コンテナ型仮想化とハードウェア仮想化の性能調査

平成 30 年度

学士学位論文

高知工科大学
情報学群
コンテナ型仮想化と
ハードウェア仮想化の性能調査

Investigation of performance of container-based virtualization and hardware virtualization

1190318　　木原裕貴

指導教員　　情報学群 敷田幹文

2019 年 2 月 28 日

高知工科大学 情報学群

要 旨

高知工科大学

情報学群

コンテナ型仮想化と ハードウェア仮想化の性能調査

木原裕貴

仮想化技術を用いたクラウドコンピューティング環境の普及に伴い, 仮想化環境は重要なプラットフォームの一つとなっている. 仮想化技術の代表的なものにハードウェア仮想化とコンテナ型仮想化がある. これらの仮想化技術は物理的なリソースをソフトウェアによって仮想的に提供する技術であり, ベアメタル環境と同様のパフォーマンスを発揮できるとは限らない. そのため, 仮想化技術を用いたサーバ構築や運用のためには仮想化技術の性能を把握し, 利用目的に適した仮想化環境を導入することが重要である.

本研究では研究室内サーバで構築されたハイパーバイザーを用いたハードウェア仮想化環境と新たに構築したコンテナ型仮想化環境を用いて, I/O パフォーマンスと HTTP リクエスト処理性能を調査する. その結果より, それぞれの仮想化環境がどのような利用目的に適しているかを考察する.

キーワード　コンテナ型仮想化, ハードウェア仮想化, 性能評価, ディスクベンチマーク

Abstract

Investigation of performance of container-based virtualization and hardware virtualization

Hiroki KIHARA

With the spread of cloud computing environments using virtualization technology, the virtual environment is one of the important platforms. Typical of virtualization technology is hardware virtualization and container-based virtualization. These virtualization technologies are technologies that virtually provide physical resources by software and can not always achieve the same performance as the bare metal environment. For this reason, it is important to grasp the performance of virtualization technology and introduce a virtualization environment suitable for the purpose of use for server construction and operation using virtualization technology.

In this research, I/O performance and HTTP request processing performance are investigated using a hardware virtualization environment using a hypervisor and a newly constructed container-based virtualization environment built on a laboratory server . From the result , I consider what kind of usage purpose each virtual environment is suitable.

key words container-based virtualization, hardware virtualization, performance evaluation, disk benchmark

目次

目次

図目次

表目次

第 1 章

はじめに

1.1 研究背景

近年, 様々なサービスが仮想化技術を用いたクラウドコンピューティング環境に移行しており, 仮想化環境は重要なプラットフォームの一つとなっている. 代表的な仮想化の手段としてハードウェア仮想化とコンテナ型仮想化が挙げられる. ハードウェア仮想化とは一台の物理マシン上で複数の仮想ホストを動作させる技術である. ハードウェア仮想化は大きくハイパーバイザー型仮想化とホスト型仮想化に分けられる. ハードウェアの入出力の際, 汎用的なホスト OS を経由するホスト型仮想化に比べ, 仮想化ソフトウェアのハイパーバイザーを経由するハイパーバイザー型仮想化の方が, オーバヘッドが少ないという特徴があり, 実システムではハイパーバイザー型仮想化が主流となっている. また, コンテナ型仮想化とはコンテナと呼ばれるプロセスグループにそれぞれアプリケーション実行環境を与える技術であり, 近年利用者が急激に増えている技術である. これらの仮想化技術は仮想化によるオーバーヘッドが少なからず発生する. よって, ベアメタル環境と同じ性能を発揮できるとは限らない. しかし, コンテナ型仮想化は急速に流行している技術であるために, 利用者の理解がまだ十分に追いついておらず, コンテナ型仮想化の特性を活かした利用ができているとは言えない. 現在, インターネットの普及に伴い様々なサービスが展開され, 国際的なデジタルデータ量は年々増加し, 2020 年には約 40 ゼタバイトに達すると予想されている [1]. このような状況の中, 仮想化技術を用いたサーバ構築や運用のためには, 仮想化技術の I/O パフォーマンス等の特性を把握し, 利用目的に適した仮想化環境を導入することが重要である.

1.2　本研究の目的

本研究では, ハードウェア仮想化環境とコンテナ型仮想化環境, そして比較対象として仮想化なしのベアメタル環境を用いて実システムを想定した I/O 性能と Web サーバ性能を測定する. そして, 測定結果からハードウェア仮想化環境, コンテナ型仮想化環境のオーバーヘッドの差を確認し, それぞれに適している実システムでの利用目的を明らかにする.

第 2 章

既存研究

ベアメタル環境とハードウェア仮想化環境の I/O 性能評価に関する研究 [2] がある．この研究では NVMe SSD を適用したベアメタル環境とハイパーバイザー型仮想化環境 KVM の I/O 性能を比較している．I/O の測定には I/O ベンチマークツール fio を使い, ブロックサイズ 4 KB, SSD へのアクセス方式にランダム I/O で直接 SSD に書き込む時の IOPS を測定している．その結果, 仮想化環境ではベアメタル環境と比べて読み込みの性能が 0.79 倍, 書き込みの性能がほぼ同等という結果が得られたことが示されている．

ベアメタル環境とコンテナ型仮想化環境の性能評価に関する研究 [3] がある．その中の I/O の性能測定で, I/O ベンチマークツール FFSB を使い, 1 MB のファイルを 4096 個作成し, その後 1 MB の読み込みを 5 分間行うことでベアメタル環境とコンテナ型仮想化環境の I/O 性能を測定している．その結果, ベアメタル環境に比べてコンテナ型仮想化環境での I/O 性能が大きく下がる結果となったことが示されている．

これらの仮想化環境の性能測定の研究では I/O 性能でベアメタル環境に対し, 仮想化環境ではオーバーヘッドにより性能が低下, もしくはほぼ同等程度の性能となったことがわかる．

また, Felter ら [4][5] は, ベアメタル環境, ハードウェア仮想化環境, コンテナ型仮想化環境の性能調査を行っている．この中でベアメタル環境, ハードウェア仮想化環境を提供する KVM, コンテナ型仮想化環境を提供する Docker の I/O 性能比較を行っている．この測定では各物理ホストに IBM FlashSystem 上の SSD へファイバチャネルで接続した構成で測定している．測定には I/O ベンチマークツール fio を使用して, ブロックサイズ 1 MB, ファイルサイズ 16 GB でアクセス方式をシーケンシャル I/O とし, スループットを測定している．測定の結果, シーケンシャル write の結果において, ベアメタル環境と Docker ではほとんど

差が見られなかったが, KVM では他の環境と比べわずかにオーバーヘッドが見られた.

この研究の測定では, Docker のデータ永続化に Docker ストレージドライバを回避した方法を用いている. この方法ではコンテナ型仮想化のオーバーヘッドを回避できるが, コンテナ間のボリューム共有ができないため, 実システムを想定しているとは言えない.

第 3 章

関連技術

3.1 VMware ESXi

ESXi はハイパーバイザー型仮想化を提供するエンタープライズ向けのハイパーバイザーである. ハイパーバイザー型仮想化は 1 台の物理マシン上に仮想ホストを複数台稼働させることを可能にする技術であり, OS からハードウェアまでを仮想的に構築する. ESXi は各仮想ホストで異なる OS をインストールでき, 仮想化なしの物理マシンと同様の操作で利用することができる. また管理ソフトウェア Vcenter Server によって複数の仮想ホスト, 複数の ESXi ホストに依存する全てのコンポーネントを一元管理できる. さらに Vcenter Server はある ESXi ホスト上で動作中の仮想マシンを停止させることなく, 別の ESXi ホストに動的に移動させる vSphere vMotion 機能によって高いメンテナンス性を実現している. ただし, 仮想ホストは OS からハードウェアまでを仮想化するために起動に時間が掛かるという問題がある. そのため実際のシステムでは仮想ホストを常時起動させ, 負荷が増大した際にサービスが停止しないように, 十分なリソースを配分する運用方法をとっている.

ESXi の有償版に ESX があるが, 基本機能は全て同じであるため, ESXi で実際のシステムを想定した性能測定を行うことが可能である.

3.2 Docker と Kubernetes

Docker[6] は現在最もメジャーなエンタープライズ向けのコンテナ仮想化ソフトウェアである. コンテナ型仮想化ではコンテナと呼ばれる独立したアプリケーション実行環境をプロ

セス単位で構築する．コンテナは OS カーネルの機能によってリソースを各コンテナごとに隔離しており, 各コンテナはそれぞれの OS を持たない為, ハードウェア仮想化と比べて起動が高速である．この特徴を活かし, クライアントからのリクエストを契機にコンテナを起動し, さらに負荷が増大したコンテナがあれば, 迅速なスケールアウトによって対応するアーキテクチャの開発が進められている [7][8][9]．ただし負荷に応じたスケールアウトには多数のコンテナを管理するシステムが必要がある．この管理を行うのが Kubernetes[10] である．Kubernetes はクラスタ上のコンテナを管理するためのオーケストレーションシステムである．Kubernetes ではコンテナを Pod という単位で管理する．基本的には 1 Pod に 1 コンテナであり, 本研究でも 1 Pod 1 コンテナで測定を行なっている．Kubernetes では CPU 負荷に応じたオートスケールや何らかの原因で停止した Pod を再配置することでサービスを限りなく無停止で運用することが可能である．

第 4 章

性能測定と結果

4.1 測定環境

測定に使用した環境は全て研究室のオンプレミス環境上で構築したものである. 研究室内測定環境の全体の構成が図 4.1 である. ハードウェア仮想化環境には研究室内サーバ上に構築されている VMware ESXi によるハイパーバイザー型仮想化環境を用いて, その上に構築した仮想ホスト上で性能を測定した. ESXi 上に構築した仮想ホストの環境を表 4.1 に示す. コンテナ型仮想化環境には 4 台の物理サーバを使い, コンテナ型仮想化ソフトウェアに Docker を用いた Kubernets クラスタを新たに構築し, Pod 上で性能を測定した. 4 台の物理サーバは全て同一スペックのマシンであり, そのうち 1 台を Master ノード, 他 3 台を Worker ノードとした. Kubernetes クラスタに用いている物理サーバ 1 台の環境を表 4.2 に示す.

またベアメタル環境には Kubernets クラスタに用いている物理サーバのうちの 1 台を使用し, ホスト OS から性能測定を行った.

ESXi 上の仮想ホストと Kubernetes クラスタのノードで, OS とメモリサイズは統一しているが, CPU は異なるものを用いている.

4.1 測定環境

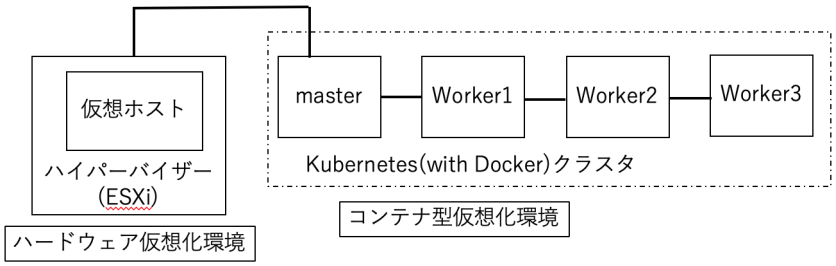

図 4.1 測定環境

表 4.1 ハードウェア仮想化環境

ハイパーバイザー	ESXi 6.5.0
ゲスト OS	CentOS 7.6.1810
CPU	Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz
メモリ	4GB

表 4.2 コンテナ型仮想化環境

ホスト OS	CentOS 7.6.1810
CPU	Intel(R) Xeon(R) CPU E3-1220 v6 @3.00GHz
メモリ	4GB
オーケストレーションシステム	Kubernetes v1.12.2
コンテナ型仮想化ソフトウェア	Docker v18.09.0
CNI	Flannel v0.10.0
分散 KVS	etcd v3.2.24

4.2　予備測定

性能測定では CPU の性能が測定結果に大きく影響する可能性が考えられる．そのため, 予備測定としてハードウェア仮想化環境とコンテナ型仮想化環境それぞれで使用している CPU の性能比較を行う．CPU 性能をシステムベンチマークツール UNIXBench で測定し, 測定結果の総合評価値で比較を行った．ハードウェア仮想化環境で使用している CPU の総合評価値は 2402.8 , コンテナ型仮想化環境で使用している CPU の総合評価値は 3280.1 であった．よって, ハードウェア仮想化環境で使用している CPU の性能よりコンテナ型仮想化環境で使用している CPU の性能が約 1.36 倍高いという結果が得られた．

4.3　物理ネットワークを介した I/O 性能測定の方法

実システムでは多くの場合, サービスを提供するサーバとサービスに用いるデータを格納するサーバは別の物理ホストとし, それぞれのサーバを物理ネットワークを介してデータをやりとりする構成をとる．そのため実システムと同様に物理ネットワークを介した I/O 性能の測定を行った．この測定ではベアメタル環境, コンテナ型仮想化環境, ハードウェア仮想化環境の各ホストから物理ネットワークを介して 1 台の NFS サーバをマウントする．ベアメタル環境からはホスト OS から NFS サーバをマウントした．コンテナ型仮想化環境からは Kubernetes の永続ストレージ手法の一つである PersistentVolumes 機能によって NFS サーバを Pod からマウントした．またハードウェア仮想化環境からは 2 つの方法を用いて NFS サーバをマウントした．1 つ目は仮想ホスト自身からマウントする方法である．2 つ目は仮想ホストがハイパーバイザーからマウントした NFS サーバを自身のストレージとする方法である．

この 4 つのホストからそれぞれが NFS サーバをマウントした環境で, dd コマンドで I/O 性能を測定した．dd コマンドを用いた I/O 性能測定のスクリプトは付録を参照することで確認できる．各ホストから, アクセス方式をシーケンシャル I/O, ブロックサイズを 64 MB, ファイルサイズを 64 MB, 128 MB, 256 MB, 512 MB, 1024 MB, 2048 MB, 8192 MB の

7 パターンに変化させ測定した．この時各ホストのキャッシュは使用せずに測定するが, NFS サーバのキャッシュは使用する．この測定を 5 回行い, スループットの平均値を算出した．この測定では各ホストの I/O 処理と物理ネットワークを介するファイル転送と NFS サーバの I/O のスループットを測ることとなる．ただし, NFS サーバのメモリサイズは 4 GB であるため, ファイルサイズが 64 MB から 2048 MB の小さなファイルサイズまでは NFS サーバのキャッシュへの I/O を測定することとなり, 大きなファイルサイズの 8192 MB では NFS サーバのディスクへの I/O を測定することとなる．

IPA によるインターネット上のサービス利用状況調査によると, 2018 年のインターネット上のサービス利用状について最も高いのは,「動画共有サイト」であった [11]．一般的に動画はファイルサイズが大きくなるため, 実際のシステムを想定する本研究ではサーバのキャッシュサイズを超えるような大きなファイルサイズの I/O が行われる可能性を考慮し, ファイルサイズ 8192 MB の I/O を測定した．

4.4　物理ネットワークを介した I/O 性能測定の結果

物理ネットワークを介した I/O 性能測定の結果を図 4.2 図 4.3 に示す．Write の結果ではどのホストでもファイルサイズによる大きな変化はなく, ベアメタル環境が最も高いスループットを発揮できることがわかった．ベアメタル環境と比べると仮想ホスト自身から NFS サーバをマウントした場合と Pod からマウントした場合の Write は 1 割程度スループットが低くなり, ハイパーバイザーからマウントした場合の Write では 3 割程度スループットが低くなることがわかった．

Read の結果ではファイルサイズが 64 MB から 2048 MB までの NFS サーバのキャッシュへの Read では変化がなく, 8192 MB の時は NFS サーバのディスクへの I/O のためスループットが低下しているが, どのファイルサイズにおいても, ベアメタル環境とハードウェア仮想化環境, コンテナ型仮想化環境でほぼ同様のスループットとなった．

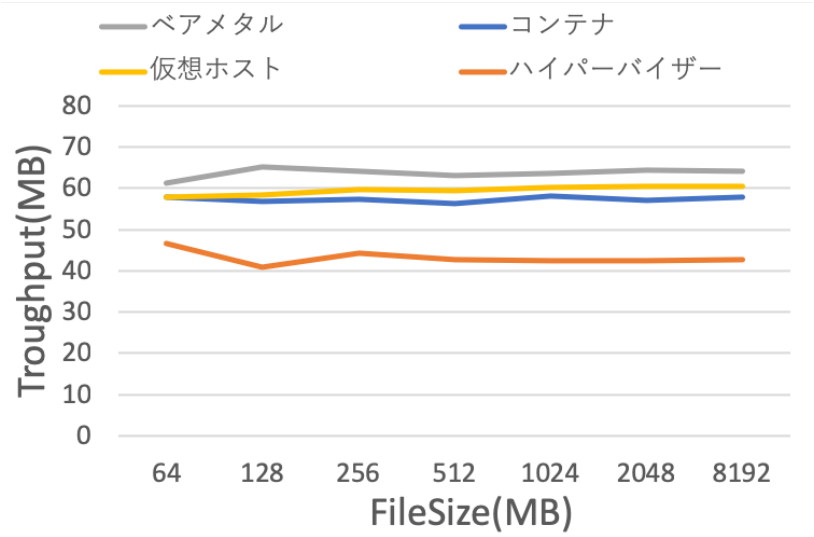

図 4.2　物理ネットワークを介した書き込みのスループット

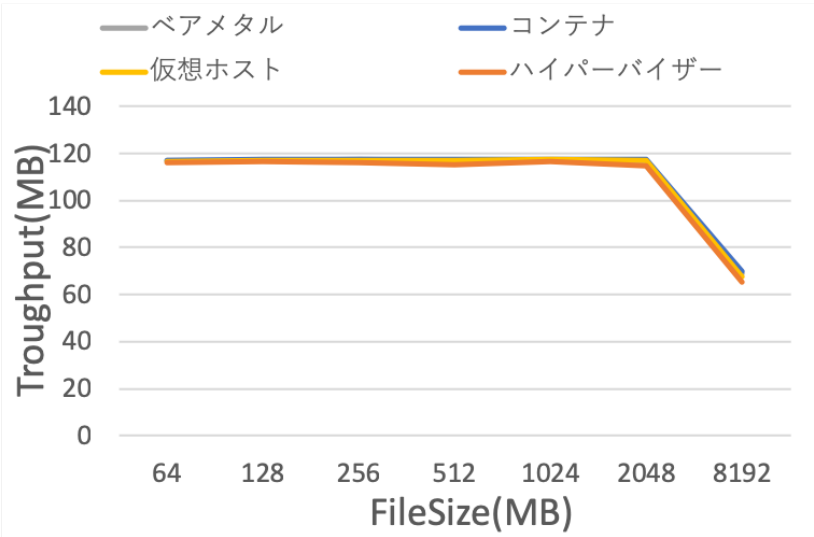

図 4.3　物理ネットワークを介した読み込みのスループット

4.5　仮想環境内ネットワークのみを介した Read 性能測定の方法

第 3.4 節より物理ネットワークを介した NFS サーバへの Read の結果ではベアメタル環境と比べてハードウェア仮想化環境, コンテナ型仮想化環境で, 仮想化オーバーヘッドによるスループットの差がほぼ無かった. この結果となった理由として, Read では仮想化のオー

バーヘッドがほぼ存在しない場合と, Read での仮想化オーバーヘッドは存在するが Read 以外の処理によって仮想化オーバーヘッドがスループットに影響していない場合が考えられる．その理由を明らかにするために仮想環境内ネットワークのみを介した NFS サーバへの Read 性能の測定を行い, 仮想化オーバーヘッドの有無を確認する．この測定ではベアメタル環境, コンテナ型仮想化環境, ハードウェア仮想化環境の各ホストから仮想環境内ネットワークのみを介して各 NFS サーバをマウントする．ベアメタル環境ではホスト OS 上に構築した NFS サーバを自身でマウントした．コンテナ型仮想化環境からは Pod が稼働している物理マシンのホスト OS 上に構築された NFS サーバを PersistentVolumes 機能によって Pod からマウントした．ハードウェア仮想化環境からはコンテナ型仮想化環境のようにホスト OS 上に NFS サーバを構築することができないため, NFS サーバを構築した仮想ホストを別の仮想ホストからマウントすることで, 仮想環境内ネットワークのみを通る構成とした．この 3 つのホストからそれぞれが NFS サーバをマウントした環境で, I/O ベンチマークツール fio を用いて I/O 性能を測定した．各ホストから, アクセス方式をシーケンシャル Read, ブロックサイズを 64 MB, ファイルサイズを 2048 MB として測定した．この時各ホストのキャッシュは使用せずに測定するが, NFS サーバのキャッシュは使用する．この測定を 5 回行い, 最大レイテンシの平均値を算出した．この測定では各ホストの Read 処理と NFS サーバのキャッシュからの Read のレイテンシを測ることとなる．

4.6　仮想環境内ネットワークのみを介した Read 性能測定の結果

仮想環境内ネットワークのみを介した Read 性能測定の結果を表 4.3 に示す．最大レイテンシがベアメタル環境と比べ, コンテナ型仮想化環境では 1.2 倍, ハードウェア仮想化環境では約 2.5 倍となった．この結果より, コンテナ型仮想化環境とハードウェア仮想化環境でそれぞれ仮想化オーバーヘッドが発生することが確認された．ただし, 仮想化オーバーヘッドが物理ネットワークを介したファイル転送時間に比べて微小な時間の処理であるため, 物

理ネットワークを介した Read のスループットに影響が出なかったと考えられる.

表 4.3　物理ネットワークを介さない時の最大レイテンシ (ナノ秒)

	ベアメタル環境	コンテナ型仮想化環境	ハードウェア仮想化環境
最大レイテンシ	25018.4	30014.3	64366.6

4.7　Web サーバ性能測定の方法

2018 年の日本のブロードバンド契約者の総トラヒック量の調査によると, 総ダウンロードトラヒックが推定で約 12.5 Tbps, 総アップロードトラヒックは推定で約 1.7 Tbps であった [12]. この調査より, 日本ではアップロードに比べ, ダウンロードのトラフィックが非常に多いことがわかる. そのため本節ではダウンロードにおける Web サーバ性能測定の方法を述べる. この測定では, コンテナ型仮想化環境とハードウェア仮想化環境に構築した Web サーバ Nginx の静的コンテンツリクエスト処理最大性能の比較を行う. Web サーバのリクエスト処理性能を上げる方法はハードウェア仮想化環境とコンテナ型仮想化環境のそれぞれの特徴を活かした方法を用いた. ハードウェア仮想化環境では 1 仮想ホスト 1 サービスを提供し, 仮想ホストのリソースを最大限使うために Nginx のワーカープロセス数を CPU コア数の 2 倍にあたる 8 とした. コンテナ型仮想化環境では迅速なスケールアウトを活かすために Pod 数を 1, 2, 4, 8, 16, 32, 64 と変化させ処理性能が最大となる Pod 数を測定する. そしてこれらの Web サーバに Web サーバ性能測定ツール ApacheJmeter を用いて 10 万リクエストを送信し, 1 リクエストあたり 600 B のファイルをクライアントへ返す. この時クライアントへ返すファイルは Web サーバのキャッシュから読み, ディスクアクセス時間を省くことで, コンテナ型仮想化環境とハードウェア仮想化環境の Web サーバの性能の差を明確にする. この測定を 5 回行い, 1 秒間あたりの処理リクエスト数の平均値を算出する.

4.8　Web サーバ性能測定の結果

Web サーバ性能測定の結果を図 4.4 に示す．コンテナ型仮想化はコンテナ間でホストのカーネルを共有することにより，マルチコンテナ環境では性能干渉が起こることが知られている [13]．そのためコンテナ数を増やせばその分だけ性能が上がるわけではなく，コンテナ型仮想化環境の 1 秒あたりの最大処理リクエスト数は Pod 数が 32 の時の 19864.9 リクエストとなった．対してハードウェア仮想化環境では 1 秒あたりの処理リクエスト数が 20512.2 であった．これらはどちらもリクエスト処理の際の CPU 使用率が 94 %であった．したがって，同じ CPU 消費でハードウェア仮想化環境の方が多くのリクエストを処理できることがわかる．さらに，第 3.2 節よりコンテナ型仮想化環境よりもハードウェア仮想化環境で使用している CPU 性能の方が 1.36 倍低いため，各仮想化環境で同一の CPU を使用した場合にはハードウェア仮想化環境の Web サーバ性能が明確に高くなると考えられる．

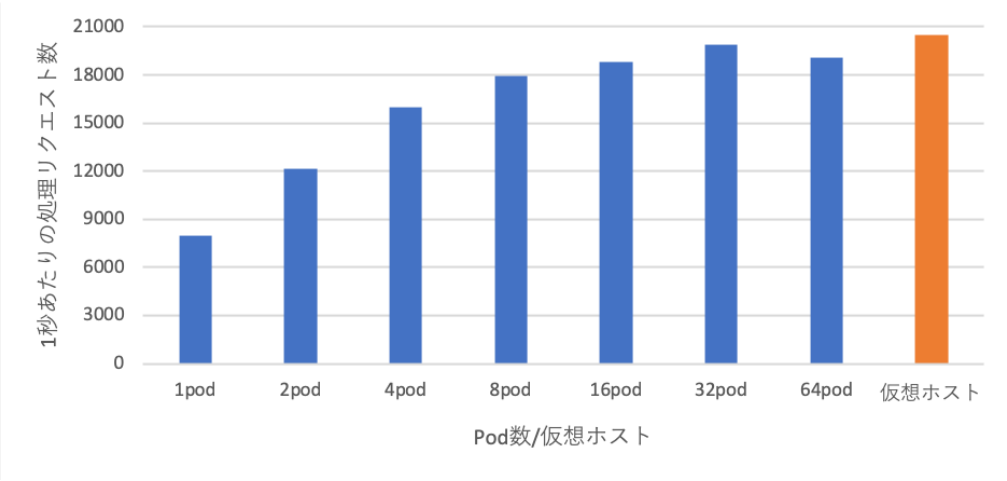

図 4.4　Web サーバ静的コンテンツリクエスト処理性能測定の結果

第 5 章

考察

5.1 各仮想化環境に適している Web サーバ運用方法

第 4.8 節でコンテナ型仮想化環境よりもハードウェア仮想化環境の方が Web サーバ性能が高いことを述べた．よって常時大量のアクセスが来るような Web サーバの運用にはハードウェア仮想化が適していると考えられる．ただし, 第 3.1 節で述べた通りハードウェア仮想化環境上の仮想ホストは起動に時間がかかるため, 突発的に非常に大量のアクセスが来た場合には対応が遅れてしまうことが懸念される．また, 突発的な大量のアクセスに備えて, 仮想ホストに多くのリソースを割いてしまうと, アクセスが少ない時に利用されない無駄なリソースが増え, リソース効率が悪くなることが考えられる．したがって, アクセス量の予測を適切に行うことが出来れば, ハードウェア仮想化のリクエスト処理性能の高さを活かした Web サーバ運用が可能になると考えられる．

コンテナ型仮想化環境はハードウェア仮想化環境に比べ Web サーバ性能では低いという結果であったが, 第 3.2 節で述べた通りコンテナは起動が早いため, 突発的に非常に大量のアクセスが来た場合でも迅速なスケールアウトによって対応できると考えられる．さらにアクセスが少ない場合にはコンテナ数をサービス可能な最低数で運用することで, 1 台の物理マシン上で独立した異なる複数の Web サービスを常時提供することができる．ただし, 第 4.8 節の通り, コンテナ数を増やし続ければ Web サーバ性能が上がり続ける訳ではないため, 性能を最大化できるコンテナ数をあらかじめ把握しておくことが, コンテナ型仮想化環境での Web サーバ運用には求められる．

5.2　実システムを想定した Web サーバの考察

第 4.7 節で示した Web サーバ性能測定の方法では, クライアントに返すファイルを Web サーバのキャッシュから読み込んでいた. しかし, 実際の Web サーバでは様々なファイルをクライアントとやり取りするため, クライアントに返すファイルが Web サーバのキャッシュにあるとは限らない. そのため, この本節ではハードウェア仮想化環境上の Web サーバが物理ネットワークを介して NFS サーバのディスクから読み込んだファイルをクライアントに返す場合を想定する. 第 4.8 節より, ハードウェア仮想化環境での Web サーバは 1 秒あたり約 2 万リクエストを返すことがわかっている. また, 第 4.4 節より物理ネットワークを介した Read では 70 MB/s のスループットが発揮できることがわかっている. これらの結果から 1 リクエストに対してクライアントに返すファイルサイズがおおよその目安として 3.5 KB 以上の時は物理ネットワークを介した Read がボトルネックとなり, 処理リクエスト数が減少していくと考えられる. さらに, クライアントへ返すファイルのサイズが大きくなるほどターンアラウンドタイムの大部分を Read の時間に割くことになる. 第 4.2 節よりハードウェア仮想化環境とコンテナ型仮想化環境の物理ネットワークを介した Read の性能に差がない為, Read に時間を割くほどハードウェア仮想化環境とコンテナ型仮想化環境の Web サーバ性能の差が無くなっていくと考えられる. よって, ファイルサイズが Web サーバ性能に影響しない程度の小さいファイルを返す Web サーバであれば Web サーバ性能の高いハードウェア仮想化環境を用いた方が良いと考えられる. しかしクライアントへ返すファイルのサイズが大きい Web サーバであれば, リソース効率や突発的なアクセスに迅速に対応できるコンテナ型仮想化が適していると考えられる.

第 6 章

まとめ

現在, ハードウェア仮想化, コンテナ型仮想化といった仮想化技術は実際のシステムを運用するための重要なプラットフォームの一つとなっている. 特にコンテナ型仮想化技術は近年流行し始めた技術であり, 仮想化環境を利用する際の選択肢の幅を広げたが, まだ利用者はコンテナ型仮想化の特性についての理解が十分に追いついていないという問題がある. 実際のサービスにおいて利用目的に適した仮想化環境を利用するためには, ハードウェア仮想化, コンテナ型仮想化それぞれの特性を理解している必要がある. 既存研究では, ベアメタル環境とハードウェア仮想化環境, コンテナ型仮想化環境のそれぞれの I/O 性能を比較した実験が行われているが, 実際のシステムを想定している実験とは言えない.

本研究では, 実際のシステムを想定しベアメタル環境とハードウェア仮想化環境, コンテナ型仮想化環境の I/O 性能を測定した. その結果, Write では仮想化オーバーヘッドがスループットに大きく影響し, Read では仮想化オーバーヘッドは存在するが, スループットに影響しないことがわかった. また, ハードウェア仮想化環境とコンテナ型仮想化環境の Web サーバ性能測定を行いハードウェア仮想化環境がコンテナ型仮想化環境よりも, 静的コンテンツリクエスト処理最大性能が高いという結果が得られた. これらの測定より, 常時大量のアクセスが来るような Web サーバの運用にはハードウェア仮想化環境が適しており, 独立した複数の Web サービスを常時稼働させたい Web サーバにはコンテナ型仮想化環境が適していると考えられる. ただし提供するファイルサイズの大きさによっては, 物理ネットワークを介した Read がボトルネックとなることも考えられる.

今後の展望として, 本研究では Kubernetes クラスタの 1 ノードでの性能測定であったため, 複数ノードへのスケールアウトによる性能の変化を測定したいと考えている. またコン

テナ型仮想化環境の CNI を変更した場合の性能測定を行いたいと考えている.

謝辞

本研究を進めるにあたり, 多くのご指導をいただいた指導教員である敷田幹文教授に深く感謝いたします. 敷田教授には卒業研究発表の練習や論文・梗概の添削などを夜分遅くまでご指導頂きました. また, 本大学にご来訪いただきコンテナに関する講習会を開いてくださり, さらには本稿で述べた性能測定の方法についてのアドバイスをして下さったヤフー株式会社/ゼットラボ株式会社の坂下幸徳様に感謝申し上げます. また，本研究の副査を担当していただいた高知工科大学情報学群 横山和俊教授，吉田真一准教授に深くお礼申し上げます. 最後に研究活動においてあらゆるサポートをしてくれた研究室一同にも心より感謝いたします.

参考文献

[1] 総務省, “平成 26 年度版 情報通信白書”, 2013, http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h26/html/nc131110.html.

[2] 水野和彦, 今田貴之, “仮想環境における I/O 性能評価及び最適化手法提案”, 情報処理学会コンピュータシステム・シンポジウム 2016 論文集, pp.86-93, 2016.

[3] 梅澤 綾果, 山口 実靖, “コンテナ型仮想化を用いた高集約環境の性能に関する考察”, 情報処理学会第 79 回全国大会, pp.1-143-1-144, 2017.

[4] IBM Research Report, RC25482 (AUS1407-001) July 21, 2014 Computer Science.

[5] Wes Felter, Alexandre Ferreira, Ram Rajamony, Juan Rubio, “An updated performance comparison of virtual machines and Linux containers”, Proceeding of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp.171-172, 2015.

[6] “Docker.”, https://www.docker.com/.

[7] 松本 亮介, 近藤 宇智朗, 三宅 悠介, 力武 健次, 栗林 健太郎, “FastContainer: Web アプリケーションコンテナの状態をリアクティブに決定するコンテナ管理アーキテクチャ”, 情報処理学会研究報告インターネットと運用技術（IOT）, 2017-IOT-38, No.14, pp.1-8, 2017.

[8] 松本 亮介, 近藤 宇智朗, 三宅 悠介, 力武 健次, 栗林 健太郎, “FastContainer: 実行環境の変化に素早く適応できる恒常性を持つシステムアーキテクチャ”, 情報処理学会インターネットと運用技術シンポジウム論文集, pp.89-97, 2017.

[9] 松本 亮介, 栗林 健太郎, 岡部 寿男, “リクエスト単位で仮想的にハードウェアリソースを分離する Web サーバのリソース制御アーキテクチャ”, 情報処理学会論文誌, Vol.59, No.3, pp.1016-1025, 2018.

[10] “Kubernetes.”, https://kubernetes.io/.

[11] IPA 独立行政法人情報処理推進機構, “2018 年度情報セキュリティの倫理に対する意識調査”, 2018, https://www.ipa.go.jp/files/000070257.pdf.

[12] 総務省, “我が国のインターネットにおけるトラヒックの集計結果 (2018 年 5 月分)”, 2018, http://www.soumu.go.jp/main_content/000568818.pdf.

[13] 飛松 秀三郎, 青田 直大, Asraa Abdulrazak Ali Mardan, 河野 健二, “コンテナ環境におけるジャーナリング I/O の制御”, 情報処理学会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）, 2018-OS-142, No.4, 2018.

付録 A

dd コマンドを用いた I/O 性能測定スクリプトの一例

```
#!/bin/bash
SUM_TIME=0.0
SUM_TP=0.0
BS=64000000
COUNT=1
for i in ‘seq 1 5‘
do
sleep 1
echo $’\n\n’ $i
RESULT=‘dd if=/dev/zero of=/mnt/write.tmp bs=$BS count=$COUNT oflag=direct 2>&1‘
echo "$RESULT" > /mnt/result.txt;
VOL=‘awk ’(NR == 3){ print $1 }’ /mnt/result.txt‘
TIME=‘awk ’(NR == 3){ print $6 }’ /mnt/result.txt‘
TP=‘awk ’(NR == 3){ print $8 }’ /mnt/result.txt‘
VOL=‘expr $VOL + 0‘
TIME=‘echo $TIME + 0.0 | bc | sed -e ’s/^\./0./g’‘
TP=‘echo $TP + 0.0 | bc | sed -e ’s/^\./0./g’‘
```

```
echo "Byte = $VOL"

echo "Time = $TIME"

echo "Throughput = $TP"

SUM_TIME=‘echo "$SUM_TIME + $TIME" | bc‘

SUM_TP=‘echo "$SUM_TP + $TP" | bc‘

rm -f /mnt/write.tmp

done

echo $’\n\n’w/r = write

echo -n BS =$’ ’

echo "$BS"

echo -n COUNT =$’ ’

echo "$COUNT"

echo -n Time_avg =$’ ’

echo "scale=7; $SUM_TIME / 5.0" | bc | sed -e ’s/^\./0./g’

echo -n Throughput_avg =$’ ’

echo "scale=2; $SUM_TP / 5.0" | bc | sed -e ’s/^\./0./g’
```