

データ利活用に向けた仮想化プラットフォーム mdx の基本性能評価

埴 敏博^{1,a)} 中村 遼¹ 空閑 洋平¹ 杉木 章義² 田浦 健次朗¹

概要: mdx は, Society5.0 で目指しているデータの利活用に向けた高性能, 柔軟かつセキュアなプラットフォームであり, 全国 9 国立大学 2 国立研究所の共同運営による稼働を始めている. 本稿では, マルチテナントに対応した仮想化プラットフォームである mdx の概要について紹介し, 主に各種ストレージの基本性能について述べる. さらに, mdx におけるソフトウェア基盤整備として, 仮想マシンテンプレートと構成管理ツール, Kubernetes によるコンテナ環境について述べる.

1. はじめに

Society 5.0 は, 地域, 年齢, 性別, 言語による格差等の課題を解消し, 地域の特色を活かした 多様な産業の活性化に貢献する社会を目指したものである [1]. データ活用社会創成プラットフォーム計画は, データ中心的研究分野や, ビッグデータ解析・データ科学の手法への期待が高い分野に対して, 高性能な情報基盤を提供するとともに, 研究分野間や産学間の連携を促進するため, 全国の大学や研究所が参加する運営, 研究コミュニティを作ることを目指したものである [2]. mdx は, データ活用社会創成プラットフォーム計画の第一歩として設計・実装されたシステムであり, 国立の 9 大学^{*1} 2 研究所^{*2} が共同で運営に当たる [3], [4], [5]. 高性能な計算ノード群と大容量のストレージを備え, 国立情報学研究所 (NII) が運用する学術情報ネットワーク SINET に直結されている. これにより, SINET に接続された全国の大学・公的研究機関と広帯域で接続され, 広域でのデータ収集機能やデータ集積機能を備えることで, 企業や自治体との共同研究も交えて日本全国でのデータ利活用の促進を目指している.

2021 年 9 月に mdx のテストリリースを行い, 一般利用を開始すると同時に, システムの機能拡張, 他システムとの連携を継続して実施している.

本稿では, mdx のシステムの技術的な詳細について述べ, 基本性能評価を行い, 設計・実装の妥当性を検証する. それとともに, ソフトウェア基盤整備として, 仮想マシンテンプレートと構成管理ツール, Kubernetes によるコンテナ環境について述べる.

2. mdx の概要

mdx は, 東京大学柏 II キャンパスに設置され, 2021 年 3 月より稼働を開始しており, ユーザ利用のための開発が続けられてきた. 2021 年 9 月にはテストリリースを行い, NII が運用する学術認証フェデレーション「学認」[6] を用いたシングルサインオン, または mdx ローカル認証により利用を始めることができる.

mdx は以下の 3 つの特徴を備えている.

(1) データ収集機能

IoT デバイスからのデータや, 大規模リアルタイムデータを円滑に収集できるよう, セキュアで大容量の通信回線を提供する.

mdx は NII が運用する学術情報ネットワーク SINET5[7] に直結されており, 2022 年 2 月現在は 100 Gbps ×2 で接続されている. 2022 年 4 月からは SINET6 に移行し, 400 Gbps×2 に増速される予定である.

さらに SINET ではモバイル網を SINET の足回りとして活用する広域的なデータ収集基盤 [9] を提供している^{*3}. これは各プロジェクトごとに, モバイル仮想閉域網 (Mobile Virtual Private Network; Mobile VPN) を提供し, その VPN をクラウドなど計算基盤まで延

¹ 東京大学 情報基盤センター

² 北海道大学 情報基盤センター

^{a)} hanawa@cc.u-tokyo.ac.jp

^{*1} 北海道大学情報基盤センター, 東北大学サイバーサイエンスセンター, 筑波大学人工知能科学センター, 東京大学情報基盤センター, 東京工業大学学術国際情報センター, 名古屋大学情報基盤センター, 京都大学学術情報メディアセンター, 大阪大学サイバーメディアセンター, 九州大学情報基盤研究開発センター

^{*2} 国立情報学研究所, 産業技術総合研究所

^{*3} SINET6 では 5G モバイル網+ローカル 5G が提供される予定である [10].

伸することを可能にしたものである。

mdx では 3.3 節で述べるように、システムの外部接続ネットワークの設計を行うに当たり、当初からこうしたモバイルデバイス群と mdx 上のプロジェクトとが VPN が接続できるような環境を想定してきた。

(2) データ解析機能

高度な解析を実現するため、最新のスーパーコンピュータと同等の高性能計算環境やストレージを提供する。

mdx では、複数プロジェクトに同時にセキュアな環境を提供するため、計算ノード、ネットワーク双方に仮想化技術を用いて、各プロジェクトに対し 1 つのテナントを割り当てる。

(3) アプリケーションプラットフォーム機能

mdx では、単なる解析環境ではなく、アプリケーションのためのプラットフォームを構築する基盤としての側面も備える。例えば、「マテリアル先端リサーチインフラ」[11] のように、ある分野におけるデータレポジトリを提供するプラットフォームの基盤としても期待されている。サービス提供のためには、インターネットとの柔軟かつセキュアな連携が必要であり、加えて大規模なデータ解析や機械学習の処理要求に対してオンデマンドに計算資源を提供することが求められる。

3. mdx の構成と設計

mdx の全体構成を図 1 に示す。

mdx を構成するハードウェアは大きく分けると以下の要素からなる。

- 計算ノード：汎用 CPU ノード、演算加速 GPU ノード
- ストレージ：高速 NVMe ストレージ、大容量 HDD ストレージ、仮想ディスクストレージ、外部共有 S3 オブジェクトストレージ
- ネットワーク：外部接続ネットワーク、内部高速ネットワーク

全体の消費電力は 630 kW である。そのうち、計算ノードについては水冷、その他の装置は空冷である。

3.1 計算ノード

汎用 CPU ノードは 368 ノードの Intel Xeon IceLake 2 ソケット搭載、演算加速 GPU ノードは、ノードあたりに Intel Xeon IceLake 2 ソケットに加えてさらに NVIDIA A100 Tensor Core GPU を 8 基搭載したものが 40 ノードで構成されている。

表 1 に計算ノードの仕様を示す。いずれも、Wisteria/BDEC-01 Aquarius[12] などの最新のスパコンにも採用されている CPU、GPU と同等以上のものが採用されている。

演算加速ノードについては、文献 [12] に示した Aquarius ノード (W-Aquarius) と同様の構成であり、2 GPU で 1 つ

の内部ネットワークインタフェースを共有する。異なる点は、CPU の内蔵コア数、W-Aquarius が IB-HDR 200 Gbps であるのに対し、mdx では RoCE 100 Gbps を用いている点と、mdx は外部インタフェースを 2 つ備えていることである。

全ての計算ノードは仮想化環境で動作する。仮想化ソフトウェアには VMWare 社 vSphere を用いている。各計算ノードにはハイパーバイザとして ESXi が搭載され、その上でゲスト OS が動作する。

演算加速 GPU ノードにおいては、各 VM から GPU へは PCIe パススルー方式で接続する。つまり、通常のペアメタルサーバと同等の直接アクセスを実現する。

通常ノード上では、複数プロジェクトの VM が同時に動作するが、専有ノードでは、単一のプロジェクトのみが専有して利用することができる。

3.2 ストレージ

ストレージには、計算ノードのみから参照可能な内部ストレージと、外部と共有するための外部ストレージの 2 種を備えている。

表 2 に、ストレージの仕様を示す。

内部ストレージには、プロジェクト内でデータを共有するなど work 領域として利用を想定している、Lustre ファイルシステムによる、高速 NVMe ストレージ、大容量 HDD ストレージを備えている。

加えて、ESXi 上で動作する仮想マシンのディスクイメージを格納する、仮想ディスクストレージも別途備えており、VMWare のデータストアとして管理されている。

Amazon AWS S3 互換の外部共有ストレージも備えている。外部との共有の他、mdx のプロジェクト間での共有にも用いる。

Lustre ファイルシステムについては、各 VLAN に異なるサブディレクトリを割り当てる、マルチテナント機能を使用しており、同じサーバであっても異なる VLAN 間では完全な分離を実現している。通常のクラウドサービスでは、それぞれのテナントに独立のストレージデバイスを割り当てるが、割り当てた結果、規模が小さい場合には、高いバンド幅性能が望めない。単一の Lustre ファイルシステムであっても、マルチテナント機能で完全に分離できるため、性能とセキュリティを両立できる。

一方、専有ノードを用いるプロジェクトに対しては、Lustre の OST のうち、専用の OST を割り当てることにより、他のプロジェクトと完全に隔離された環境を構築することができる。

3.3 ネットワーク構成

mdx のネットワークは、仮想マシンがインターネットと通信するための外部接続ネットワークと、仮想マシン同士

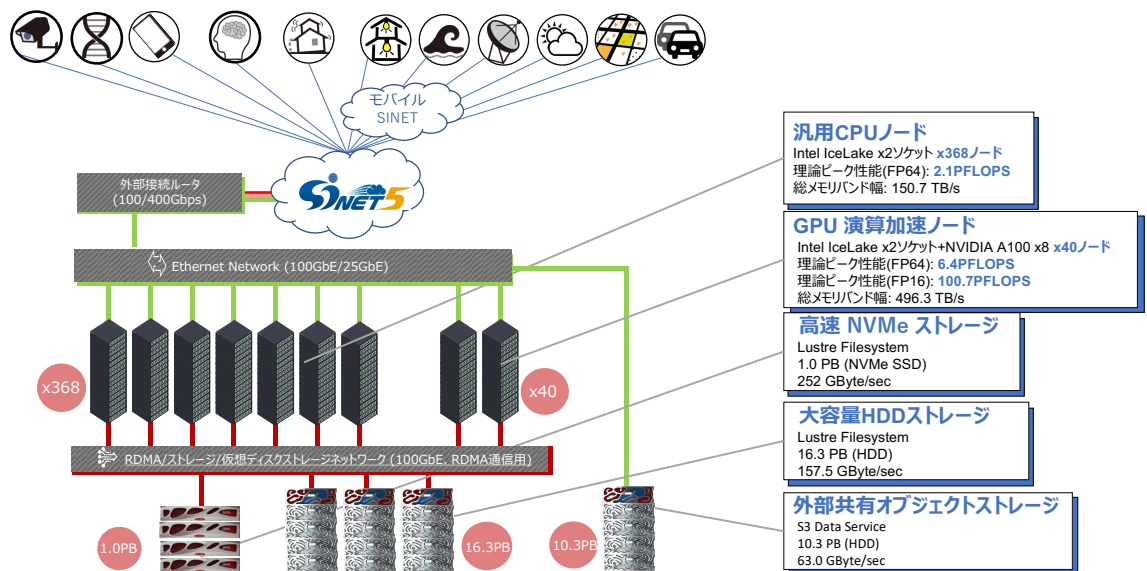


図 1 mdx の概要



図 2 mdx の外観：(左) ストレージラック (中) ネットワークラック (右) 計算ノードラック

が高速に通信するための内部高速ネットワークに分かれている。

外部接続ネットワークは国立情報学研究所の運用する学術情報ネットワークである SINET[7] と 200 Gbps で接続されており、さらに SINET の L2VPN サービス [8] を用いて外部から VLAN を mdx 内に持ち込むことが可能である。外部接続ネットワークは、図 3 に示すように、Juniper Networks MX480 2 台をコアルータとし、QFX10002 を Spine、QFX5120 を Leaf とする Spine Leaf トポロジで構成されている。またプライベートアドレスを持つ仮想マシンに対して外部との通信時の Network Address Translation (NAT)、及び外部から仮想マシンにパブリックアドレスをマッピングする機能を SRX4600 が行っている。ハイパーバイザは 25 Gbps NIC で外部ネットワークに接続されており、仮想マシンは最大 25Gbps で外部及び仮想マシン同士で通信可能である。

一方、内部高速ネットワークは、複数の仮想マシンが連携して高性能計算を行うため Remote Direct Memory Access (RDMA) をサポートした広帯域なネットワークである。内部高速ネットワークの構成を図 4 に示す。内部高速ネット

ワークは NVIDIA/Mellanox SN3700 による Spine Leaf トポロジで構成されており、ハイパーバイザは 100 Gbps NIC で接続され、そのバイセクションバンド幅は 100% である。RDMA を行うネットワークを構成する際は InfiniBand が一般的だが、mdx ではクラウドとしてのユーザ同士のネットワークを分離する必要があるため、Ethernet によるネットワークを選択する必要がある。InfiniBand の pKey によるパーティション機能では、Subnet manager の乗っ取りを完全には防ぐことができないため、InfiniBand では要求を満たせない。一方、Ethernet では、leaf 側をポート VLAN に設定することで、物理的に分離することができる。

Ethernet 上で RDMA を行うにあたっては RDMA over Converged Ethernet version 2 (RoCEv2) を採用した。

いずれのネットワークについても、VXLAN (Virtual eXtensible Local Area Network) によるオーバーレイネットワークを用いている。これにより、物理ノードやスイッチと VLAN の構成を分離し、複数の物理ノードにまたがった VLAN を動的に作成・削除することが可能になる。具体的には、各プロジェクトに 1 つ以上の VLAN を割り当てているが、全体として数千の規模を収容可能である。

表 1 計算ノードの仕様

	汎用 CPU ノード	演算加速ノード
ノード	富士通 PRIMERGY CX2550 M6	富士通 PRIMERGY GX2570 M6
ノード数	368	40
理論ピーク性能 (FP64)	2.1 PFLOPS	6.4 PFLOPS
(FP32)	4.2 PFLOPS	6.7 PFLOPS
CPU	Intel Xeon Platinum 8368 (IceLake, 38 コア/76 スレッド, 2.4 GHz) ×2 ソケット	
メモリ	256 GiB	512 GiB
バンド幅	DDR4-3200 ×8ch, 409.6 GB/s	
GPU		NVIDIA A100 Tensor Core GPU 40 GiB ×8 基 NVLink 3.0+NVSwitch 接続, GPU 間 300 GB/s
外部 IF	Intel XXV710 25 GbE	
搭載数	1	2
内部 IF	Mellanox/NVIDIA ConnectX-6 Dx 100 GbE	
搭載数	1	4

表 2 ストレージの仕様

	内部ストレージ			外部ストレージ
	高速 NVMe	大容量 HDD	仮想ディスク	外部 S3 オブジェクト
サーバ	DDN ES7990X	DDN ES400NVX	DDN IntelliFlash HD2160	DDN ES7990X
台数	4	5	2	2
ファイルシステム	Lustre File System			DDN S3 Data Service on Lustre File System
容量 (PB)	1.0	16.3	0.55	10.3
バンド幅 (GB/s)	252	157.5	31.5	63.0

新規仮想マシンの作成時や削除時のネットワーク構成変更は mdx ポータルから Ansible によって行われる。IaaS 環境で仮想マシンを VLAN に接続する場合、ハイパーバイザ間でオーバーレイネットワークを利用するのが一般的であるが、mdx では Leaf スイッチからハイパーバイザに VLAN を出す構成を採用している。その理由として、RDMA を行うためには仮想マシンに SR-IOV の仮想インタフェースをアタッチする必要がある、その場合ハイパーバイザの仮想スイッチ機能を利用することはできない。またハイパーバイザの仮想スイッチでオーバーレイや Layer-3 ルーティングを行う場合は性能上の懸念があった。これらの理由から、mdx では Leaf スイッチから VLAN を出す構成となっている。

3.4 VM とネットワークの接続

VM から外部接続ネットワークへは、VMWare が提供する準仮想化インタフェースである、VMXNET3 を用いて接続する。

一方、内部高速ネットワークへの接続方式として、

- (1) VMware デフォルト方式 (TCP 接続)
- (2) 準仮想化 RDMA (PVRDMA)
- (3) SR-IOV

を提供している。(1), (2) は VMWare によって提供されており、(3) は NVIDIA/Mellanox によるドライバをインストールすることで可能になる。

ノード間の RDMA 転送は (2)(3) で可能であるが、Lustre への RDMA 転送は (3) のみ可能である。

4. 仮想マシンテンプレートと構成管理ツール machine-configs

mdx では、プロジェクト単位でのコンピューティング環境の分離を実現するために、仮想マシンとネットワーク仮想化機能を採用している。mdx ユーザは、仮想マシンに任意の OS をインストールし、自身の用途に沿ったコンピューティング環境を構築できる。一方で、mdx のハードウェア機能である GPU やネットワーク、Lustre 機能を利用するためには、各種デバイスドライバのインストールと設定が必要であり、既存のクラスタ環境に比べて環境構築の手間が必要になる。mdx ではこれらのユーザ環境構築を支援する目的で、仮想マシンテンプレートと仮想マシンの構成管理スクリプトである machine-configs を提供している。本節では、mdx が提供するこれら機能の詳細を述べる。

4.1 仮想マシンテンプレート

mdx の仮想マシンテンプレートは、mdx を構成するハードウェアリソースの利用に必要なデバイスドライバと設定を実施したインストール済み OS イメージである。仮想マシンテンプレートの OS イメージは、HashiCorp の Packer[16] で作成している。Packer を利用することで、任

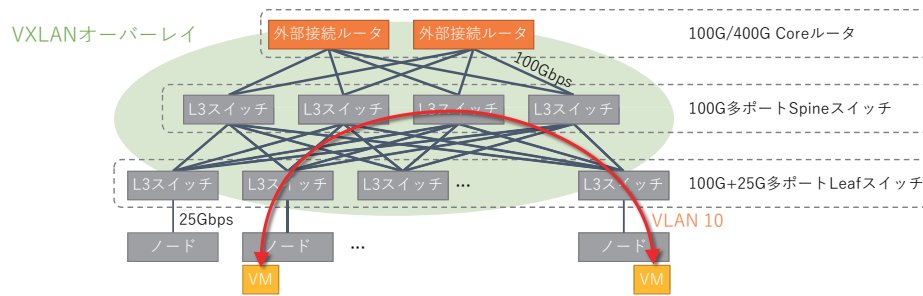


図3 外部接続ネットワークの構成

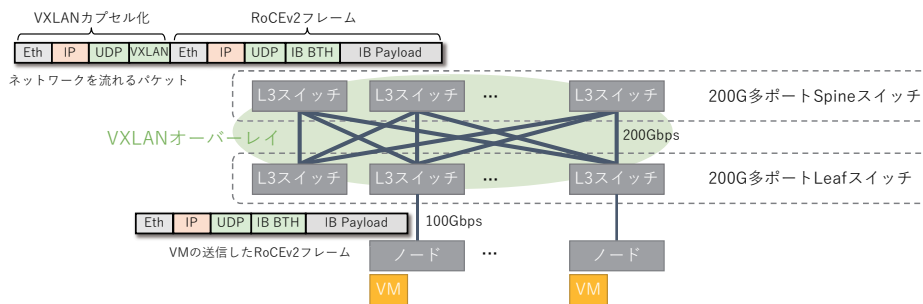


図4 内部高速ネットワークの構成

意の OS イメージ形式でインストール済みの OS イメージを作成できる。mdx の仮想マシン基盤は VMWare 社の vSphere を採用していることから、Packer を使用して OVA 形式の OS イメージを作成し、mdx ポータルのテンプレート一覧に登録している。OS イメージの初期設定は、Packer の ansible 機能を使用し、OS イメージ作成時にセキュリティ設定やライブラリソフトウェアなどのインストールを実施している。

mdx では、mdx 上に仮想マシンテンプレートビルド環境を構築しており、定期的に OS 付属のパッケージアップデートやカーネルアップデートを実施し、提供テンプレートを更新している。mdx ユーザは、ポータル画面から任意の仮想マシンテンプレートを選択して自身のプロジェクトにデプロイする。現在、mdx では、Ubuntu20.04 の server と desktop を、それぞれ GPU ドライバの有り無しのテンプレートで提供している。GPU テンプレートでは、GPU ドライバをインストールして OS イメージを配布している。また、GPU 周りの設定としては、Persistence mode を有効化設定することで、OS 起動時に GPU 依存ライブラリとドライバの読み込みするように設定している。今後、mdx では、CentOS7 などの Ubuntu 以外の OS イメージや、利用シーンを想定してのアプリケーション込みのテンプレートを提供する。例えば、大規模オープンデータを Web 公開するなどの用途を想定し、よりセキュリティを配慮したテンプレートなどの提供を検討している。

また、mdx の提供する OS イメージは、mdx のハードウェア用に修正した Lustre ライブラリと NIC ドライバをインストールした状態で提供している。このような独自ビ

ルドした Linux のカーネルモジュールは、OS のカーネルバージョンが変化するたびに、カーネルモジュールの再コンパイルが必要になる。そのため、mdx のような管理者がユーザ OS にアクセスできない仮想マシン環境では、これら Linux の out-of-tree のカーネルモジュールの管理方法が課題になる。そこで、現在 mdx では、mdx 側が提供している Linux カーネルモジュールをすべて、Linux の dkms フレームワークを有効化した状態でカーネルモジュールをビルドしている。Linux の dkms フレームワークを使用することで、ユーザがカーネルバージョンを更新した際に、自動で dkms モジュールが再コンパイルされる。

4.2 machine-configs

仮想マシンテンプレートは、OS のインストール作業の手間を省くことができるが、一方で、ユーザごとで異なるような設定をテンプレートに埋め込むことが難しい。そこで、mdx では、デプロイ後の Linux 環境を対象に、プロジェクト内の複数台の仮想マシンを一括してセットアップする構成管理スクリプトである machine-configs を提供している [17]。machine-configs は、Ansible のスクリプト集であり、Github を通して公開している [18]。machine-configs を使うことで、よくある計算機環境の設定、例えば、NFS サーバとクライアントの設定や、LDAP によるユーザ管理、CUDA Toolkit、Jupyter 環境の設定などを、複数仮想マシンに対して一括してインストールと設定ができる。

mdx では、今後も仮想マシンテンプレートや machine-configs のような、ユーザ支援機能の拡充を継続していく。

5. 基本性能評価

5.1 CPU のメモリバンド幅

演算加速 GPU ノード上に構成した 8 GPU を確保したノード最大サイズの VM を用いた。このとき、利用可能な CPU コア数は 144 である。また、NUMA ドメインは 2 つ存在していた。実際には HyperThreading が有効になっており、物理コアとしては 72 コアを使用するのが妥当である。従って、両ドメインから 36 コアずつ使用する。

このような条件で、CPU のみで stream ベンチマークを測定したところ、Triad で最大 **265.0 GB/sec** を得た。

他のベアメタルな IceLake 環境（但し CPU は 72 コア）を使った結果では 322.0 GB/sec が得られており、VM 上では約 85% の性能が出ていることがわかる。

5.2 ノード内 GPU 間通信

前述と同じ環境で、NVIDIA 社が提供する p2pbandwidthLatencytest ツールを用いて、ノード内の GPU 間の通信レイテンシとバンド幅を確認した。

物理的には、GPU 間は NVLink および NVswitch で接続されており、同様のアーキテクチャを持つ Wisteria-Aquarius では、P2P モード enable では 275 GB/s の性能が得られている。

一方、mdx では、ハイパーバイザが間に入り PCIe パススルーを用いるため、P2P モードは enable にならないため、上記のような性能は得られず、最大 20 GB/s 程度の性能である。

ただし、Wisteria-Aquarius でも P2P disable の場合は 20 GB/s 程度の性能である。これは PCIe の性能を表しており、VM においてもパススルー接続しているため、ベアメタルと同等の性能が得られていることがわかる。

5.3 ストレージ性能

今回は、各ストレージのバンド幅性能を測定した。用いたのは IOR-3.4.0+dev (2022/2/5 commit のもの)[14] で、libS3 オプションを有効にしてビルドした。libs3 は執筆時 (2022/2/18) の最新版 [15] を用いた。

測定は、演算加速 GPU ノード上に構成した 8 GPU を確保したノード最大サイズの VM を用いた。

試行は各 3 回行い、試行のうちで最高の性能の結果を採用した。

実行時のコマンドラインはそれぞれ以下の通りである。

- POSIX

```
mpirun -np $procs ior -w -r -C -g -i 3 -s  
$segcount -b $blksize -t $tsfsize -a POSIX  
-e -F -o dir --posix.odirect
```

- POSIX (Direct IO)

```
mpirun -np $procs ior -w -r -C -g -i 3 -s  
$segcount -b $blksize -t $tsfsize -a POSIX  
-e -F -o dir --posix.odirect
```

- S3-libS3

```
mpirun -np $procs ior -w -r -C -g -i 3  
-s $segcount -b $blksize -t $tsfsize  
-a S3-libS3 --S3-libS3.host=s3ds.mdx.jp  
--S3-libS3.bucket-name=prefix=ior  
--S3-libS3.access-key=...  
--S3-libS3.secret-key=...
```

評価に用いたストレージは以下の通りである。

local OS 起動ディスク=仮想ディスクストレージに対するアクセス (ローカル)

large 大容量 HDD ストレージに対する RDMA アクセス

large-TCP 上記ストレージへの TCP アクセス

fast 高速 NVMe ストレージに対する RDMA アクセス

fast-TCP 上記ストレージへの TCP アクセス

S3 S3 オブジェクトストレージに対するアクセス (TCP)

blksize, tsfsize は、S3 を除いて全て 1m (1 MB) に設定した。segcount は、容量に余裕のある large, fast は 8192 に設定した。local については、容量の制限があったため、合計書き込みサイズが最大 32 GB になるように segcount を調整した。S3 については、blksize, tsfsize に動作可能な最大サイズ 4g (4GB) に設定し、segcount は 1 にした。

図 5 に、POSIX インタフェースによる結果を示す。グラフ中の数字は、用いたプロセス数=並列数 (上記の procs) を示している。まず、local の読み出しが他に比べて突出して高い値を示しているが、これは VMWare のデータストアを利用しているため、キャッシュの効果が高いことが考えられる。一方で、書き込み性能はスケールせず、並列数によらずほぼ 2,000 MiB/s 程度である。

右図からは、16 並列までは RDMA アクセスはよくスケールしている。しかし、32 並列、64 並列では極端に性能が低下しており、ネットワークインタフェースデバイスの制御が飽和している可能性がある。TCP アクセスでも、読み出しの場合には 8 並列までは RDMA と同等の性能が得られているが、それを超えると著しく性能が低下している。RDMA アクセスでは、100 Gbps × 4 リンク (インタフェースとしては 8 個に見える) を用いているが、TCP アクセスでは 100 Gbps 1 個しか利用できないため、このような差が生じると考えられる。

図 6 に、Direct IO オプションをつけた場合の結果を示す。RDMA の場合でも、Direct IO オプションなしの場合に比べて 1/7~1/4 に低下しているが、32 並列までは、並列数に応じて良くスケールしている。また、large の読み出し性能が書き込み性能に比べて約半分の性能に低下しているのに対し、fast の読み出し性能は書き込み性能と大き

な違いが無い。

図 7 に、S3 オブジェクトストレージでの結果と、TCP による large, fast の結果を併せて示す。性能差は最大で large で約 7.5 倍 (2 並列), fast で約 18 倍 (8 並列) であった。しかし、ネットワークインタフェースが large, fast の場合は 100 Gbps であるのに対し、S3 は 25 Gbps^{*4}であり、それを考慮すると、実質 4~5 倍程度の差になる。

6. 高性能な Kubernetes 環境構築の検証

本項目では、運用開始時点の mdx の調達には含まれていないコンテナおよび Kubernetes を基礎としたデータ利活用に向けたソフトウェア基盤整備に関する機能性および性能の評価を行う。

本項目では、単に mdx 環境上で Kubernetes を稼働させるだけではなく、下記の検証を目標とする。

- mdx が備える高性能なサーバ、ネットワークおよびストレージを最大限活用し、従来のスパコン性能に近い高性能な Kubernetes 環境を実現する。
- 上記の Kubernetes 環境上で、データ収集・解析、機械学習や AI、HPC アプリケーションおよびそれらのワークロードの混在に関する研究を実施する。
- 学術的なアプリケーションコンテナの流通やコンテナ間のサービス連携による mdx を中核とした複数学術機関の連携、および学際的な研究環境の構築を実施する。
- mdx のみに限らず、将来的な学術基盤整備に関する知見を得る。

上記の目標には長期的なものも含まれているが、本論文では、これらの検証の最初の成果について報告する。

6.1 検証環境

6.1.1 仮想マシン構成

構築したプロトタイプ環境の概要を図 8 に示す。仮想マシンは Kubernetes クラスタを構成するサーバと、ログインサーバなどの周辺サーバで構成されている。

周辺サーバは、SSH で mdx 外部との接続性を確保する踏み台サーバ (bastion) が 1 台と、プライベートコンテナレジストリサーバ (harbor) が 1 台で構成されている。

Kubernetes クラスタでは、マスタ (kube-master) を 1 台配置し、計算ノードとなる CPU ノード (kube-cpu) と GPU ノード (kube-gpu) が各 2 台で構成されている。

各グループでのノードは必要最小限の台数としており、必要に応じて増減させることが可能である。特に mdx では、台数の増減も容易であるが、構成変更による CPU コア数や GPU 数の増減も仮想マシンを停止させるだけで可能であり、小規模インスタンスで検証を行い、152 仮想コアなどの mdx の物理ノードをほぼ専有するようなインス

^{*4} 実際には 25 Gbps×2 であるが、ハイパーバイザが正しく利用できているか不明である。

表 3 コンテナ検証環境のソフトウェア構成
Table 3 List of Kubernetes software components

構成要素	実行ランタイム
Kubernetes	k0s
CRI	containerd (標準) NVIDIA Container Toolkit (GPU ノード)
CSI	Exascaler File CSI Driver
CNI	Kube-router (標準) Multus CNI (複数ネットワーク用)
負荷分散	Metal LB (L2 モード)
監視	Prometheus, Grafana
運用	LENS
HPC サービス	MPI Operator (Kubeflow)

タンスで本格計算を行うことも可能である。

検証環境では、周辺サーバに 4 CPU パック、Kubernetes クラスタを構成する各ノードに 18 CPU パックあるいは 1 GPU パックを使用した。仮想ディスクは各ノードに 100GB 割り当てたが、既に 50%以上のディスク占有率となっている。Kubernetes クラスタでは、複数ネットワークの機能性検証から、セカンダリネットワークも配線した。

また、Kubernetes クラスタを構成する各ノードでは性能面の検証から SR-IOV 対応の仮想マシンを使用し、周辺サーバでは Port Group を使用した。PVRDMA 対応の CPU ノード (kube-pv) についても、検証を進めている。

仮想マシンテンプレートは、東京大学が提供する 00 Ubuntu-2004-server (Ubuntu 20.04.3) の CPU 版および GPU 版を使用した。

DNAT および ACL 設定による外部接続性は踏み台サーバに対する SSH 接続のみとし、Kubernetes 上で稼働する各コンテナサービスに対しては、Metal LB を経由して DNAT で必要に応じて割り当てる。

6.1.2 ソフトウェア構成

検証環境のソフトウェア構成を表 3 に示す。mdx が採用する VMware vSphere と親和性の高い VMware Tanzu もあるが、計算ノードのオーバーヘッドを極力低減する目的から k3s や k0s などの軽量 Kubernetes ディストリビューションを採用する。本検証環境では、k0s を使用した。

コンテナ間の性能分離に関しては、Topology Manager, CPU Manager, および Memory Manager を導入した。ある程度、NUMA を考慮したコンテナ配置が可能となっているが、実際の CPU 割り当ては vSphere ESXi の制御下にある。さらには、Intel CPU Manager for Kubernetes (cmk) も導入可能である。

コンテナランタイム (CRI) は k0s 標準の containerd を用いるが、GPU ノードでは NVIDIA Container Toolkit を使用し、NVIDIA K8s Device Plugin および GPU Feature Discovery も導入した。

コンテナネットワーク (CNI) に関しては、k0s 標準の

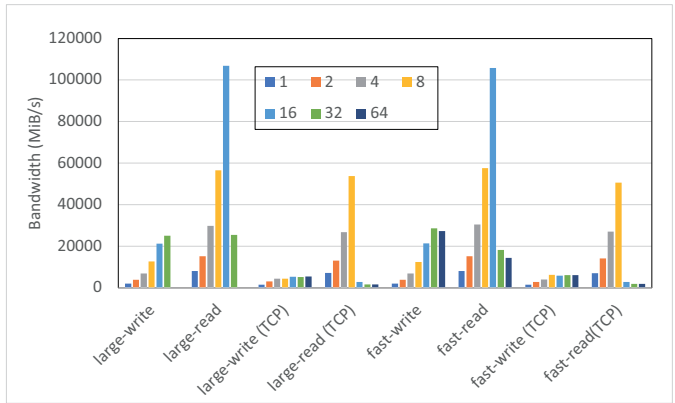
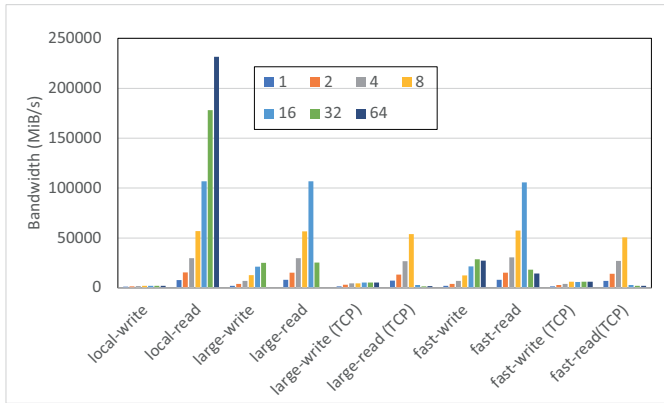


図 5 POSIX インタフェースによる各ストレージバンド幅 (右図は local を除いたもの)

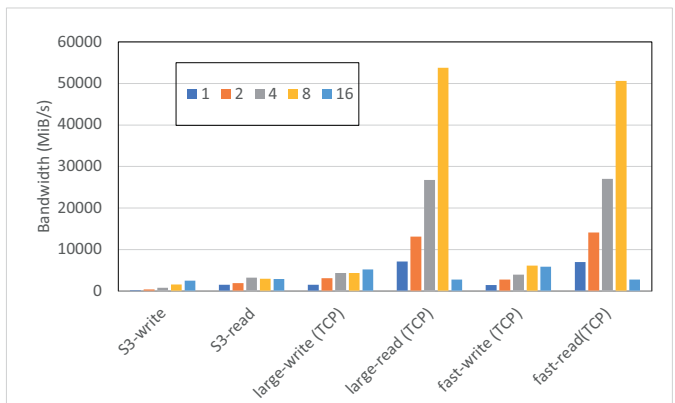
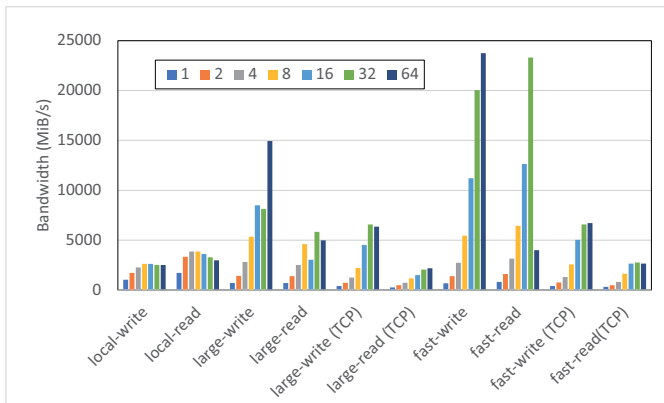


図 6 POSIX インタフェース (Direct IO 付き) の各ストレージバンド幅

図 7 S3 インタフェースのバンド幅比較

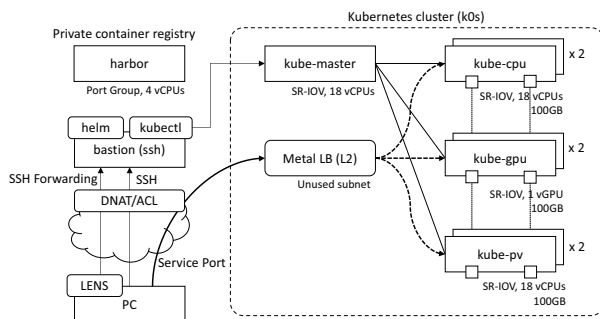


図 8 コンテナ検証環境のシステム構成

Fig. 8 System setup for Kubernetes cluster

Kube-router を当初用いるが、Multus CNI により Pod に複数のインターフェースを割り当てることが可能となっている。将来的には、eBPF を基礎とした Cilium など、他の CNI の性能や機能の検証も行う。

コンテナストレージ (CSI) に関しては、DDN が提供する Exascaler File CSI Driver を使用し、コンテナに対して、高速内部ストレージ (/fast) および大容量内部ストレージ (/large) から動的プロビジョニングを行い、コンテナの UID および GID で読み書き可能であることを確認した。

プライベートコンテナレジストリは Harbor を使用した。これは将来的な運用を考慮して選択するとともに、Trivy

などのコンテナイメージの脆弱性診断サービスとの連携も想定している。

6.1.3 アプリケーションの検証

アプリケーションでは、HPC 利用も想定し、MPI アプリケーションに関するもの、データの収集や解析に関するものの 2 つのグループで検証を進めている。

本研究ではまず mdx 上に構築された Kubernetes 環境の性能評価を目的として、Kubeflow の一部として開発されている MPI Operator を導入した。MPI Operator 記述例の TensorFlow Benchmarks (tf_cnn_benchmarks) にて、GPU ノードを 2 台使用し、最適化を行わない状態で 835.28 画像/秒以上のスループットが得られている。

データの収集や解析に関しては、まだ機能性の評価にとどまっているが、Kubernetes ネイティブ実行の Apache Spark、および Strimzi 版の Apache Kafka が既に動作している。

一般に Kubernetes 上で稼働するアプリケーションの配布には、YAML ファイル、Helm、および Operator の三つの方法がある。今後、mdx における学術アプリケーションの提供や配布において、どの方法が最適か検討を進める予定である。

6.2 現状と今後の予定

評価は引き続き進めているが、本プロトタイプの成果はそのまま mdx における Kubernetes 利用のリファレンス実装となる。今回、mdx を利用した感想として、計算機の世代が新しく、非常に高速に仮想環境の構築や破棄ができる。特に、ストレージ性能が非常に高いため、仮想マシンのデプロイも速い。mdx ポータルも慣れれば、かなり扱いやすい。

今後は論文執筆時点では間に合わなかったコンテナ内部における RDMA 転送 (ROCEv2) の検証や、性能の最適化を進める予定である。

また、mdx の Kubernetes 環境上におけるセキュリティやプライバシーの強化も進める。mdx では、強い分離を一部導入するとともに、Nested VM の実行が許可されており、実際に Firecracker によるマイクロ VM が起動した。今後、Kata Containers などの動作検証を進めるとともに、認証やアクセス権の制限を導入する。なお、Intel SGX は現時点においてゲスト仮想マシン内部からは利用が許可されていない。

さらには今後、検証作業が収束した段階で machine-configs から派生した Ansible Playbook 化を進める予定である。これにより、mdx に最適化された Kubernetes 環境の構築が容易となる。本研究と GKS, AKS, EKS などのパブリッククラウドによる Managed Kubernetes との比較は、そのまま mdx とパブリッククラウド利用の比較となる。本研究では、標準的な Kubernetes コンポーネントを採用しつつ、mdx への最適化を進める。また、Rancher Fleet などによる複数 Kubernetes クラスターの運用も検証する予定である。最近、バージョン 1.21 以降の Kubernetes では、Indexed Jobs や Suspended Jobs などのバッチ機能の強化が進められている。これらをもとにした HPC スケジューラの実装・評価や、先行研究 [13] の成果をもとに、SPIFFE/SPIRE によるマイクロサービス層での学術機関サービス連携も進める予定である。

7. おわりに

mdx は、データ利活用に向けて、高性能とセキュリティの両立を目指して仮想化技術を取り入れたプラットフォームであり、2021 年 9 月のテストリリース以降、試験利用が継続しており、2022 年度中には本格利用に移行する予定である。本稿で紹介した、ソフトウェア基盤の整備を充実させ、利用者への利便性を高めるとともに、今後、実際に学際的な応用への適用を加速していく予定である。

謝辞 日頃から mdx の運営に携わっている関係者の皆様、運用や機能拡張にご尽力いただいている富士通株式会社、VMware 関係者の皆様に深く感謝いたします。

**本研究は JSPS 科研費 20K11837 の助成を受けたものです。

参考文献

- [1] 内閣府: Society5.0 とは, https://www8.cao.go.jp/cstp/society5_0/ (2019)
- [2] データ活用社会創成プラットフォームの推進に向けた当面の整備方策, データ活用社会創成プラットフォームの推進に関する有識者会合, 文部科学省, https://www.mext.go.jp/b_menu/shingi/chousa/shinkou/059/siryo/001.html (2020).
- [3] データ活用社会創成プラットフォーム mdx を導入～9 大学 2 研究機関が共同運営しデータ活用の産学官連携・社会実装・研究を推進～(2021).
- [4] 中島他: Society 5.0 実現に向けた (計算+データ+学習) 融合、255-257、大学 ICT 推進協議会 2019 年度 年次大会 (2019).
- [5] 鈴木他: データ活用社会創成プラットフォーム mdx の設計・実装・運用～多様な学際領域における共創に向けて～, 大学 ICT 推進協議会 2021 年度 年次大会 (2021).
- [6] 学術認証フェデレーション「学認 (GakuNin)」, 入手先 (<https://www.gakumin.jp/>)
- [7] 学術情報ネットワーク SINET5, <https://www.sinet.ad.jp>
- [8] L2VPN/VPLS, https://www.sinet.ad.jp/connect_service/service/l2vpn
- [9] SINET 広域データ収集基盤実証実験, <https://www.sinet.ad.jp/wadci>
- [10] 次期学術情報ネットワーク SINET6, <https://www.sinet.ad.jp/sinet6>
- [11] 文部科学省「マテリアル先端リサーチインフラ」https://www.mext.go.jp/b_menu/boshu/detail/material_research_results_00001.html
- [12] 堀他: 「計算・データ・学習」融合スーパーコンピュータシステム Wisteria/BDEC-01 の性能評価, 情報処理学会研究報告, Vol. 2021-HPC-180, No. 22, pp. 1-8 (2021).
- [13] 杉木: 高水準なマイクロサービス層における複数ドメインを連携させたインタークラウド HPC 環境実現の検討, 情報処理学会研究報告, Vol. 2021-OS-153, No. 9, pp. 1-6 (2021).
- [14] HPC IO Benchmark, <https://github.com/hpc/ior>
- [15] libs3 library, <https://github.com/bji/libs3>
- [16] Packer by HashiCorp, <https://www.packer.io/>
- [17] mdx: 複数仮想マシンによるクラスターの作成例, https://docs.mdx.jp/ja/main/deploy_cluster_example.html
- [18] mdx: machine-configs リポジトリ, <https://github.com/mdx-jp/machine-configs>