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, 以下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, これまで利用されてきた旧システムについて述べる
- 第 4 章は, 教育計算機システムの構築について述べる
- 第 5 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 6 章は, 教育計算機システムの評価について述べる
- 第 7 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。

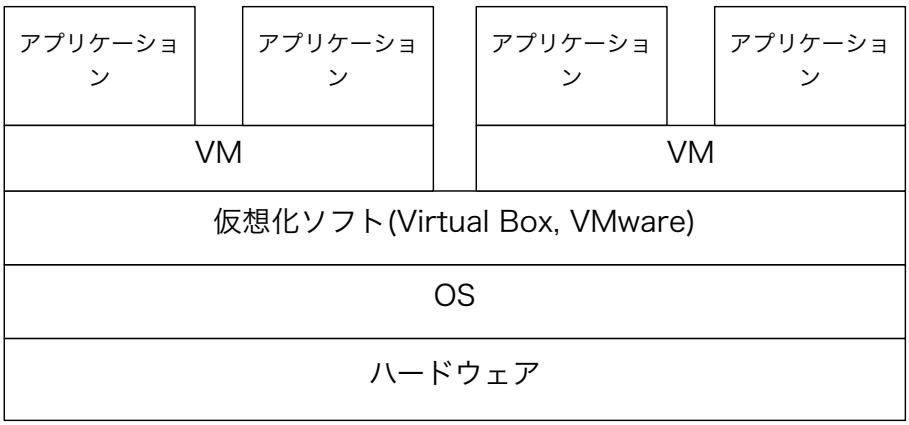

図 2.1: ホスト型

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。

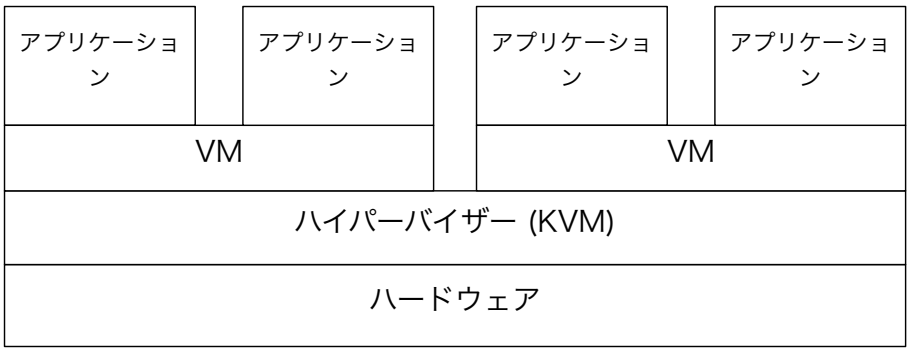

図 2.2: ハイパーバイザー型

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。

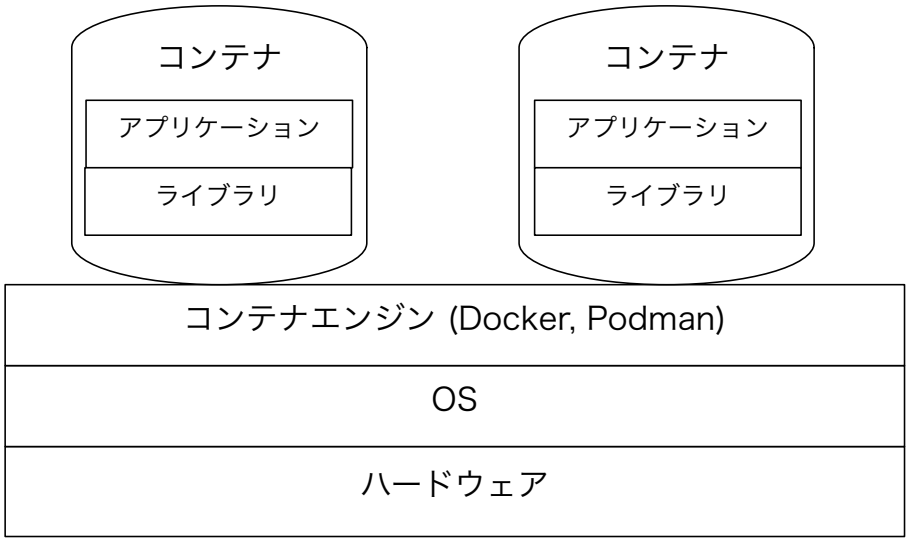

図 2.3: コンテナ型

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)。

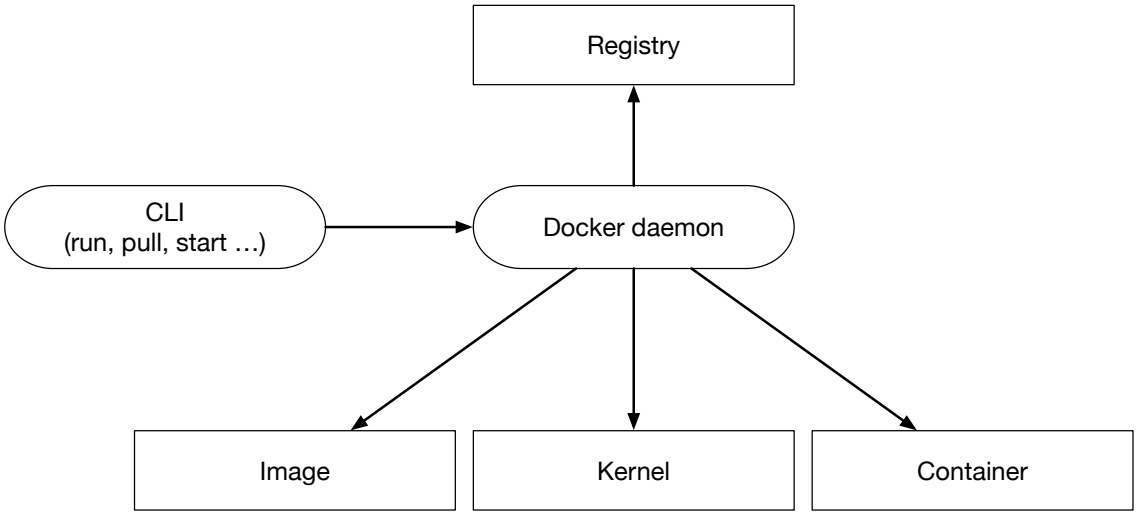

図 2.4: Docker

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。

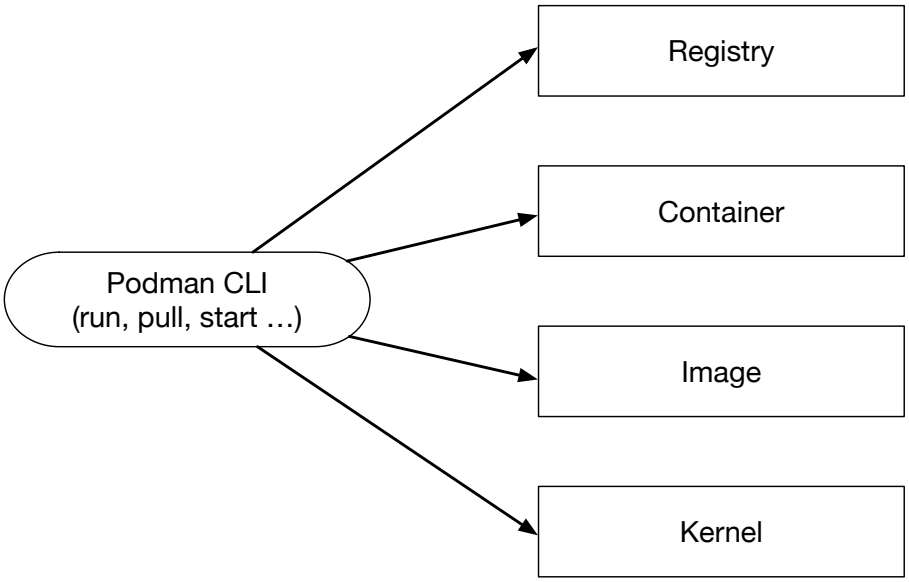

図 2.5: Podman

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。

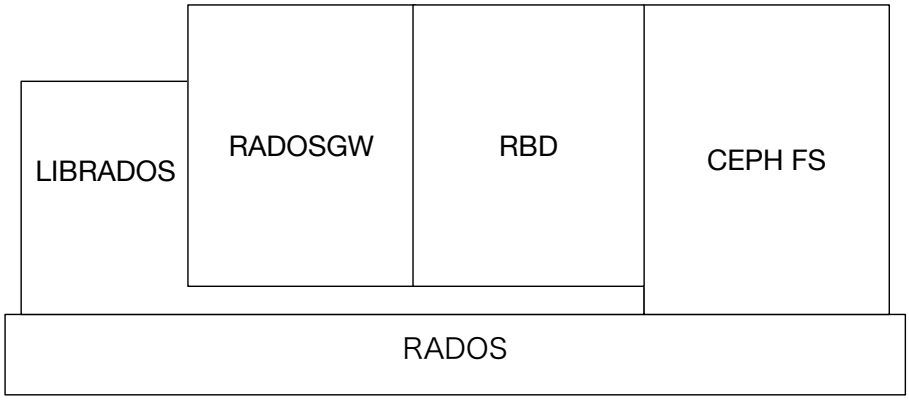

図 2.6: Ceph のアーキテクチャ

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。

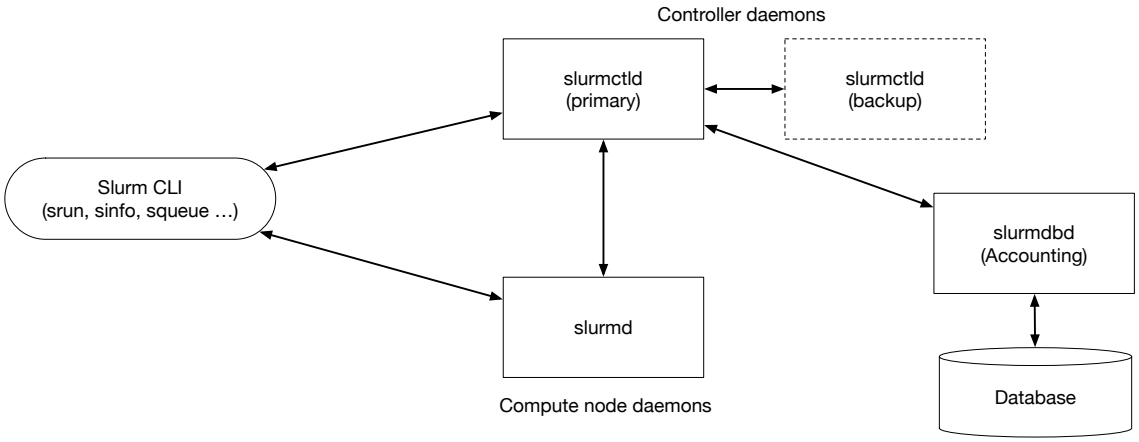

図 2.7: Slurm のアーキテクチャ

2.9 rsnapshot

rsnapshot[10] は rsycn に基づく増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。

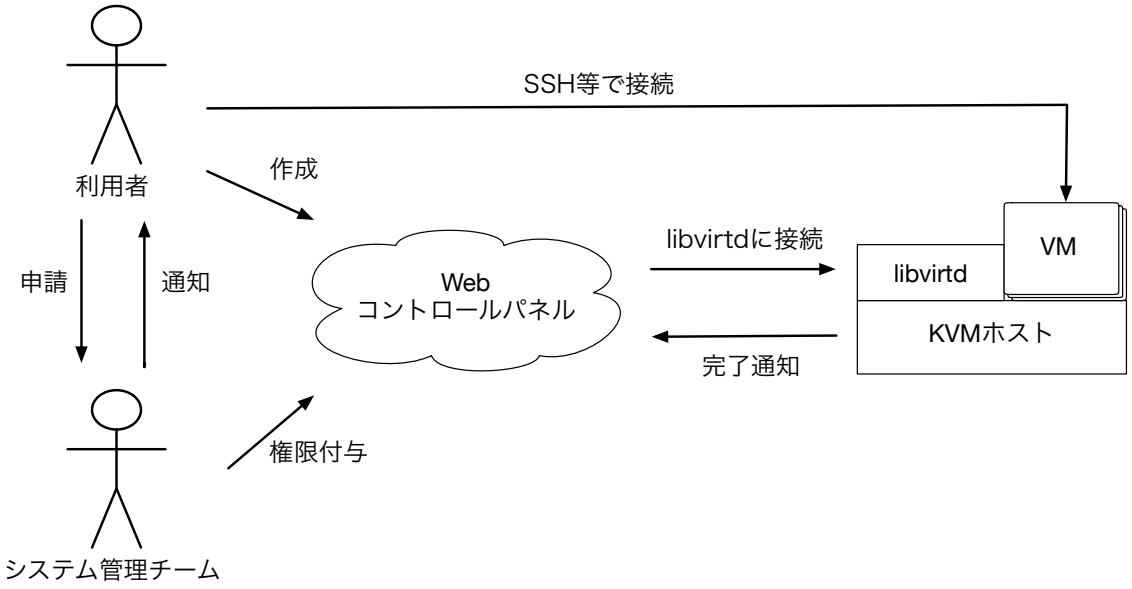

図 3.1: Akatsuki の概要

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　VM 貸出サービスの移行

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 コンテナ環境の導入

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　ジョブスケジューラの構築

旧システムではVMベースのため, 利用者が演習や研究等のプログラムは決められたリソースで実行する必要があった。新システムはコンテナベースに変更したことにより, 利用者は汎用サーバのリソースを利用できる。そのため, 複数の利用者が多くのリソースを要求するプログラムを実行した場合, リソース不足やリソースの競合が考えられる。そこで, 汎用サーバのリソースを効率よく利用できるようにするため, ジョブスケジューラであるSlurmにより管理を行う。Slurmは最悪待ち時間を減らすのではなく, 計算リソースの利用効率を上げることを重視する。そのため, Jobの優先順位は以下のように設定を行う。

- 要求するリソースの少ないJobの優先度を高くする
- 実行時間が短いJobの優先度を高くする
- これまでのJobの実行履歴で優先度は変化しない

また, Slurmに登録されるJobはバックフィルを採用する。バックフィルは図4.1のように, 後から投下されたJobが, 現在処理されているJobの実行時間以内であり, 空きリソースで処理可能ならば, 先に投下されたJobより先に処理される。

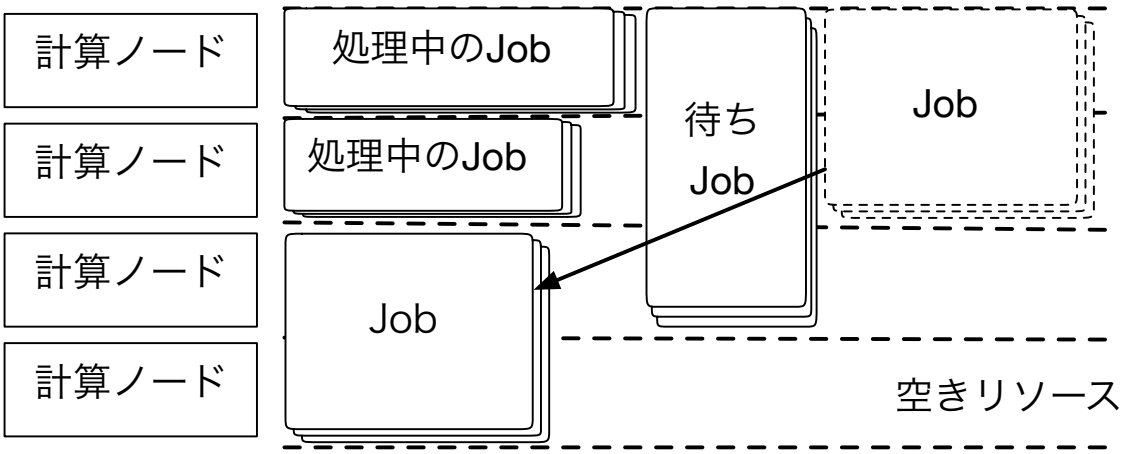

図 4.1: バックフィル

4.1.4　ファイルシステムの構築

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。

4.1.5 バックアップ戦略

旧システムには SAN 用ストレージの他に大容量ストレージが導入されており, バックアップ用として利用されていた。バックアップは 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フルバックアップ, 週に一度差分バックアップを行っていた。しかし, 新システムではストレージサーバ 2 台のため, 毎月フルバックアップではディスク容量を圧迫してしまう。そこで, 新システムでは, さくらインターネット株式会社 (以下, さくらインターネット) が提供する専用サーバへバックアップを行う。専用サーバは 4TB の SAS ディスクを 12 台搭載しており, 実行容量は 24TB を有している。その専用サーバへのバックアップは Web のデータ, ユーザのホームディレクトリを rsnapshot を用いて増分バックアップを行う。旧システムより容量は少ないが, 増分バックアップのため使用されるディスク領域は継続的に増えることがない。また, データの復元には rsync などでコピーを行うだけのため, クラウドサーバであっても特定のファイルのみを迅速に復旧できる。rsnapshot は以下のように設定を行い, 1 年分のデータを保存する。

- 毎日 0 時に増分バックアップを実行し, 最大 7 個のスナップショットを保存する
- 毎週月曜の 9 時に一週間分のスナップショットを取得し, 最大 4 個のスナップショットを保存する
- 毎月 1 日の 12 時に 1ヶ月分のスナップショットを取得し, 最大 12 個のスナップショットを保存する

4.1.6 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 台も含める。ディスクサーバは OSD を持ち, 汎用サーバが MON, MDS, MGR を担当する。汎用

サーバを含めることで, 最大 1 台の障害を許容できるようになる。そのため, 利用者への影響を少なくすることができる。これらの技術を用いて構成したシステム構成図を図 4.2 に示す。

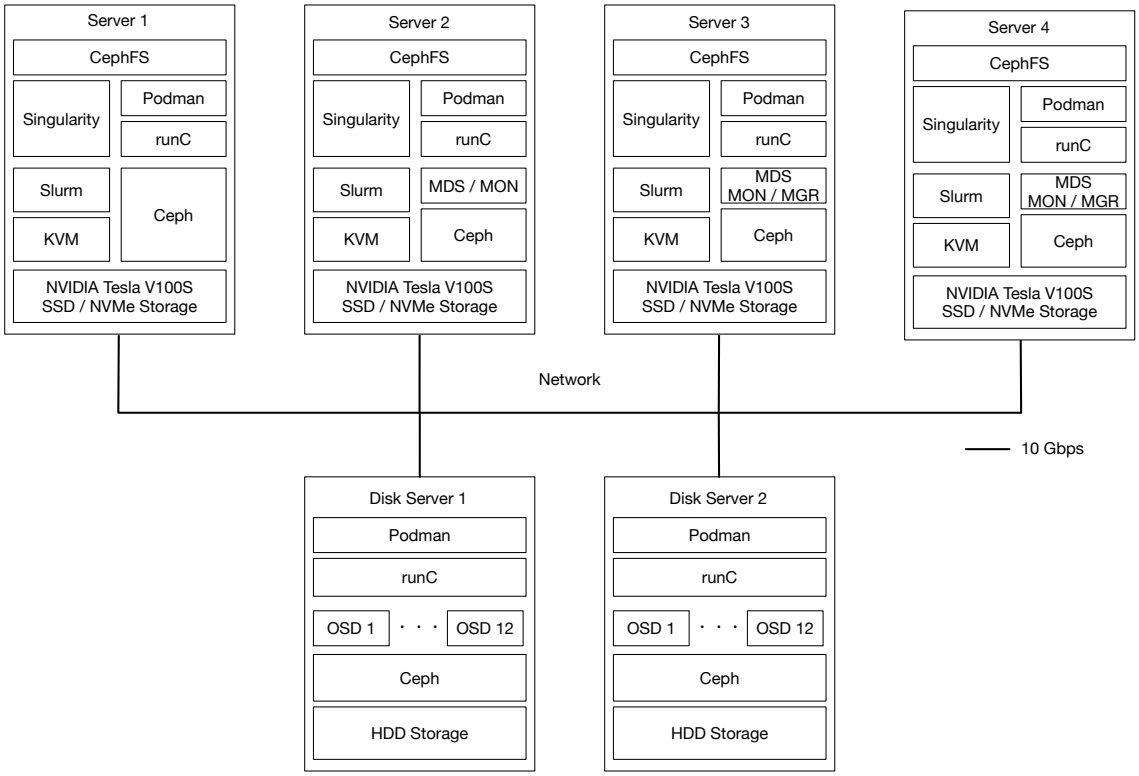

図 4.2: システム構成図

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方法, 利用方法について述べる。

5.1　LDAP による権限管理

教育計算機システムは, 本コースの学生や教師のユーザアカウントを LDAP で管理している。また, ユーザの権限も管理される。VM 貸出サービスやコンテナ環境は LDAP からユーザアカウントの情報を取得することで, 他の利用者のリソースに対して操作を制限する。

5.2　VM 貸出サービスの利用

ie-virsh は旧システムより利用されている。新システムでも利用を継続し, 新たに機能の追加を行った。

これまでの ie-virsh では VM を利用するには, 手元の PC で作成した VM イメージをサーバにアップロードを行い, ie-virsh でドメインの定義が必要であった。そこで, 新たに VM のテンプレートのイメージを用意し, テンプレートから差分で VM を作成できるようにした。VM のテンプレートから新たに VM を作成するには下記のように行う。また, –gdb オプションを付与することで, GDB による Linux Kernel の Debug を行える。

```
$ ie-virsh define --template Ubuntu-20 [VM_NAME]
```

VM の作成で利用できるテンプレートは下記の操作で確認することができる。

```
$ ie-virsh templates
```

また, 作成した VM のデフォルトのリソースは旧システムから変更はなく, CPU1 コア, メモリ 1GB, ディスク容量 10GB となっている。これまでは, リソースの変更は申請が必要であったが, 下記の操作で VM のリソースを変更が行える。しかし, ディスク容量の変更には申請が必要となっている。

```
$ ie-virsh edit [VM_NAME]
```

作成したVMにIPアドレスを割り当てるには, IPアドレス管理サービスにMacアドレスを登録する必要がある。VMのMacアドレスはie-virshが作成するXMLファイルに定義されており, dumpxmlでXMLを出力し確認できる。しかし, XMLを出力すると必要とする情報以外も多く表示されるため, 下記の操作でVMのインターフェース情報を出力できるようになっている。

```
$ ie-virsh domiflist [VM_NAME]
```

5.3 コンテナ環境の利用

新システムではコンテナエンジンであるPodmanとSingularityを導入した。また, Podmanや Singularityを使用する上での不便な点を補うために作成したie-podmanについて説明を行う。

5.3.1 ie-podman

ie-podmanはPodmanをwrappし複数ユーザで利用することができるコンテナ管理ツールである。Podmanはマルチユーザに対応しているため, ie-podmanを利用せずともコンテナの作成などを行える。だが, コンテナへのIPアドレスの割り当てには, root権限が必要となるためrootlessでは実行できない。そのため, Webなどを実行しアクセスするにはポートフォワードを設定し, SSHポートフォワードを行う必要がある。そこで, ie-podmanではPodmanのすべての機能をwrappするのではなく, rootlessでは実行できない機能を提供する。表5.1はie-podmanで利用できる機能である。

表 5.1: ie-podmanのコマンド

build	Containerfileの指示に従いイメージを作成する
cp	コンテナにファイルを送信する
exec	起動中のコンテナでプロセスを実行する
images	コンテナイメージの一覧を表示する
info	コンテナの情報を表示する
logs	コンテナのlogを表示する
ps	起動中のコンテナの一覧を表示する
registry	学科のレジストリの操作を行う
rm	コンテナを削除する
run	コンテナを作成する
sif	イメージをsifファイルに変換する
start	コンテナを起動する
stop	コンテナを停止する

新しいコンテナの作成は, Podman の run と同じように実行できる。しかし, ie-podman 独自のオプションを提供する。run 時に–gpu オプションを指定することでコンテナ内で GPU を使用できる。また, –ip オプションを指定することで, 使用されていない IP アドレスが割り振られる。コンテナ名は指定することもきるが, ユーザ名で補完されるため, 他ユーザと重複することはない。ie-podman を使用して, 新しいコンテナの作成は下記のように行う。

```
$ ie-podman run --ip --gpu --name [CONTAINER_NAME] [IMAGE]
```

新システムにインストールされている Podman は rootless でコンテナイメージの作成は低速である。これは, 開発段階ということ, 新システムのユーザのホームディレクトリは CephFS で提供されているためである。ie-podman は root の Podman を利用し SSD 上に作成されるため, Podman と比較し高速である。イメージ名はコンテナ名と同じくユーザ名で補完されることで, 他ユーザと重複することはない。ie-podman を使用して, 新しいイメージの作成は下記のように行う。

```
$ ie-podman build --tag [IMAGE_NAME] [CONTEXT]
```

また, 作成したコンテナイメージは下記の操作で一覧を表示することができる。一覧で表示されるイメージは ie-podman で作成したイメージのみである。

```
$ ie-podman images
```

Singularity は Docker で作成したイメージを sif ファイルに変換し, Singualrity で利用できる。sif ファイルへの変換は docker デーモンへリクエストを行う必要があるが, Podman はデーモンレスで動作する。そのため, Podman で作成したイメージを Singularity に変換するには一手間掛かる。Singularity は Definitionfile を作成し, ビルドすることで, sif ファイルを作成できる。だが, イメージのビルド中にエラーが発生すると, 一からビルドを行う必要がある。Docker や Podman はイメージのビルド時にレイヤーごとにキャッシュされるため, Containerfile に追加や編集を行っても前回のキャッシュが使用されることで, 高速にビルドが行われる。そこで, ie-podman で作成したイメージを sif ファイルへ変換する機能を作成した。ie-podman でイメージを作成し, 下記の操作を行うことで sif ファイルへ変換を行う。

```
$ ie-podman sif [IMAGE_NAME]
```

新システムではコンテナベースでシステムを構築するため, コンテナレジストリを導入した。レジストリの利用は学内ネットワークから可能であり, push や pull が可能である。ie-podman からだけでなく, Podman や手元の PC の Docker からも利用できる。レジストリへの登録には, 登録するイメージに tag を付け push する必要がある。ie-podman では本コースのレジストリを利用しやすくするため, 簡単に操作できる機能を作成した。ie-podman では下記の操作で本コースで利用するレジストリへ登録できる。

```
$ ie-podman registry push [IMAGE_NAME]
```

また, レジストリに登録されているイメージ一覧を表示することも可能である。下記の操作でイメージ一覧を表示を行う。イメージ名を指定することで, イメージの tag 一覧の表示も可能である。

```
$ ie-podman registry search
$ ie-podman registry search [IMAGE_NAME]
```

5.3.2 GPU を利用した演習

新システムでは, 汎用サーバに搭載される GPU をコンテナから利用できる。コンテナから GPU の利用は, Podman と Singularity の両方から可能である。だが, プログラムの実行には Slurm に Job として投下する必要がある。そのため, イメージを単一ファイルベースとして扱え, ユーザのホームディレクトリがコンテナにマウントされる Singularity を主に利用する。Singularity のコンテナの実行には, 下記の操作で行える。また, 実行時に–nv オプションを指定することで, コンテナから GPU を利用することが可能になる。

```
$ singularity run --nv [SIF_NAME]
```

実行には run, exec, shell のサブコマンドがあり, run では sif ファイルを作成する際に指定が可能な runscript が実行される。指定されない場合は shell が起動する。また, exec ではイメージ内にインストールされている任意のコマンドを実行することが可能である。これらのサブコマンドを利用し, Slurm に Job を投下する際の batch ファイルを作成する。batch ファイルはソース 5.1 の 2~8 行目ように, Job に必要なリソースを定義する。リソースの定義後に, 実行したい処理を記述する。

Listing 5.1: batch ファイル

```
1  #/bin/bash
2  #SBATCH --job-name sample
3  #SBATCH --output logs/%x-%j.log
4  #SBATCH --error logs/%x-%j.err
5  #SBATCH --nodes 1
6  #SBATCH --cpus-per-task 8
7  #SBATCH --gpus tesla:1
8  #SBATCH --time 01:00
9
10 singularity exec --nv [SIF_NAME] [COMMANDS]
```

batch ファイルを作成後, 下記の操作で Job を投下することが可能である。

```
$ sbatch [BATCH_FILENAME]
```

また, Job の各種情報は, 下記の操作で表示することが可能である。

```
$ squeue
```

投下した Job を停止するには, 下記の操作で行うことができる。Slurm はユーザごとに Job が管理されるため, 他ユーザの Job を停止するこはできない。

```
$ scansel [JOB_ID]
```

第6章　教育計算機システムの評価

本章では, 教育計算機システムの評価を行う。新たに採用した Ceph との比較, VM 貸出サービス, コンテナ環境の評価を述べる。

6.1　ファイルシステムの評価

旧システムの VM 保存場所として利用していた GFS2, ユーザのホームディレクトリとして利用していた NFS との速度比較を行う。

6.1.1　実験概要

実験に使用する汎用サーバ環境は, 表 4.1 を使用する。また, Ceph の構成は図 4.2 となっている。

ベンチマークには dd というファイル変換やコピーを目的とした GNU/Linux のコアユーティリティを用いる。dd は, 低レベルの I/O フロー制御機能を備えており, シーケンシャル書き込み, または読み取りの速度を測定できる。データの変換方法に fdatasync を指定することで, 書き込み終了の直前に sync を 1 回要求するため, 実際の動作に近い動作で測定が可能である。

書き込みには下記のコマンドを用いる。

```
$ dd if=/dev/zero of=benchmark bs=64K count=2K conv=fdatasync
```

また, ファイルサイズは 128MB, 256MB, 512MB, 1GB, 2GB, 4GB の書き込みを行う。5 回測定を行い平均を比較する。

6.1.2　ファイルシステムの速度比較

図 6.1 は CephFS, Ceph RBD, GFS2, NFS におけるファイルサイズに対する書き込み速度である。

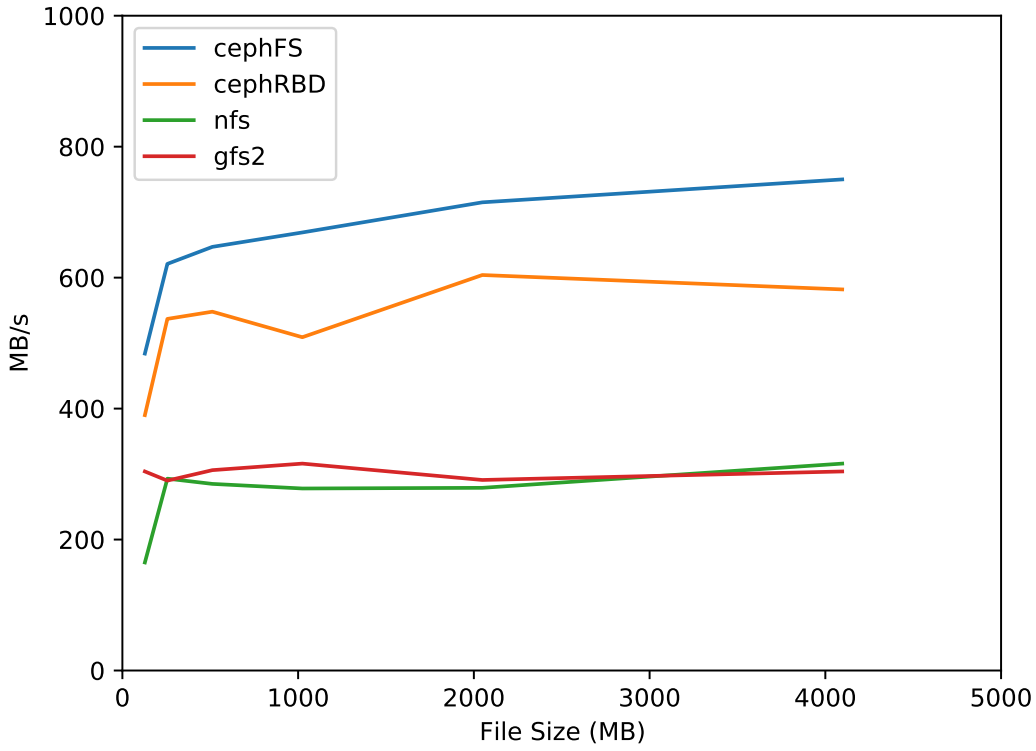

図 6.1: 書込み速度の比較

6.1.3 考察

旧システムのホームディレクトリは, iSCSI 経由でマウントされたデバイスを NFS から提供していた。iSCSI の通信には 10Gbps の回線で接続されているが, NFS の提供は VM で行われており, 1Gbps で提供されていた。そのため, 10Gbps の回線で接続し, マウントしている Ceph では書き込み速度の改善が見られる。しかし, GFS2 は 10Gbps で接続されたクラスタで構成されているが, Ceph より低速である。旧システムでは, パッケージ等のアップデートがされておらず, Kernel の更新もされていなかった。Kernel は I/O に関する多くの機能を提供するため, GFS2 の書き込みより, Ceph が高速になったのではないかと考えられる。

今回の計測では, 読み込み速度の測定を行えなかった。これは, 旧システムで読み込み時にバッファキャッシュを削除せずに測定を行ったためである。そのため, 純粋な読み込み速度を測定することができなかったことは反省点である。

6.2 VM 貸出サービスの評価

本コースの VM 貸出サービスでは, VM の作成やスペックの変更にはシステム管理チームへの申請が必要であった。これは, VM がリソースを過剰に使用できないようシステム管理チームで管理するためである。VM の作成やスペックの変更申請が届くと, システム管理チームと利用者の間でやりとりを行い対応していた。しかし, システム管理チームの状況によっては迅速に対応できず利用者を待たせることもあった。新システムでは, VM の作成やスペックの変更は ie-virsh により対応できるようになり, システム管理チームや利用者の手間を減らすことが可能になった。

また, これまでの VM 貸出サービスはテンプレートの VM イメージを Clone していたため, 新しく VM を作成すると 10GB の容量を使用していた。そこで, ie-virsh では差分で VM を作成機能が追加され, 新しく作成される VM は数十 MB になることで, 使用するディスク容量を抑えることが可能になった。

6.3 コンテナ環境の評価

新しく導入した Podman は, Docker と同じ CLI を提供するため, Docker を利用したことがある学生は抵抗なく利用できる。また, ie-podman により特権が必要な機能も, 利用者に特別な権限を与えることなく利用できるようになる。Singularity の導入により, GPU を手軽に利用できるため, 演習や研究等での利用も広がると考えられる。

VM では新しく VM を作成するたびに環境構築が必要であったが, コンテナイメージを作成することで環境構築は一度のみになる。また, 学科のレジストリーにイメージを登録することで, 他の利用者に共有も可能となる。Singularity では単一ファイルの sif のコピーを配布することで, 同じ環境でプログラムの実行することができる。

これまでの VM ベースのシステムから, コンテナベースのシステムへの移行により, 汎用バーバのリソースを効率よく利用できる。また, 学生も気軽に利用できるようになったのではないかと考えられる。

第7章　まとめ

本章では, 本論文で述べたことのまとめ, 今後の課題について述べる。

7.1　総括

本論文では, 2020 年 9 月に行われたシステム更新を中心に, 本コースの学生や教師などが利用できる教育計算機システムの構築について述べた。

以下, 本論文について振り返る。

第 1 章では, 本研究の背景や目的, また本コースのシステムの運用管理を担当するシステム管理チームについて述べた。

第 2 章では, 本研究で利用した技術概要について述べた。

第 3 章では, 2015 年から稼働していた旧システムのオンプレミス環境, 演習や研究等で利用できる VM 貸出サービスについて述べ, これらの問題点を示した。まず, VM 貸出サービスには Akatsuki と ie-virsh があり, Akatsuki はテンプレートから新しく VM を作成し利用できるサービスである。ie-virsh は手元の PC で作成した VM イメージを学科サーバへデプロイを行うサービスである。Akatsuki は VM 貸出だけでなく, 有線 LAN 接続サービスの機能を持っているが, VM 貸出サービスはあまり周知されていなかった。また, VM 貸出サービスは本コースの学生や教師が利用可能であるが, VM の作成やスペックの変更には申請が必要で余り利用されていなかった。そのため, 旧システムの汎用サーバのリソースは余っていた。

第 4 章では, 新システムで構築したオンプレミス環境, 仮想環境, 新たに導入したコンテナ環境やジョブスケジューラについて述べた。また, これらのシステムでの構成についても述べた。新システムでは新たに GPU が搭載され, 学生が利用できる環境が求められた。そこで, VM ベースでシステムを構築するのではなく, コンテナベースで構築した。また, 学生もコンテナを利用できるよう, マルチユーザに対応している Singularity と Podman を導入した。マルチユーザで利用できるため, リソースの競合を防ぐためにジョブスケジューラである Slurm を導入した。一方, ファイルシステムには Ceph を採用した。Ceph は RBD やブロックデバイス, POSIX 互換のファイルシステムなど, 複数のアクセス方法が利用可能であるため, 用途により柔軟に対応することができる。

第 5 章では, 第 4 章で構築したシステムの管理や利用方法について述べた。システムのユーザの管理には旧システムと同様に LDAP を採用した。また, 旧システムから移行し改良を加えた VM 管理サービスの利用, 管理方法について述べた。次に, 新しく導入した Podman の利用方法, Singularity と Slurm を用い GPU を使用する方法について述べた。

第 6 章では, 第 4 章で構築したシステムの評価について述べた。

7.2 今後の課題

7.2.1 教育計算機システムの周知

7.2.2 ie-podman のネットワーク構成

rootless の Podman ではコンテナに個別に IP アドレスを割り当てられないため, Podman を wrapper した ie-podman を作成した。ie-podman を利用することで, コンテナに個別に IP アドレスを割り当てられる。しかし, 現在のネットワーク構成はプレフィックス長が 24 のため, 最大 254 個の IP アドレスしか割り当てできない。コンテナを削除することで IP アドレスは返却されるが, コンテナを削除せず停止のままでは, 割り当て可能な IP アドレスが枯渇する。そのため, ie-podman が利用するネットワーク構成の変更を行うか, コンテナが停止のまま数日経つ場合に ie-podman から自動削除する必要がある。

7.2.3 ジョブスケジューラの周知

ジョブスケジューラの構成は汎用サーバ 1 台がログインノードと計算ノードを兼用し, 残りの 3 台が計算ノードとなっている。コンテナからプログラムの実行は Slurm に Job を投下しなくても, ログインノードで実行できる。また, GPU リソースを要求していなくても, Singularity や Podman から GPU を使用るすことが可能である。Slurm による Job やリソースの管理を行うことで, 利用者同士のリソースの競合を防ぐことができる。そのため, Slurm に Job の投下方法や必要なリソースの要求などを定期的な周知する必要がある。

7.2.4 バックアップの運用

新システムのバックアップには Ceph とさくらインターネットが提供する専用サーバを利用している。Ceph はブロックデバイスとしアクセスを行い, XFS でフォーマットを行い利用する。だが, 新システム構築の試験時に Ceph の MON の IP アドレスを変更後, Ceph に保存したデータが全て取れない問題が発生した。これは Ceph の MON が IP アドレスの変更を想定していないためであった。これから Ceph を運用するにあたり障害が起きないとは限らない。Ceph は自己修復機能を搭載しているが, 万が一修復できない場合, ユーザのホームディレクトリや, バックアップデータが消える恐れがある。その時に備え専用サーバにも保存しているが, 専用サーバではさくらインターネットからデータ取得するため, ホームディレクトリ等を復元するには時間が掛かる。そのため, Ceph と専用サーバ意外にもバックアップ先を用意する必要がある。

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