データ利活用に向けた
高性能Kubernetes環境構築の検討

杉木 章義
（北海道大学情報基盤センター）

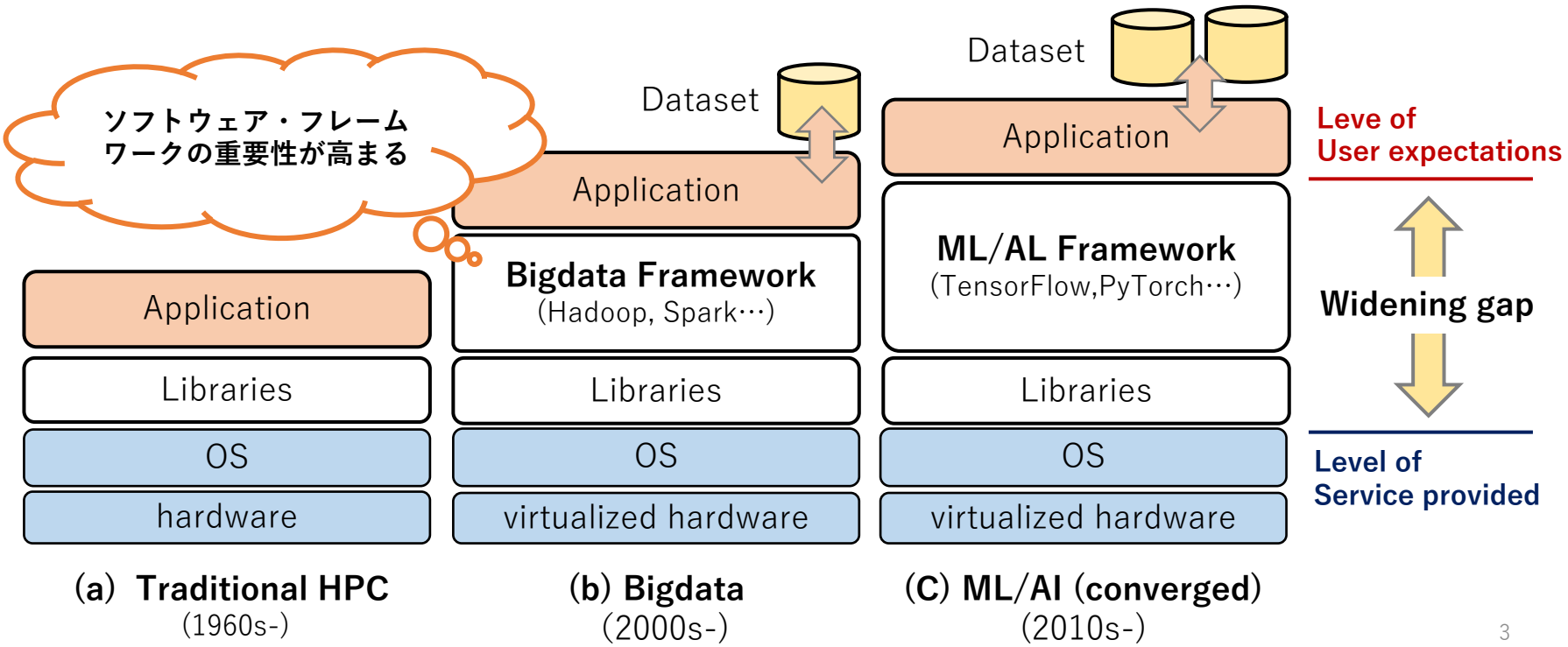
背景：HPC利用の拡大

- **従来のスパコン・学術クラウドでは対応できない計算資源要求が増加している**
  - 大規模データ処理，ML/AIと様々な学術分野への応用
    - 米国 National Research Cloud (NRC)の提唱
      - クラウド基盤，データに基づく探索，シミュレーションを21世紀型研究における三つの柱と定義
- **近年，新しい計算資源提供の試みが行われている**
  - 9大学・2研究所* mdx（データ活用社会創成基盤）
    - Society 5.0の社会像を受けて，産学官連携のデータ活用を目標
  - 産業技術総合研究所 ABCI（AI橋渡しクラウド）
  - 東京大学 Wisteria/BDEC-01（「計算・データ・学習」融合スパコン）
  - 大阪大学 ONION（データ集約基盤）

* 北海道大学，東北大学，筑波大学，東京大学，東京工業大学，名古屋大学，京都大学，大阪大学，九州大学，国立情報学研究所，産業技術総合研究所

問題：拡大する需給ギャップ

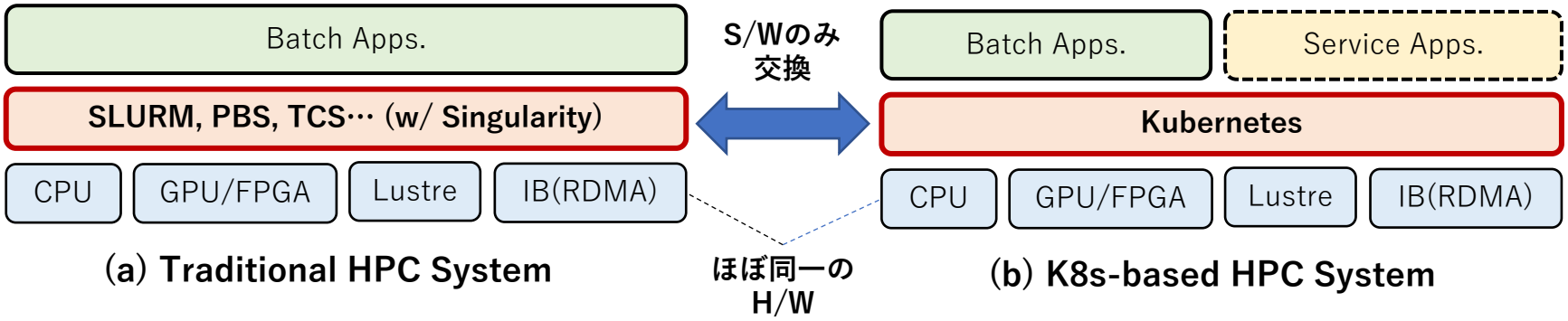
- **研究者の期待は大きくなる一方で，サービス水準は計算機（計算ノード・ストレージ）の提供のまま**
  - 教育プログラム，研究支援サービス（人的リソース）の制約
  - 研究者相互の共同研究による解消も期待されている



本研究の目標

既存研究（HPC183）からはAnsible自動構築化，
K8sを構成するコンポーネントをさらに最適化

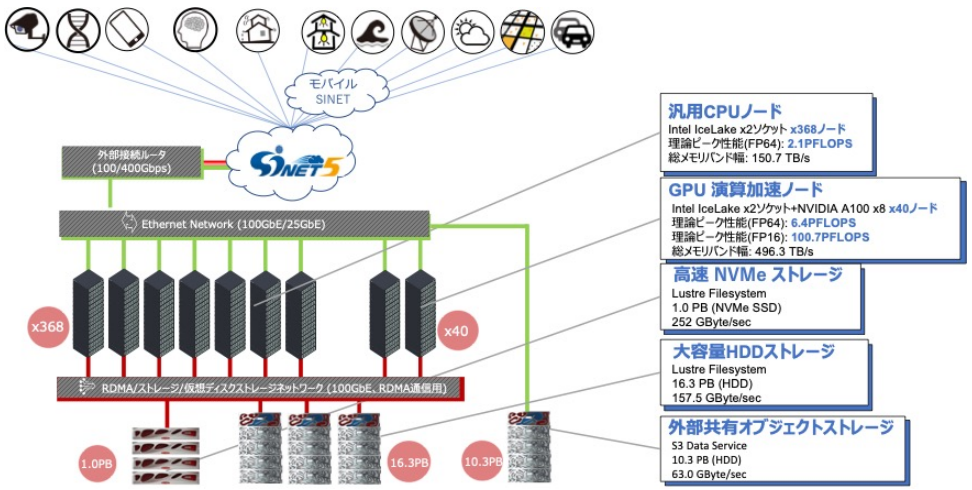
- **従来のスパコンに近い高性能なKubernetes (k8s) 環境のプロトタイプを構築する**
  - **(1) KubernetesベースのHPC実現可能性の検討**
    - 従来のジョブスケジューラ（バッチ処理）によらないHPCシステムの検討
    - 従来のスパコンに近い，高性能な計算ノード・インターコネクト通信・並列分散ファイルシステムを活用
    - 将来的なHPCシステムの可能性を探る
  - **(2) mdxにおけるマネージドKubernetesの提供**
    - より直接的に，利用者にmdxに最適化されたKubernetes環境を提供



最終的には両システムの併存を目指す
k8sの方が今後のアプリケーション整備が進みやすいと期待

mdx（1/2）

- **データ活用社会創成プラットフォーム（mdx）**
  - 内閣府Society 5.0の社会像を受けて，産学官連携によるデータ利活用を目指す
  - mdx計画の第1弾システム
    - 2021.3 稼働，2021.9 運用開始（試験運用期間中）
    - 東京大学柏IIキャンパスに設置
    - サービス提供水準はIaaS（仮想マシン）
      - **研究者相互の共創による環境整備が期待されている**



引用：データ活用社会創成プラットフォーム

計算機（2種類）
(1) 汎用CPUノード（IceLake-SP）
CPUパック（1仮想コア単位）
(2) 演算加速GPUノード（A100）
GPUパック（1GPU単位）
SR-IOV, PVRDMA, Port Group
最大限RDMA可能なのは，SR-IOV

ストレージ（3種類）
(1) 高速ストレージ（/fast, SSD）
(2) 大容量ストレージ（/large, HDD）
DDN Exascalerベース
(3) オブジェクトストレージ
S3互換，外部連携

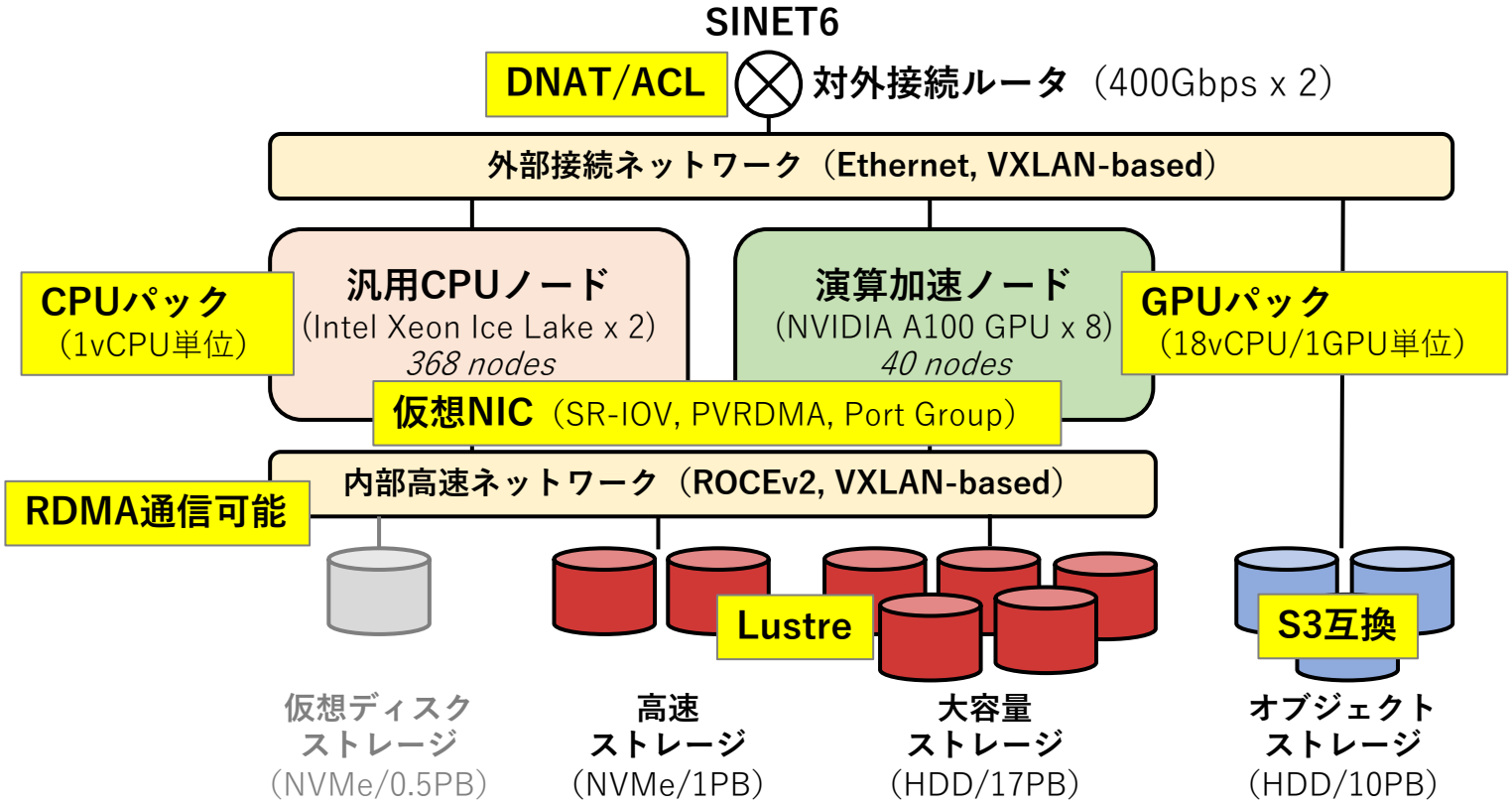
mdx (2/2)

mdxの利用手順

(1) 計算ノードの選択（CPU・GPUパック，仮想NIC（SR-IOV・PVRDMA・PortGroup），仮想マシンテンプレート）

(2) 内部共有ストレージの選択（高速・大容量ストレージ）

(3) 通信の設定（DNAT/ACLの設定）



既存HPCシステムの課題

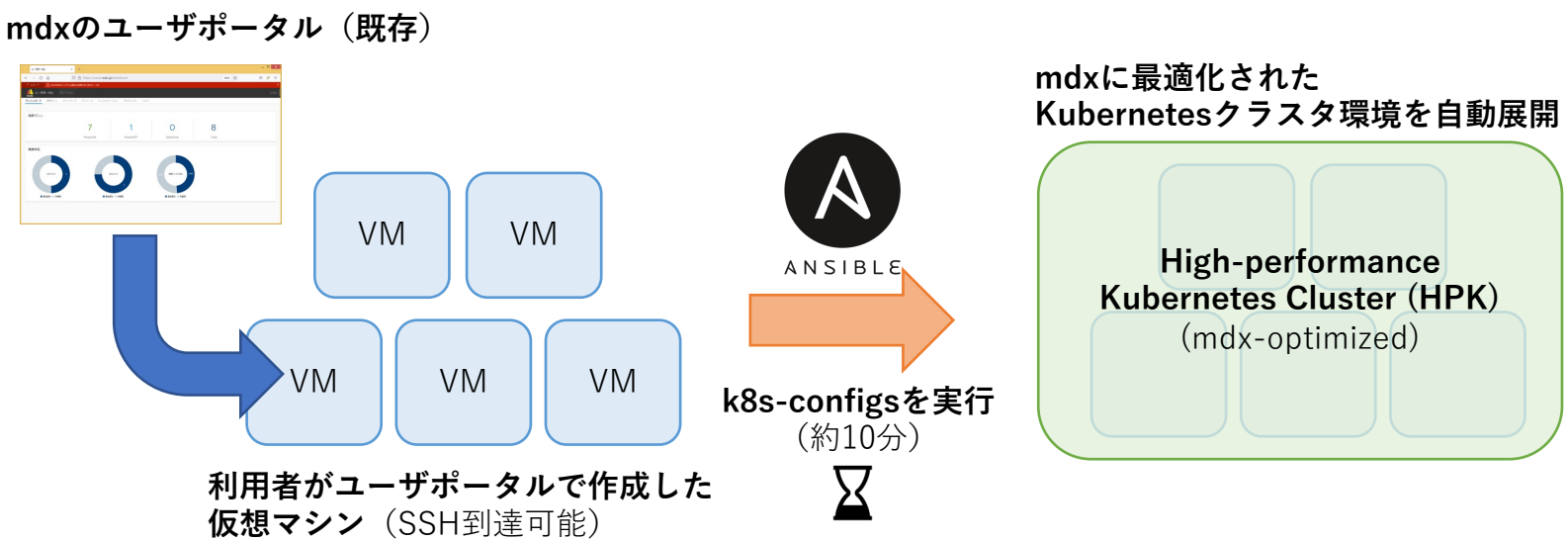
- **対話環境への対応**
  - インタラクティブジョブとJupyter環境の相性がよくない
- **計算機利用率の向上**
  - 近年のGPU（V100/A100/H100）は非常に高い性能
  - 必ずしも全てのジョブでGPUの最大性能は必要ない（MIG: Multi-Instance GPU）
- **計算とI/Oの比率**
  - ETL処理では，CPUはそれほど必要ないが，I/Oを並列分散
- **管理者権限を要するソフトウェア導入**
  - Singularity，Condaなどの導入により改善
- **研究成果のサービス化**
  - バッチ処理では実行まで時間差が発生（プリエンプション・優先処理・資源予約を導入しても）

今回のアプローチ（プロトタイプ構築）

- **既存Kubernetes用コンポーネントを最大限利用**
  - 新規開発は行わない
  - 既存のCRI/CNI/CSIなどを活用・組み合わせる
- **KubernetesにHPCの概念を持ち込む**
  - NUMA（CPU・メモリ）, CPU pinning
  - GPUの活用
  - 並列FS（Lustre）のコンテナ永続ボリュームでの利用
  - RDMA通信に対応（Infiniband or ROCEv2）
- **今回は性能と運用可能性を中心に評価**
  - セキュリティ・プライバシーの評価は次回以降に持ち越し

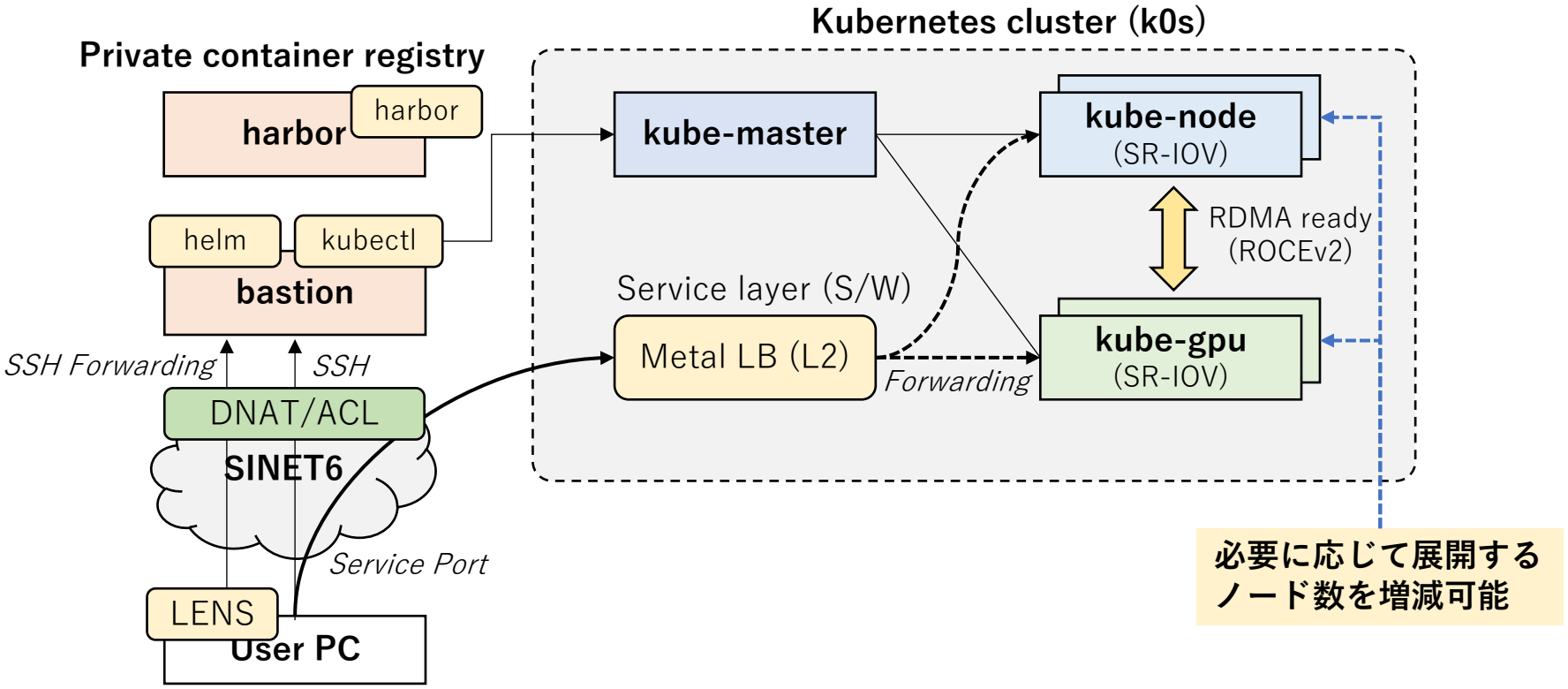
実装

- **k8s-configs**
  - mdxに最適化されたKubernetes環境を自動構築するスクリプト（Ansibleスクリプト）
    - 先行するmachine-configsの影響を受ける
    - ユーザポータルで仮想マシンをセットアップした状態から約10分でKubernetesクラスタを自動展開
      - 仮想マシンのセットアップは今後，mdx REST APIで改善
        - mdx REST APIは別プロジェクト（JST共創）で開発された成果



仮想マシン構成

- **Kubernetesクラスタ**
  - k8sマスタ，k8s計算ノード（CPUノード，GPUノード）
- **周辺サーバ**
  - 踏み台サーバ（bastion），コンテナレジストリ（harbor）



ソフトウェア構成

構成要素	実行ランタイム
Kubernetes distro	k0s（軽量Kubernetes）
NUMA対応	Topology Manager, CPU Manager, Memory Manager
CRI（コンテナ実行環境）	Containerd（k0s標準）
	NVDIA GPU Operator（GPUノードのみ） NVIDIA Container Toolkit，NFD，nv-peermem, MIG対応，※CUDA・GPUドライバはmdx標準を使用
CSI（永続ストレージ対応）	DDN Exascaler File CSI Driver
CNI（ネットワーク対応）	Kube-router（k0s標準）
	Mellanox Network Operator（ROCEv2/RDMA対応） k8s RDMA shared device plugin, Multus, Whereabouts ※Mellanoxドライバはmdx標準を使用
Load Balancer（サービス接続）	Metal LB（L2モード）
運用・監視	LENS, Kube Prometheus Stack
バッチジョブ対応	Kubeflow MPI Operator

※mdx固有に導入した部分

予備実験(1)：自動クラスタ構築時間

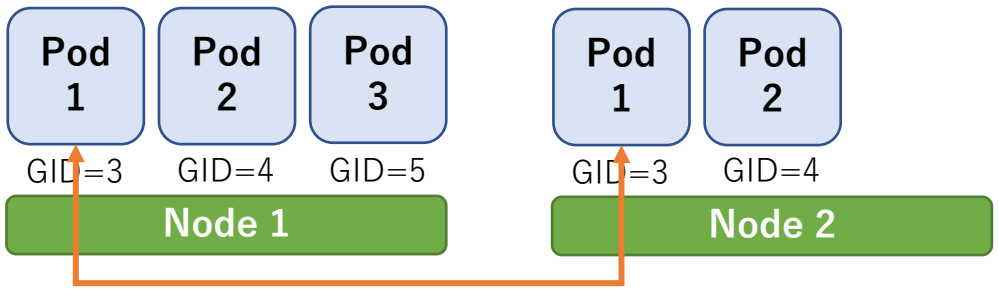
- **最小クラスタの構築時間を3回測定**
  - CPUノード・GPUノード各２台
  - 測定結果：7分55秒，7分52秒，7分55秒で構築完了

ノード	台数	パック数	ディスク	接続
bastion	1台	4 CPUパック	100GB	Port Group
harbor	1台	4 CPUパック	**160GB**	Port Group
kube-master	1台	8 CPUパック	100GB	Port Group
kube-node	**2台**	18 CPUパック	**150GB**	**SR-IOV (RDMA対応)**
kube-gpu	**2台**	**1 GPUパック**	**150GB**	**SR-IOV (RDMA対応)**

予備実験(2)：GPU性能

- **TensorFlow Benchmarksで測定**
  - MPI Operator付属のtf_cnn_benchmarks
    - ResNet50モデル, バッチサイズ64, fp16有効，Horovod使用
    - mdxはCUDA11のため，CUDA10からCUDA11に変更
    - ワーカ・マスタの起動順序を調整（マスタ起動の遅延を挿入）

接続	スループット
Ethernet (frontend)	1933.02 images/sec
ROCEv2 (Socket)	2232.02 images/sec
ROCEv2 (IB)	3346.78 images/sec



K8s RDMA shared device pluginでは異なるPodに異なるInfiniBandのGIDを計算ノードごとに発行
（GIDが合わないと通信できない）

MPI実行パラメータ

- **MPI Operator**
  - mpirun --allow-run-as-root -np 2 -bind-to none -map-by slot -x NCCL_DEBUG=INFO -x NCCL_IB_DISABLE=0 –x NCCL_NET_GDR_LEVEL=1 -x LD_LIBRARY_PATH -x NCCL_IB_GID_INDEX=7 -x PATH -mca pml ob1 –mca btl ^openib python3 tf_cnn_benchmarks/tf_cnn_benchmarks.py --model=resnet50 --batch_size=64 --use_fp16=true --variable_update=horovod

※MPI Operatorサンプルからの変更箇所（暫定的な解決策，ノード間で異なるGIDを発行する場合には対応できない）

アプリケーション対応（現在進行中）

- **HPCサービス**
  - Kubeflowの一部のMPI Operatorを導入（バッチ対応）
  - k8s標準の1.21以降の強化されたバッチ機能の検証
- **ストリーミング処理サービス**
  - Spark on k8s Operator
    - Spark RAPIDS（GPU対応）とUCX Shuffle Manager（IB対応）
  - Strimzi Kafka
- **フロントエンドサービス**
  - JupyterHubまたはBinderHub
    - 認証以外の動作は確認済み（認証方式は検討中）
- **DBサービス**
  - MySQL Operatorの導入（動作は確認済み）

まとめ

- **データ利活用に向けてmdxに最適化された高性能Kubernetes環境のプロトタイプを作成した**
  - 従来のスパコンと同等のハードウェア，ソフトウェアスタックをKubernetesに置き換えたシステムの検証
    - 現状，アプリケーションの整備・本格的な評価は未達成
    - 当初はOCI，Azure HPCなどのパブリッククラウドでの実験を検討していたが，あまり費用を気にせず検証できた
  - mdxにおけるマネージドKubernetesの提供
    - 僅かな設定項目で約10分でKubernetes環境を自動構築

k8s-configs (GitHub):
https://github.com/a-sugiki/k8s-configs

謝辞

- 本研究はJSPS科研費20K11837の助成，学際大規模情報基盤共同利用・共同研究拠点および革新的ハイパフォーマンス・コンピューティング・インフラの支援を受けている(課題番号：jh220058)．また，データ活用社会創成プラットフォームmdxを使用して研究を実施した．