンテナにより, 手軽に開発し試せる技術が普及している。だが, 手元の PC 上で VM や コンテナを立ち上げ, 開発を行うことはできるが, VM や コンテナの使用には高性能 PC や 有料のクラウドサービスが必要になる場合がある。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。

本コースでは希望する学生に学科の汎用サーバから仮想環境を貸出するサービスを行っている。貸出をする VM の基本スペックとして CPU 1 コア, メモリ 1GB, ストレージ 10GB である。基本スペックでは不足する場合は要望に応じてスペックの変更を行っている。しかし, 機械学習などの演習では CPU より GPU が求められる場合がある。VM 上で GPU を共有するには PCI パススルーを利用することで可能である。だが, PCI パススルーでは GPU と VM は 1 対 1 の関係となり, GPU を希望する利用者すべてに割り当てることはできない。

本研究では, 学生が貸出 VM だけでなく, 学科の汎用サーバのリソースを効率的に利用できる教育計算機システムを提案する。教育計算機システムには複数の汎用サーバと大容量ストレージサーバが存在する。複数のサーバを利用するにあたり, 分散ストレージが必要となる。また, 学習環境として利用されることから, 複数の並列なアクセスに耐えられ, 信頼性の高いファイルシステムが必要である。この用件を満たすストレージソフトウェアとして Ceph を採用した。汎用サーバのリソースを効率的に利用するために, コンテナエンジンである Podman, Singularity, ジョブスケジューラである Slurm を採用した。これらのソフトウェアを合わせ教育計算機システムの構築を行った。

1.1　システム管理チーム

本コースで利用されている教育情報システムの運用管理は, 平成 24 年まで演習科目の 1 つとして行われてきた [1]。しかし, サービスの多様化やシステムの高度化により, 演習科目として行うには困難になった。そこで, 平成 25 年度に学生と教職員らの有志による「システム管理チーム」が発足した。本チームはシステムの構築, 運用管理やシステム利用者のサポートを行っている。

1.2　論文の構成

本論文では, 6 章で構成され, いかに各章の詳細を示す。

- 第 1 章は, 本研究の背景と目的を述べる
- 第 2 章は, 本論文で必要な技術概要を述べる
- 第 3 章は, 教育計算機システムの構築について述べる
- 第 4 章は, 教育計算機システムの管理と利用方法について述べる
- 第 5 章は, 教育計算機システムの評価について述べる
- 第 6 章は, 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　技術概要

本章では, 本研究で使われる技術, 本コースで利用しているサービスについて概要を説明する。

2.1　仮想化

仮想化はコンピュータの CPU やメモリ, ディスクなどハードウェアのリソースを分割又は統合して, 仮想的なコンピュータやネットワーク環境を生成し提供する技術である。仮想化技術にはホストのどの部分から仮想化するかによってホスト型, ハイパーバイザー型, コンテナ型に分けることができる。

2.1.1　ホスト型

ホスト型の仮想化は, ホストとなる OS 上 (以下, ホスト OS) に仮想化ソフトウェアをインストールし, 仮想化ソフトウェア上で別の OS (以下, ゲスト OS) を稼働させる手法である (図 2.1)。仮想化ソフトウェアをホスト OS のアプリケーションの 1 つとして導入及び管理できるため, 手軽に仮想化を実現することができる。しかし, ゲスト OS の処理はホスト OS を経由しなければならないため, オーバーヘッドが大きくなる。

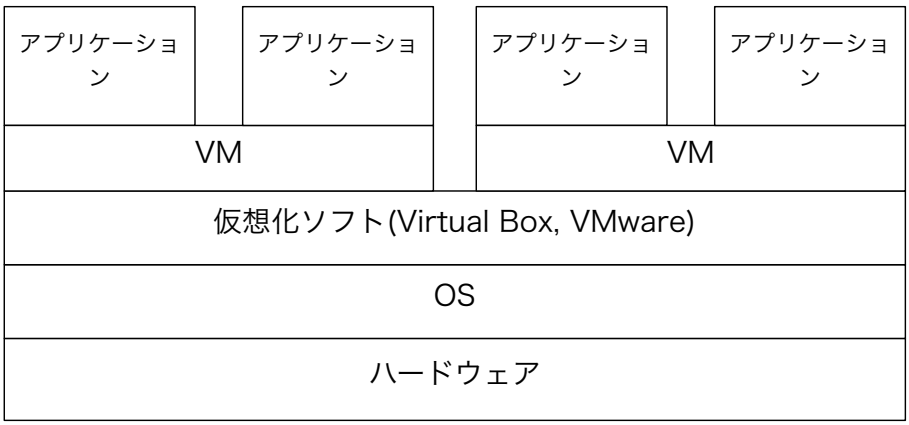

図 2.1: ホスト型

2.1.2　ハイパーバイザー型

ハイパーバイザー型の仮想化は, 仮想化システムを直接ハードウェアにインストールし, ハイパーバイザー上で複数のゲスト OS を稼働させる手法である (図 2.2)。ハイパーバイザーが直接ハードウェアを管理するため仮想化によるオーバーヘッドを小さくすることで, リソースを効率的に利用することができる。

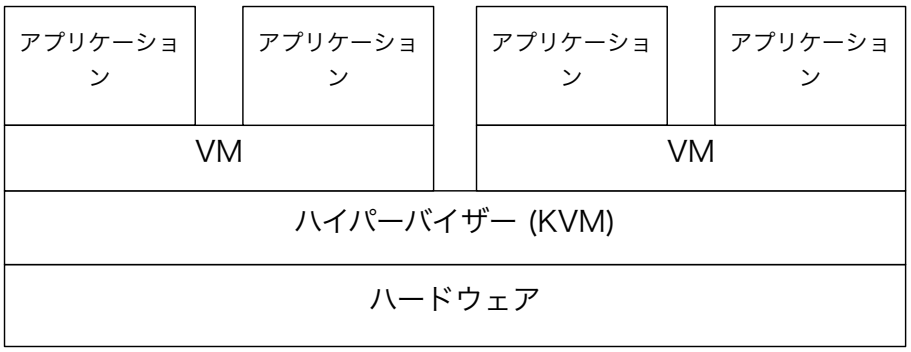

図 2.2: ハイパーバイザー型

2.1.3　コンテナ型

コンテナ型の仮想化は, OS レベルの仮想化技術を利用して複数のコンテナと呼ばれる独立空間を形成し, 独立空間でアプリケーションをそれぞれ構築することができる手法である (図 2.3)。各コンテンはオペレーティングシステムカーネルによって独立したプロセスとして実行される。前述のホスト型やハイパーバイザー型と比べ, コンテナはゲスト OS を起動することなくアプリケーションを実行することができるため, リソース効率が良く処理が軽量である。

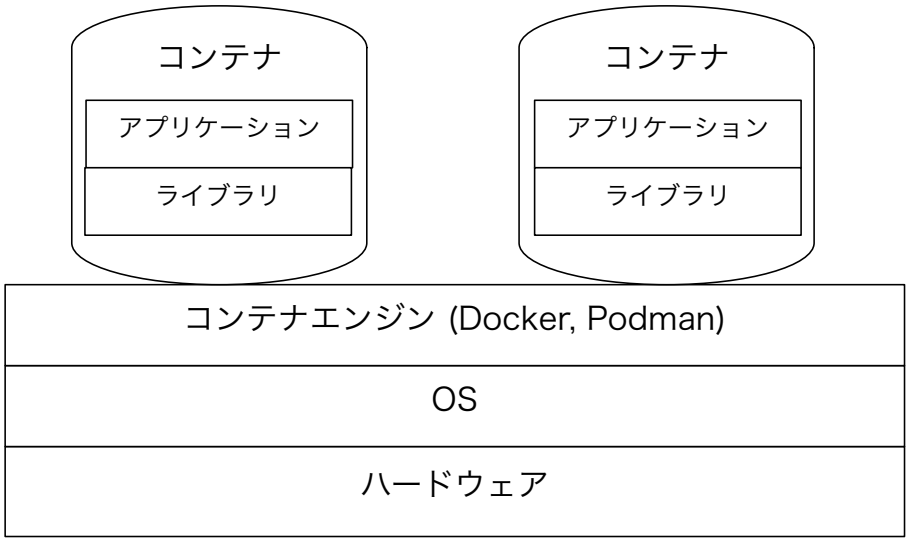

図 2.3: コンテナ型

2.2 KVM

KVM (Kernel-based Virtual Machine)[2] は Linux カーネル 2.6.20 以降に標準搭載されているハイパーバイザーである。KVM は Intel VT 及び AMD-V を含む x86 ハードウェア上の完全仮想化をサポートしている。KVM はハイパーバイザーと各仮想マシン間のレイヤーとして Virtio API を使用して, 仮想マシンに準仮想化デバイスを提供する。これにより, 仮想化によるオーバーヘッドを少なくできる。

2.3 Docker

Docker[3] は Docker 社が開発, 提供する Linux 上で動作する隔離された Linux コンテナをデプロイ, 実行するアプリケーションである。Docker はコンテナを実行するだけでなく, コンテナイメージの作成や共有する仕組みも提供している。Docker コマンドを処理するには Docker daemon と呼ばれるデーモンプロセスを実行する必要がある。この Docker deamon は Docker で行う処理を一箇所で実施する (図 2.4)。

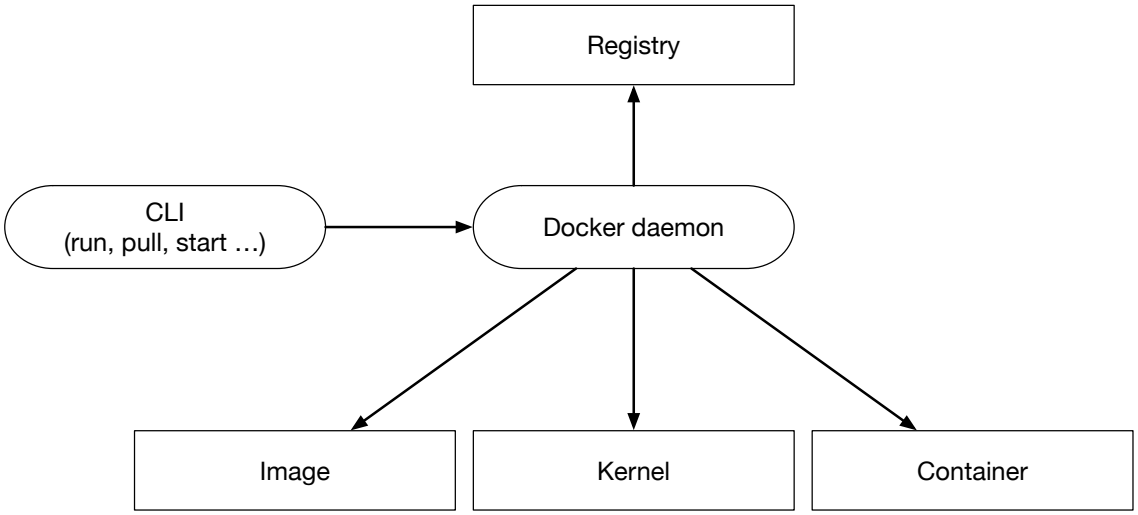

図 2.4: Docker

2.3.1 Docker Registry

Docker Registry は Dcoker イメージを保存, 配布できるサーバサイドアプリケーションである [4]。以下の場合に利用される。

- イメージの保存場所を厳密に管理する
- イメージを配布するパイプラインを全て所有する
- イメージの保存と配布を社内や学内の開発ワークフローに密に統合する

2.4 Podman

Podman は RedHat 社が開発, 提供する Linux 上で OCI コンテナを開発, 管理, 実行するためのデーモンレスコンテナエンジンである [5]。Podman は OCI 準拠のコンテナランタイムに依存するため, 前述した Docker など他のコンテナエンジンと互換性を持つ。また, Podman CLI は Docker CLI と同じ機能を提供する。Podman はコンテナとイメージストレージ, コンテナランタイムを介して Linxu カーネルと直接対話することで, デーモンレスで実行される (図 2.5)。Podman の制御下にあるコンテナは, 特権ユーザ又は非特権ユーザのいずれかによって実行することができる。

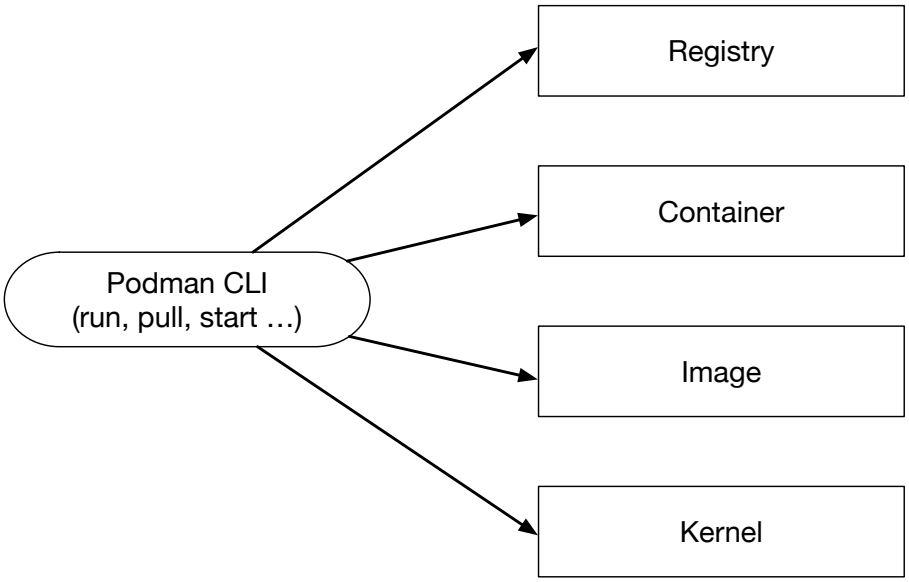

図 2.5: Podman

2.5 Singularity

Singularity[6] とは, HPC 環境向けに設計されたコンテナプラットフォームである。Singularity は マルチユーザに対応しており，コンテナ内での権限は実行ユーザの権限を引き継ぐため，ユーザに特別な権限の設定が必要ない。またデフォルトで, $HOME, /tmp, /proc, /sys, /dev がコンテナにマウントされ, サーバ上の GPU を簡単に利用できる。コンテナイメージは Singularity Image Format (以下, sif) と呼ばれる単一ファイルベースのため, アーカイブや共有が容易である。

2.6 Ceph

Ceph は, RedHat 社が開発, 提供する分散ファイルシステムである。Ceph は分散オブジェクトストレージである RADOS (Reliable Autonomic Distributred Object Storage)

がベースとなっている (図 2.6)。オブジェクトストレージはデータをオブジェクトという単位でやり取りをするストレージシステムである。複数のストレージを束ねて利用できるオブジェクトストレージが分散オブジェクトストレージである。RAODS では, Object Storage Daemon (OSD) にデータ格納する。オブジェクトの配置には, クラスタマップを元に Controlled Replication Under Scalable Hashing (CRUSH) アルゴリズムによりオブジェクトの格納先を選択する。配置の計算に必要とする情報はごくわずかであるため, Ceph クラスタ内のすべてのノードは保存されている位置を計算できる。そのため, データの読み書きが効率化される。また, CRUSH はデータをクラスタ内のすべてのノードに均等に分散しようとする。

RODOS はクラスタに保存されるデータの管理を待ち受け, 保存オブジェクトへのアクセス方法として Object Gateway, RADOS Block Device (以下, RBD), CephFS がある。Object Gateway は HTTP REST 経由でクラスタに保存されるオブジェクトへ直接アクセスが可能である。RBD はブロックデバイスとしてアクセスが可能で, libvirt を組み合わせて VM のディスクとして使用できる。また, RBD ドライバを搭載した OS にマップし ext4 や XFS などでフォーマットして利用できる。CephFS は POSIX 互換のファイルシステムである。

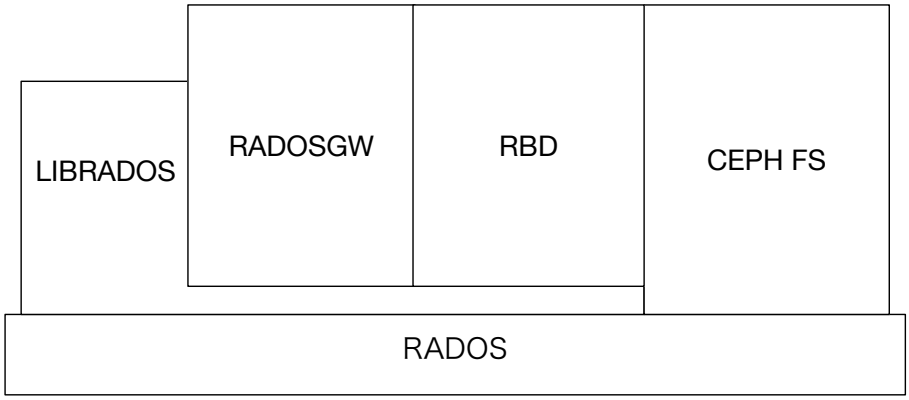

図 2.6: Ceph のアーキテクチャ

Ceph では, ノードとはクラスタを構成するサーバであり, ノードでは以下の 4 つのデーモンが実行できる。

- Ceph Monitor
- Ceph OSD
- Ceph Manager
- Ceph Metadata Server

2.6.1 Ceph Monitor

Ceph Monitor (以下, MON) ノードはクラスタのヘルス状態に関する情報, データ分散ルールを維持する。障害が発生した場合, クラスタ内の MON ノードで Paxos という合意アルゴリズムを使用して, どの情報が正しいかを多数決で決定する。そのため, 奇数個の MON ノードを設定する必要がある。

2.6.2 Ceph OSD

Ceph OSD (以下, OSD) は物理ストレージになる。このデーモンは 1 台の HDD などの物理ストレージに対して, 1 つのデーモンが動作する。OSD は MON と通信し, OSD デーモンの状態を提供する。

2.6.3 Ceph Manager

Ceph Manager (以下, MGR) ノードはクラスタ全体から状態情報を収集する。MGR は MON と共に動作し, 外部のモニタリングシステムや管理システムのインターフェースとして機能する。

2.6.4 Ceph Metadata Server

Ceph Metadata Server (以下, MDS) ノードは CephFS のメタデータを保存する。

2.7 Ansible

Ansible[8] は RedHat 社が開発, 提供するシステム構成, ソフトウェアの展開などを行う自動化ツールである。あらかじめ用意した設定ファイルに従ってソフトウェアのインストールや設定を自動的に実行できるため, コンピュータクラスタを構築する際に時間の短縮やミスの削減に有用である。Ansible の特徴としてエージェントレスがある。構成管理を行う機器が Python が使用可能で SSH で疎通することが可能であれば対象とすることができる。Ansible の一連の処理は Playbook という単位にまとまられ, YAML 形式で記述される。YAML 形式で記述されていることで, 可読性が高く学習が容易である。また, インフラストラクチャをコードとして残すことができる。

2.8 Slurm

Slurm[9] は Linux クラスタ向けのフォールトトレラント設計のジョブスケジューリングシステムです。Slurm には以下の 3 つの主要機能を提供する。

- 計算を実行するユーザに対してリソースへの排他的, 非排他的なアクセスを割り当てる
- 割り当てられたノード上のジョブの開始, 実行, モニタリングを行う
- 待機中のジョブキューを管理することにより, リソースの競合を解決する

Slurm では主に slurmctld と slurmd で構成される (図 2.7)。また, slurmdbd を有効にすることで, データベースへアカウンティング情報を記録することができる。アカウンティング情報を記録することで, ジョブの優先度を調整することが可能となる。

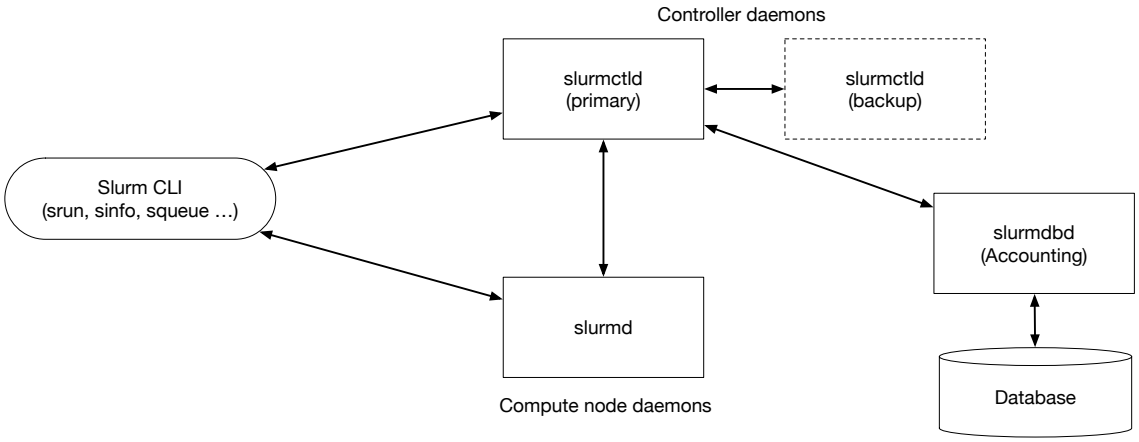

図 2.7: Slurm のアーキテクチャ

2.9 rsnapshot

rsnapshot[10] は rsycn に基づく差分バックアップユーティリティである。ローカルマシンやリモートマシンのスナップショットを取ることができる。リモートマシンとは SSH 経由で通信を行う。rsnapshot は設定された数のスナップショットを保持するため, 使用されるディスク領域は継続的に増加することはない。データの復元にはバックアップの保存先から rsync などを用いてコピーを行うことで, 特定のファイルの復旧などにも迅速に対応できる。バックアップを自動化するには cron などと併用する必要がある。

2.10 Akatsuki

Akatsuki は本コースで利用している VM 貸出システム, 有線 LAN 接続サービス, 内部 DNS の機能を提供する Web コントロールパネルである。Ruby で記述されており, フレームワークとし Ruby on Rails を採用している。本コースの学生は学科のアカウントでログインし VM の作成などを行う。現在はシステム管理チームが管理, 保守を行っている。

2.11 ie-virsh

ie-virsh[11] は本コースで利用している virsh をラップした VM 管理ツールである。ユーザの UID 及び GID 情報を使用し, 他のユーザ VM を操作させない仕組みを持つ。ie-virsh は VM 管理だけでなく, Linux Kernel のデバッグを行うことができる。そのため, 本コースの Operating System という授業で, OS について学ぶ一環として課題で利用されている。現在はシステム管理チームが管理, 保守を行っている。

第3章　旧システム

本章では, 2020年8月まで使用されていたシステムの環境, 演習や研究用に利用できるVM管理システムについて述べる。

3.1　オンプレミス環境

旧システムは, KVMを利用したVMベースのシステムを構築していた。VMは本コースのWebやDNS等の基幹システムや, 学生が演習や研究用で利用できる貸出VMで利用されていた。そのため, 利用者が必要とする十分なスペックを提供するため, 表3.1のスペックの汎用サーバを4台導入した。

表 3.1: 旧システムの物理サーバ

CPU	Intel Xeon E5-2699 v3 (2.30GHz/18Core)
CPUユニット数	2
メモリ	768GB
HDD	600GB

次にVMのイメージを保存するために表3.2のストレージを2台導入した。ハードディスクドライブの故障が想定されるため, RAID6を採用し信頼性及び可用性の向上を行った。ストレージと汎用サーバとの接続プロトコルはiSCSIを採用した。KVMは標準でライブマイグレーションに対応している。そこで, クラスタファイルシステムとして利用可能なファイルシステムである, GFS2を採用した。

表 3.2: 旧システムのSAN用ストレージ

HDD	SAS 1.2TB x 24
回転数	15000rpm
RAID	6
実行容量	19.7TB

最後にシステムのバックアップを行うために表3.3の大容量ストレージを2台導入した。大容量ストレージには本コースのWebやデータベース, ユーザのホームディレクトリなどを月に一度バックアップを行う。

表 3.3: 旧システムの汎用ストレージ

HDD	SAS 4.0TB x 24
回転数	7200rpm
RAID	6
実行容量	68.5TB

3.1.1 Akatsuki

Web コントロールパネルから有線 LAN 接続サービスや VM 貸出サービスを管理している。利用者はシステム管理チームへ VM の利用申請を行い, VM 作成の権限を取得する。権限を取得後, Web コントロールパネルより VM 作成, 電源操作を行えるようになっている。VM のリソースは CPU1 コア, メモリ 1GB, ストレージ 10GB となり, 申請を行うことでリソースを増やすことができる。VM 貸出サービスの概要を図 3.1 に示す。

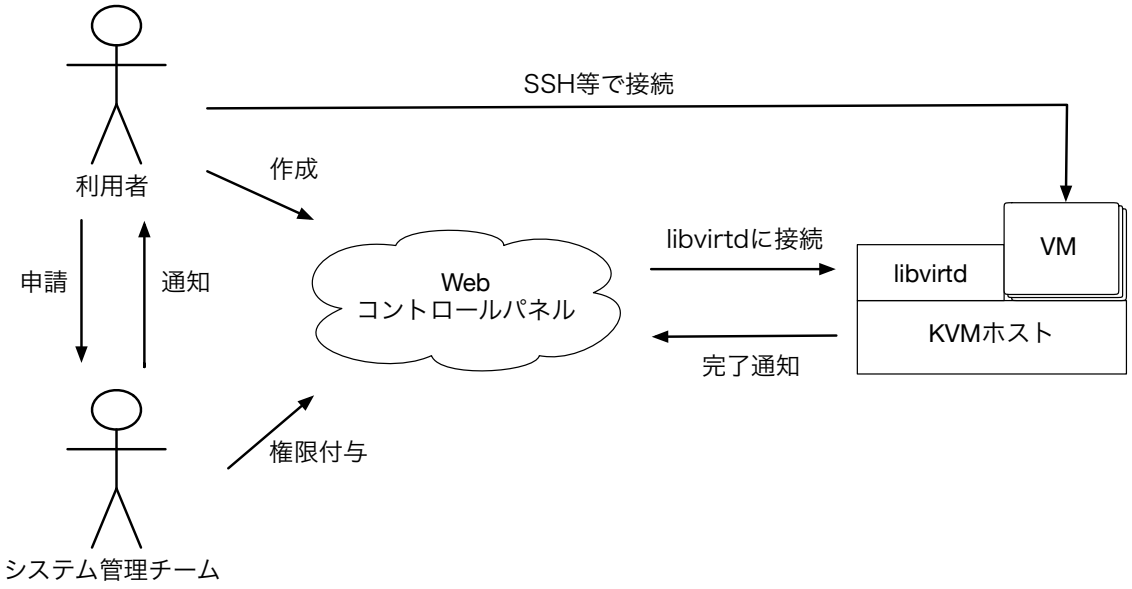

図 3.1: Akatsuki の概要

3.1.2 ie-virsh

ie-virsh は手元の PC で作成した VM を学科のブレードサーバにデプロイできるサービスである。ユーザの UID 及び GID 情報を取得することで, マルチユーザに対応している。表 3.4 は ユーザが利用できる ie-virsh の機能である。ie-virsh は手元の PC で作成した VM を実行できるため, ユーザが好みの OS や環境を構築できるなど自由度が高い。

表 3.4: ie-virsh のコマンド

define	XML の template を下に domain を作成
undefine	define で作成した domain を削除
list	define で作成した domain の一覧表示
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

3.1.3 ie-docker

ie-docker は Docker をラップしたツールであり, ユーザは学科のブレードサーバへ ssh で接続を行い CUI から利用することができる。ie-virsh と同じく, ユーザの UID 及び GID 情報を取得することで, マルチユーザに対応している。表 3.5 は ie-docker で利用できる機能である。コンテナで使用するイメージを管理者が用意する必要がある。

表 3.5: ie-docker のコマンド

ps	起動中のコンテナの一覧を表示する
run	コンテナを作成する
start	コンテナを起動する
stop	コンテナを停止する
attach	起動しているコンテナに attach する
cp	コンテナにファイルを送信する
rm	コンテナを削除する

3.1.4 問題点

旧システムでは, 学生が演習などで利用できる環境として貸出 VM のみであった。そのため以下のような問題が生じた。

- 仮想環境の貸出サービスにおいて, 新しく仮想環境を作成するにはシステム管理チームへ申請が必要であった。そのため, 一部学生は申請の方法が分からなかったり, 貸出サービスがあることが周知されていなかったため, 旧システムのリソースが余っていた。
- 機械学習の演習では GPU が求められる。だが, 旧システムには GPU が搭載されていないため, 要求されるリソースを提供できない。そのため, 貸出サービスではなく研究室ごとの機器が多く利用された。

第4章　教育計算機システムの構築

本章では, 2020 年 9 月に行われたシステム更新, 演習や研究用に利用できる仮想環境について述べる。。

4.1　新システムのオンプレミス環境

新システムでは, 表 4.1 の汎用サーバを 4 台採用した。旧システムのストレージは HDD であったが, SSD の大容量化, 低価格化により SSD を搭載した。また, 演習や研究等で利用できるよう GPU も搭載した。

表 4.1: 新システムの物理サーバ

CPU	Intel Xeon Gold 6238 (2.10GHz/22Core)
CPU ユニット数	2
GPU	Nvidia Tesla V100S
メモリ	512GB
SAS SSD	5TB
NVMe SSD	1.5TB

次にユーザのデータなどを補完するために, 表 4.2 のストレージサーバを 2 台採用した。2 台のストレージサーバには Ceph を構築するため, RAID を構成せず利用する。そのため, 旧システムでは全体容量が 40TB だったが, 新システムでは 90TB と増加した。

表 4.2: 新システムのストレージサーバ

CPU	Intel Xeon Silver 4208
メモリ	32GB
SAS HDD	300GB/15000rpm x 2
NLSAS HDD	4TB/7200rpm x 12

4.1.1　Virtual Machine

旧システムでは VM ベースでシステムを構築していたが, 新システムではコンテナベースでの構築行った。しかし, VM 貸出サービスである Akatsuki, ie-virsh は利用を継続する。

また, ie-virsh は新たに以下の機能を追加した。

- 手元の PC の VM をデプロイするだけでなく, VM のテンプレートから差分で新しく VM を作成する
- 作成した VM のリソースを変更する

新システムでは旧サーバと比べディスク容量が増加したため, VM イメージを汎用サーバのディスクドライブに保存することで, VM の起動速度を高速化を図ることができる。旧システムでは VM の作成は申請が必要であったが, 利用者は申請をせず VM を作成できるように機能を追加した。しかし, 利用者が制限なく VM を作成するとディスクリソースを圧迫する恐れがある。そこで, VM の作成にはクローンではなく差分で作成することで, VM イメージサイズを小さくすることができる。

4.1.2 コンテナ

新システムでも VM 貸出サービスを継続するが, 新しく搭載される GPU を VM で利用するためには PCI パススルーなどの設定が必要となる。しかし, PCI パススルーでは, VM と GPU が 1 対 1 の関係になるため, GPU 希望する利用者全てに割り当てることができない。また, 貸出 VM は利用者の好み環境構築ができる反面, VM を作成するごとに同じような作業が必要となり利用者の手間となる。そこで, アプリケーションの実行環境として採用されているコンテナ技術を利用する。

システムは学生や教授などが利用するため, マルチユーザで利用できるコンテナエンジンが必要となる。そのため, コンテナエンジンにはマルチユーザに対応している Podman と Singularity を採用する。Podman は開発段階でもあるため一部機能が不安定だったり, 設定が上書きされる場合がある。管理するシステム管理チームの学生の教育には適しているが, 演習や研究用で利用するには適さない場合がある。そのため, HPC 環境に設計されている Singularity も同時に利用する。

Singularity はコンテナ内で実行ユーザの権限を引き継ぐため, 利用者が作成したプログラムの実行には向いている。だが, Web など特権が必要なサービスを実行することはできない。特権が必要な Web などをを実行する場合は Podman を利用する。Podman はネットワーク設定を行うことで, コンテナ個別に IP アドレスを割り当てることができるが, ルートレスでは割り当てができない。IP アドレスの割り当てにはネットワークデバイスの関連付けが必要だが, root 権限が必要なためである。rootless で Web などのサービスを実行しアクセスするにはポートフォワードを設定する必要がある。だが, 利用者が使用するポートを汎用サーバで開放することはセキュリティ的にできない。そこで, podman を wrapper した ie-podman を作成した。ie-podman はコンテナに個別の IP アドレスを割り当てる際に利用する。

4.1.3 ファイルシステム

旧システムではVMのイメージをクラスタファイルシステムであるGFS2に保存し運用していた。このGFS2の運用には別途クラスタを構成する必要があるため, 単一障害が発生により多くのサービスに影響を与えることがあった。また, ユーザのホームディレクトリもVMでGFS2をマウントしNFSで提供されていた。そのため, NFSを提供するVMが停止することでユーザへの影響があった。そこで, 新システムではVMイメージの保存には汎用サーバのディスクドライブ, ユーザのホームディレクトリにCephを採用する。

新システムでは汎用サーバにSAS SSDが5TBと旧システムより多く搭載されている。2台のサーバに演習や研究用で利用する貸出VMのイメージを保存し, 残り2台には本コースで利用しているサービスを提供するVMを保存する。汎用サーバに保存することで, 単一障害時の影響を小さくすることができる。Cephは自己修復と自己管理機能を持つため, 信頼性の高いファイルシステムとして利用できる。そのため, ユーザのホームディレクトリを配置するファイルシステムとして利用する。また, CephはObject Gateway, ブロックデバイス, POSIX互換のファイルシステムなど, 用途によって柔軟にアクセス方法を変更できる。ブロックデバイスとしてアクセスすることでVMイメージのバックアップとしても利用できる。

4.1.4 構成

新システムでは, 各サーバに演習や研究用で利用できるPodmanとSingularityを用い, ジョブスケジューラであるSlurmを用いて管理を行う。汎用サーバ1台をSlurmのコントローラ/計算ノードとし, 残りは計算ノードとすることで, システムのリソースを最大限利用可能にする。Cephはディスクサーバのみで構成するのではなく, 汎用サーバ3台をMON, MDSとすることで, 最大1台の障害を許容できるため, 利用者への影響を少なくできる。これらの技術を用いて構成したシステム構成図を図4.1に示す。

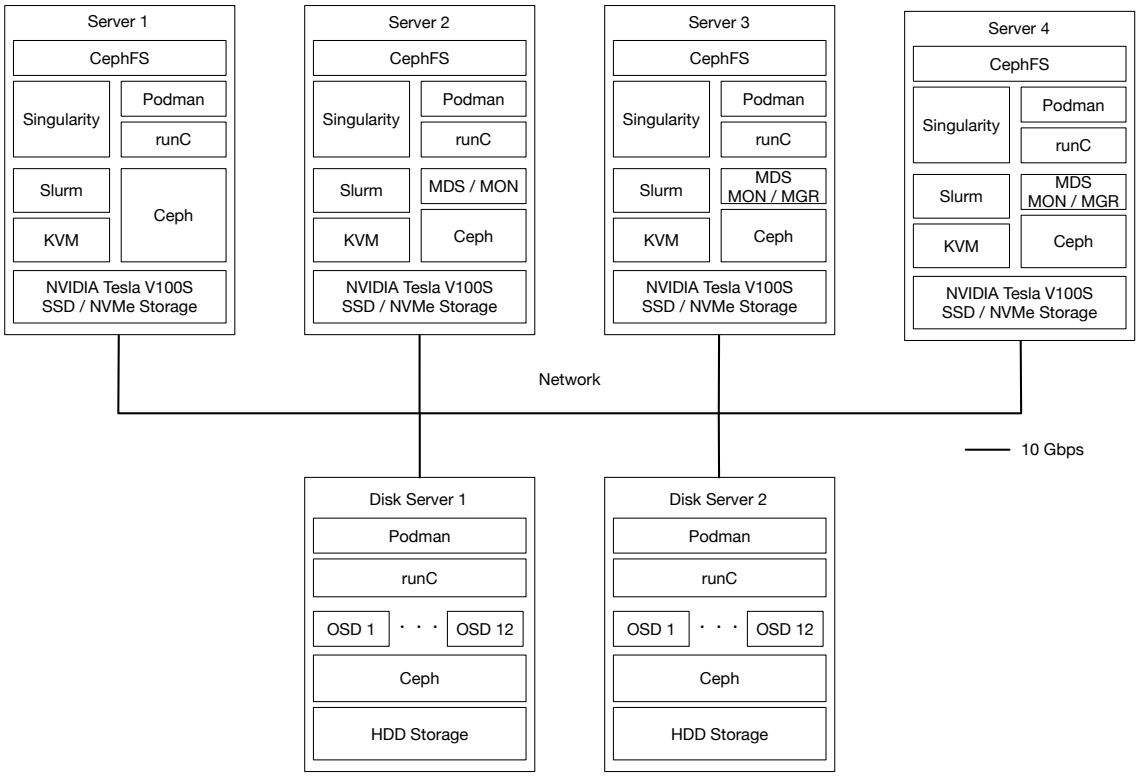
Server 1
CephFS
Singularity
Podman
runC
Slurm
Ceph
KVM
NVIDIA Tesla V100S
SSD / NVMe Storage
Server 2
CephFS
Singularity
Podman
runC
Slurm
MDS / MON
KVM
Ceph
NVIDIA Tesla V100S
SSD / NVMe Storage
Server 3
CephFS
Singularity
Podman
runC
Slurm
MDS
MON / MGR
KVM
Ceph
NVIDIA Tesla V100S
SSD / NVMe Storage
Server 4
CephFS
Singularity
Podman
runC
Slurm
MDS
MON / MGR
KVM
Ceph
NVIDIA Tesla V100S
SSD / NVMe Storage
Network
10 Gbps
Disk Server 1
Podman
runC
OSD 1
· · ·
OSD 12
Ceph
HDD Storage
Disk Server 2
Podman
runC
OSD 1
· · ·
OSD 12
Ceph
HDD Storage

図 4.1: システム構成図

第5章　教育計算機システムの管理

本章では, 構築した教育計算機システムの管理の方法, 利用方法について述べる。

5.1　FileSystem の管理

5.2　Podman

5.2.1　ie-podman の利用方法

5.3　Singularity と Slurm を利用した演習

第6章　まとめ

6.1　今後の課題

参考文献

[1] 金城篤史，城間政司，比嘉哲也，長田智和，玉城史郎，谷口祐治：“情報工学系学科における教育用計算機システムの自主構築に関する取り組み”，教育システム情報学会論文誌，Vol.26，No.1，pp.79-88，2009/1

[2] KVM，https://www.linux-kvm.org/，2021/1/8.

[3] Docker，https://www.docker.com/，2021/1/8.

[4] Docker Registry，https://docs.docker.com/registry/，2021/1/8.

[5] Podman，https://podman.io/，2021/1/4.

[6] Singularity，https://sylabs.io/singularity/，2021/1/8.

[7] Ceph，https://docs.ceph.com/en/latest/，2021/1/12.

[8] Ansible，https://www.ansible.com/，2021/1/12.

[9] Slurm, https://slurm.schedmd.com/overview.html, 2021/1/14.

[10] rsnapshot, https://rsnapshot.org/, 2021/1/15.

[11] 平良 太貴 and 河野 真治，OS 授業向けマルチユーザ VM 環境の構築，研究報告システムソフトウェアとオペレーティング・システム (OS)(2014).

[12] 城戸翔太，安里悠矢，城間政司，長田智和，谷口祐治，”情報系学科における教育情報システムの構築及び運用管理に関する取り組み”，研究報告インターネットと運用技術 (IOT)(2016).

謝辞

感謝します。

2021 年 2 月
宮平 賢