Vol.2018-HPC-167 No.13
2018/12/17

コンテナ型仮想化におけるNVMe-oFの性能評価

畑中 智之[1,a)]　建部 修見[2]

概要：NVMe over Fabrics は，リモートの計算機ノードに接続された NVMe SSD に直接アクセスする技術である．計算機ノードのローカルストレージとして採用された NVMe SSD を活用する，バーストバッファを実現するための技術としての利用が考えられている．コンテナ型仮想化は，アプリケーションとともにライブラリやコンパイラなどの依存関係をコンテナにまとめて実行する技術である．コンパイラやライブラリなどをプログラムの開発環境と HPC 上で統一することで再現性を向上させることができるため，HPC での利用が進むと想定される．そこで，本研究ではコンテナ型仮想化を実現するソフトウェアのひとつである Singularity から，NVMe over Fabrics を用いてリモートの計算機ノードに接続された NVMe SSD にアクセスした場合の性能について評価を行った．そして，Singularity を介してアクセスした場合でも，ネイティブでアクセスした場合と遜色のない性能を得られることを確認した．

1. はじめに

近年，NVMe SSD[1] の普及が進んでいる．それに伴い，NVMe over Fabrics[2]，NVMeDirect[3] といった技術が登場している．NVMe over Fabrics は，リモートの計算機ノードに接続された NVMe SSD に直接アクセスする技術である．NVMeDirect は，HDD に最適化された旧来の Kernel IO で目立つようになったボトルネックを解消するため，ユーザレベルから直接 NVMe SSD を操作する技術である．また，NVMe SSD の普及に伴い，計算機ノードのローカルストレージに NVMe SSD が採用される例が増加している．計算機ノードのローカルストレージとして採用された NVMe SSD の活用方法に，バーストバッファとしての利用が考えられている．既存のバーストバッファの実装に，GPFS[4]，CXFS[5] が存在する．

コンテナ型仮想化は，アプリケーションとともにライブラリやコンパイラなどの依存関係をコンテナにまとめて実行する技術である．コンテナ型仮想化を実現するソフトウェアには，エンタープライズ向けのデファクトスタンダードである Docker[6] のほか，共有計算機環境向けの Singularity[7][8]，Shifter[9][10]，CharlieCloud[11] などが存在する．コンテナ型仮想化を利用することで，プログラムの開発環境と実行環境においてパッケージやコンパイラのバージョンを統一することが容易となることから，HPC 分野においてもコンテナ型仮想化の導入が進むと考えられる．

本研究における最終的な目的は，計算機ノードのローカルストレージに採用された NVMe SSD を活用し，バーストバッファとして利用できるノード間ファイル共有システムを設計することである．このシステムは，コンテナ型仮想化の利用が進むと考えられることを鑑み，コンテナ型仮想化ソフトウェアと親和性の高い設計とすることを目標としている．そこで，本研究ではシステムを設計する上での留意点を明らかにする目的で，コンテナ型仮想化ソフトウェアのひとつである Singularity から，NVMe over Fabrics を用いてリモートの計算機ノードに接続された NVMe SSD にアクセスした場合の性能について評価を行った．第 2 節で関連技術について，第 3 節で NVMe over Fabrics を用いたファイル共有について述べる．第 4 節で NVMe over Fabrics の評価を示し，第 5 節で研究のまとめを述べる．

2. 関連技術

2.1 NVMe over Fabrics

NVMe over Fabrics[2] は，リモートの計算機ノードに接続された NVMe SSD に直接アクセスする技術である．概略図を図 1 に示す．図 1 は，NVMe over Fabrics を用いることで，リモートの計算機ノードからローカルの計算機ノードに接続された NVMe SSD に接続できることを示している．NVMe over Fabrics により，リモートの計算機ノードにおいても当該 NVMe SSD をブロックデバイスとして扱うことが可能となる．

[1] 筑波大学情報学群情報科学類
[2] 筑波大学計算科学研究センター
[a)] hatanaka@hpcs.cs.tsukuba.ac.jp

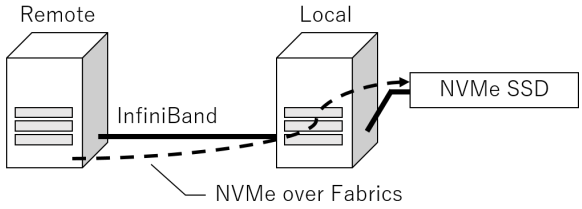

図 1　NVMe over Fabrics の概略図
NVMe-oF Configuration[12] を参考に作成

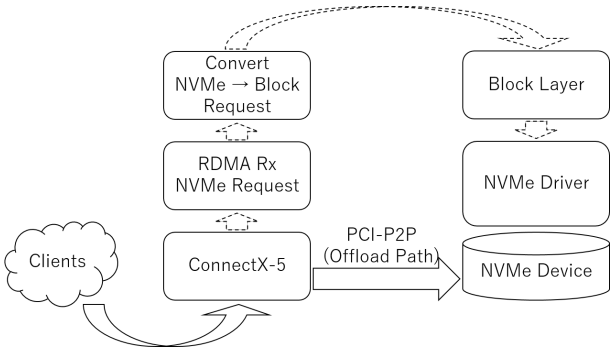

図 2　NVMe over Fabrics Target Offload の概略図
Simple NVMe-oF Target Offload Benchmark[14] を参考に作成

2.2 NVMe over Fabrics Target Offload

NVMe over Fabrics Target Offload[13][14] は，NVMe over Fabrics においてローカルの計算機ノードの CPU 負荷を削減する技術で，Mellanox Technologies より提供されている．概略図を図 2 に示す．リモートの計算機ノードからのリクエストを，ローカルの計算機ノードにおいて ConnectX-5 から PCI peer-to-peer を用いて NVMe デバイスに送ることで，CPU 負荷の削減を実現している．NVMe over Fabrics Target Offload を使用するには，ローカルの計算機ノードの InfiniBand アダプタに ConnextX-5 以上が必要である．

2.3 Docker

Docker[6] は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，デファクトスタンダードとなっている．Docker コンテナの実行には root 権限が必要であるため，共有計算機環境での実行には向かないという問題がある．

2.4 Singularity

Singularity[7][8] は，共有計算機環境向けのコンテナ型仮想化ソフトウェアのひとつである．Singularity に関連した研究に [15][16] がある．[15] では，産総研 AI クラウド上で Singularity を実行した結果を評価しており，ベアメタルな環境での実行と遜色のない性能が得られたとしている．[16] では，BIDS 形式 [17] のニューロイメージングデータを分析するコンテナが提案されている．

3. NVMe over Fabrics を用いたファイル共有

NVMe over Fabrics により，複数の計算機ノードから同一の NVMe SSD に接続することが可能となる．これを活用して，複数の計算機ノードでファイル共有を行うことを考える．

まず，既存のファイルシステムでフォーマットされた NVMe SSD を，ローカルの計算機ノードとリモートの計算機ノードの双方でマウントすることを考える．複数の計算機ノードで同時にマウントすることを想定していないファイルシステムでは，複数の計算機ノードから変更が行われることでファイルシステム自体が壊れる可能性がある．このため，ローカルの計算機ノードでは読み書き可能な状態で，リモートの計算機ノードでは読み取り専用でマウントする必要があるが，これについて，ext4 でフォーマットした NVMe SSD で予備実験を行った結果，ローカルの計算機ノードでの変更がリモートの計算機ノードに伝わらないことを確認している．

次に，SAN ファイルシステムを使用することを考える．SAN ファイルシステムには，GPFS[4]，CXFS[5]，GFS などが存在する．NVMe over Fabrics で SAN を構築し，SAN 上にファイルシステムを構築することで，複数の計算機ノードでファイル共有を行うことが可能となると考えられる．

最後に，NVMe SSD をブロックデバイスとして扱うことを考える．NVMe over Fabrics により，複数の計算機ノードから同一の NVMe SSD をブロックデバイスとして扱うことができるため，ファイルの表現形式を決め，書き込み制御を適切に行うことで，ファイル共有を行うことが可能となると考えられる．

以上の議論により，複数の計算機ノードで NVMe over Fabrics を用いてファイル共有を行うには，SAN ファイルシステムを使用する方法と，ファイルシステムを使用せずにブロックデバイスとして扱う方法が考えられる．

4. 評価

4.1 評価手法

評価は，ディスク IO を測定するベンチマークツールである fio[18] を用いて行った．ローカルの計算機ノード上とリモートの計算機ノード上のそれぞれにおいて，ネイティブで fio を実行した場合と Singularity を介して fio を実行した場合の結果を比較した．

4.1.1 評価項目

評価項目は，バンド幅，IOPS，レイテンシとした．

バンド幅を評価するため，Sequential Read，Sequential Write それぞれにおいて，a) ジョブ数を 4 に固定し，ブロックサイズを 4KiB から 16MiB に変化させて; b) ブロックサ

```
Bootstrap: docker
From: centos:6
%post
    yum install -y epel-release
    yum install -y fio
    mkdir /work
%runscript
    cd /work && fio "$@"
```

図 3　Singularity の Recipi ファイル

イズを 4MiB に固定し，ジョブ数を 1 から 8 に変化させて fio を実行した．

IOPS とレイテンシを評価するため，Random Read，Random Write それぞれについて，ブロックサイズを 4KiB に固定し，ジョブ数を 1 から 8 に変化させて fio を実行した．

4.1.2　Singularity コンテナについて

Singularity のコンテナは，図 3 の Recipi ファイルをもとにビルドを行った．/work ディレクトリは，コンテナ内で fio を実行するディレクトリである．Singularity の --bind オプションを用いて，性能を評価する NVMe SSD 内のディレクトリをコンテナ内にバインドするために用意した．

なお，Singularity で実行するコンテナの OS は，ホスト OS と同じ Centos6 とした．また，コンテナ内の fio のバージョンは，ホスト OS と同じ fio-2.0.13 とした．

4.2　評価環境

評価に用いた計算機を表 1 に，ソフトウェアのバージョンを表 2 に示す．NVMe over Fabrics のセットアップは，Mellanox Technologies の提供しているコミュニティの情報 [19] に基づいて行った．

4.3　評価結果

4.3.1　ブロックサイズ対バンド幅について

図 4 は，Sequential Read，Sequential Write において，ジョブ数を 4 に固定し，ブロックサイズを 4KiB から 16MiB に変化させたときの，バンド幅の評価結果である．

Sequential Read について，ローカルとリモートのそれぞれにおいてネイティブで実行したとき，a) ブロックサイズが 4KiB のとき，リモートのバンド幅は，ローカルのバンド幅の-28%; b) ブロックサイズが 256KiB のとき，リモートのバンド幅は，ローカルのバンド幅の-11%; c) ブロックサイズが 1MiB 以上のとき，リモートのバンド幅は，ローカルのバンド幅の ±2% となった．ローカルにおいて，ネイティブと Singularity を介した実行を比較したとき，ブロックサイズによらず，Singularity を介したバンド幅は，ネイティブのバンド幅の-4%から+2% となった．リモートにおいて，ネイティブと Singularity を介した実行を比較したとき，ブロックサイズによらず，Singularity を介したバンド幅は，ネイティブのバンド幅の-1%から+4% となった．

Sequential Write について，ローカルとリモートのそれぞれにおいてネイティブで実行したとき，a) ブロックサイズが 4KiB のとき，リモートのバンド幅は，ローカルのバンド幅の-43%; b) ブロックサイズが 64KiB のとき，リモートのバンド幅は，ローカルのバンド幅の-4%; c) ブロックサイズが 256KiB 以上のとき，リモートのバンド幅は，ローカルのバンド幅の-2%から+1% となった．ローカルにおいて，ネイティブと Singularity を介した実行を比較したとき，a) ブロックサイズが 4KiB のとき，Singularity を介したバンド幅は，ネイティブのバンド幅の+8%; b) ブロックサイズが 16KiB 以上のとき，Singularity を介したバンド幅は，ネイティブのバンド幅の ±1% となった．リモートにおいて，ネイティブと Singularity を介した実行を比較したとき，a) ブロックサイズが 4KiB のとき，Singularity を介したバンド幅は，ネイティブのバンド幅の-4%; b) ブロックサイズが 16KiB 以上のとき，Singularity を介したバンド幅は，ネイティブのバンド幅の ±1% となった．

4.3.2　ジョブ数対バンド幅について

図 5 は，Sequential Read，Sequential Write において，ブロックサイズを 4MiB に固定し，ジョブ数を 1 から 8 に変化させたときの，バンド幅の評価結果である．

Sequential Read について，ローカルとリモートのそれぞれにおいてネイティブで実行したとき，a) ジョブ数が 1 のとき，リモートのバンド幅は，ローカルのバンド幅の-6%; b) ジョブ数が 2 以上のとき，リモートのバンド幅は，ローカルのバンド幅の-3%から+1% となった．ローカルにおいて，ネイティブと Singularity を介した実行を比較したとき，a) ジョブ数が 1 のとき，Singularity を介したバンド幅は，ネイティブのバンド幅の+7%; b) ジョブ数が 2 以上のとき，Singularity を介したバンド幅は，ネイティブのバンド幅の ±1% となった．リモートにおいて，ネイティブと Singularity を介した実行を比較したとき，ジョブ数によらず，Singularity を介したバンド幅は，ネイティブのバンド幅の-3%から 0% となった．

Sequential Write について，ローカルとリモートのそれぞれにおいてネイティブで実行したとき，a) ジョブ数が 1 のとき，リモートのバンド幅は，ローカルのバンド幅の-6%; b) ジョブ数が 2 以上のとき，リモートのバンド幅は，ローカルのバンド幅の-2%から+1% となった．ローカルにおいて，ネイティブと Singularity を介した実行を比較したとき，ジョブ数によらず，Singularity を介したバンド幅は，ネイティブのバンド幅の-2%から+1% となった．リモートにおいて，ネイティブと Singularity を介した実行を比較したとき，a) ジョブ数が 1 のとき，Singularity を介したバンド幅は，ネイティブのバンド幅の+4%; b) ジョブ数が 2

表 1　評価に用いた計算機

	Local	Remote
OS	CentOS release 6.10 (Final)	CentOS release 6.10 (Final)
Linux Kernel	4.9.135	4.9.135
CPU	Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz x2	Intel(R) Xeon(R) CPU E5-2695 v2 @ 2.40GHz x2
Memory	64GB	64GB
InfiniBand	Mellanox Technologies MT27700 Family [ConnectX-4]	Mellanox Technologies MT27500 Family [ConnectX-3]
MLNX_OFED	4.0-2.0.0.1	4.0-2.0.0.1
NVMe SSD	Intel SSD DC P3700 Series	N/A

表 2　ソフトウェアのバージョン

fio	fio-2.0.13
Singularity	2.6.0

以上のとき，Singularity を介したバンド幅は，ネイティブのバンド幅の-1%から+2% となった．

4.3.3　ジョブ数対 IOPS について

図 6 は，Random Read，Random Write において，ブロックサイズを 4KiB に固定し，ジョブ数を 1 から 8 に変化させたときの，IOPS の評価結果である．

Random Read について，ローカルとリモートのそれぞれにおいてネイティブで実行したとき，ジョブ数によらず，リモートの IOPS は，ローカルの IOPS の-20%から-12% となった．ローカルにおいて，ネイティブと Singularity を介した実行を比較したとき，ジョブ数によらず，Singularity を介した IOPS は，ネイティブの IOPS の ±1% となった．リモートにおいて，ネイティブと Singularity を介した実行を比較したとき，ジョブ数によらず，Singularity を介した IOPS は，ネイティブの IOPS の-4%から+3% となった．

Random Write について，ローカルとリモートのそれぞれにおいてネイティブで実行したとき，a) ジョブ数が 1 のとき，リモートの IOPS は，ローカルの IOPS の+12%; b) ジョブ数が 2 以上のとき，リモートの IOPS は，ローカルの IOPS の-45%から-41% となった．ローカルにおいて，ネイティブと Singularity を介した実行を比較したとき，a) ジョブ数が 2 のとき，Singularity を介した IOPS は，ネイティブの IOPS の-23%; b) ジョブ数が 1，4 以上のとき，Singularity を介した IOPS は，ネイティブの IOPS の-1%から+5% となった．リモートにおいて，ネイティブと Singularity を介した実行を比較したとき，a) ジョブ数が 1 のとき，Singularity を介した IOPS は，ネイティブの IOPS の-8%; b) ジョブ数が 2 以上のとき，Singularity を介した IOPS は，ネイティブの IOPS の-3%から+1% となった．

4.3.4　ジョブ数対レイテンシについて

図 7 は，Random Read，Random Write において，ブロックサイズを 4KiB に固定し，ジョブ数を 1 から 8 に変化させたときの，レイテンシの評価結果である．

Random Read について，ローカルとリモートのそれぞれにおいてネイティブで実行したとき，ジョブ数によらず，リモートのレイテンシは，ローカルのレイテンシの+55%から+70% となった．ローカルにおいて，ネイティブと Singularity を介した実行を比較したとき，a) ジョブ数が 4 以下のとき，Singularity を介したレイテンシは，ネイティブのレイテンシの-3%から 0%; b) ジョブ数が 8 のとき，Singularity を介したレイテンシは，ネイティブのレイテンシの+20% となった．リモートにおいて，ネイティブと Singularity を介した実行を比較したとき，ジョブ数によらず，Singularity を介したレイテンシは，ネイティブのレイテンシの 0%から+6% となった．

Random Write について，ローカルとリモートのそれぞれにおいてネイティブで実行したとき，ジョブ数によらず，リモートのレイテンシは，ローカルのレイテンシの+107%から+142% となった．ローカルにおいて，ネイティブと Singularity を介した実行を比較したとき，ジョブ数によらず，Singularity を介したレイテンシは，ネイティブのレイテンシの-8%から+9% となった．リモートにおいて，ネイティブと Singularity を介した実行を比較したとき，ジョブ数によらず，Singularity を介したレイテンシは，ネイティブのレイテンシの 0% となった．

4.4　考察

バンド幅について，Sequential Read，Sequential Write それぞれでブロックサイズとジョブ数を変化させて評価を行った．ネイティブでの実行では，ブロックサイズが小さい場合に，リモートの計算機ノードでのバンド幅がローカルの計算機ノードと比較して小さくなることを確認した．ブロックサイズを大きくすると，リモートの計算機ノードでのバンド幅とローカルの計算機ノードでのバンド幅の差異が確認されなくなったことから，ブロックサイズが小さいときにバンド幅が小さくなることは，NVMe over Fabrics が介することでレイテンシが高くなるためであると考えられる．ジョブ数が小さい場合においても同様の傾向が見られたが，これも NVMe over Fabrics が介することでレイテンシが高くなるためであると考えられる．Singularity を介した実行では，Singualrity が介することでバンド幅が一律に小さくなることはなく，影響はないと考えられる．

IOPS，レイテンシについて，Random Read，Random

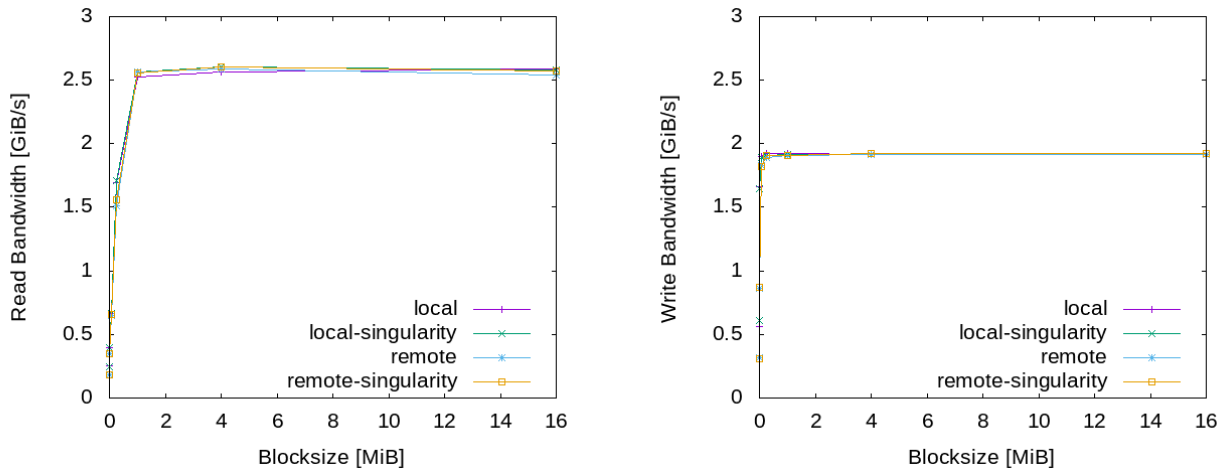

図 4　ブロックサイズを増やしたときのバンド幅
（左）Sequential Read（右）Sequential Write

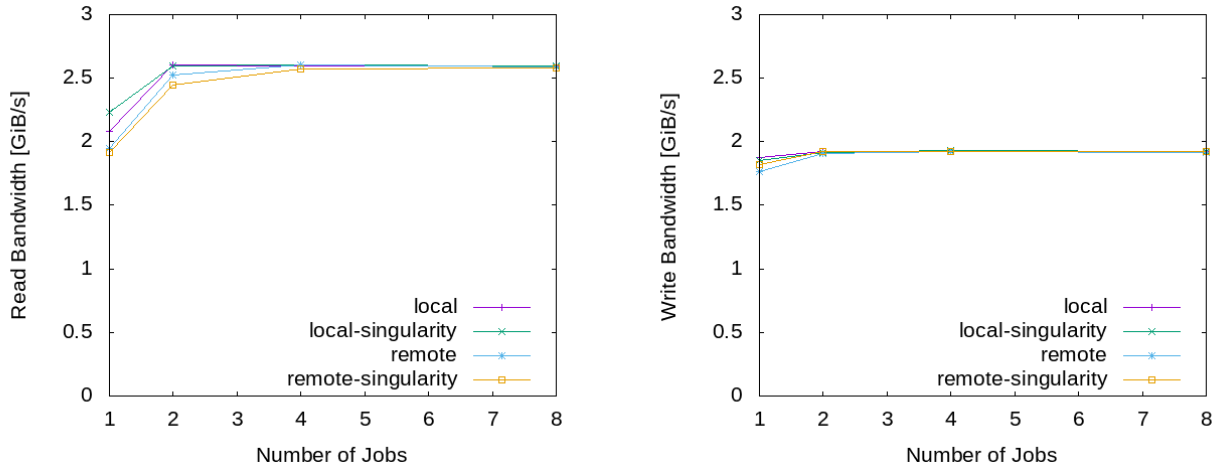

図 5　ジョブ数を増やしたときのバンド幅
（左）Sequential Read（右）Sequential Write

Write それぞれでジョブ数を変化させて評価を行った．ネイティブでの実行では，ジョブ数に関係なく，リモートの計算機ノードの IOPS がローカルの計算機ノードと比較して小さく，リモートの計算機ノードのレイテンシがローカルの計算機ノードと比較して高い結果となった．これは，NVMe over Fabrics を用いた接続であるためレイテンシが高くなり，IOPS が低下したためと考えられる．Singularity を介した実行では，Singularity の有無によって結果が約 20%異なるケースを確認している．しかし，該当するケースが一部であることと，ジョブ数によらないことから，偶発的なものであると考えている．Singularity を介した場合においても，IOPS とレイテンシへの影響はないと考えられる．

5. まとめ

本研究は，ノード間ファイル共有システムを設計する上での留意点を明らかにする目的で，Singularity から NVMe over Fabrics を用いてリモートの計算機ノードに接続された NVMe SSD にアクセスした場合の性能について評価を行った．その結果，a)NVMe over Fabrics を用いてリモートの NVMe SSD にアクセスするとき，ローカルの NVMe SSD にアクセスするときと比較して，レイテンシが高くなること; b) ローカルの計算機ノード上でネイティブと Singualrity を介した実行と，リモートの計算機ノード上でネイティブと Singularity を介した実行のそれぞれで，Singularity の有無による有意な差は認められないことを確認した．これより，ノード間ファイル共有システムを設計する上で，a) リモートの NVMe SSD へのアクセスが少ないほうが望ましいこと; b) コンテナ型仮想化ソフトウェアを対象に性能面での最適化を行う必要がないことを確認した．

今後は，ノード間ファイル共有システムの設計と実装を行う．

謝辞　本研究の一部は，JST CREST JPMJCR1303, JST

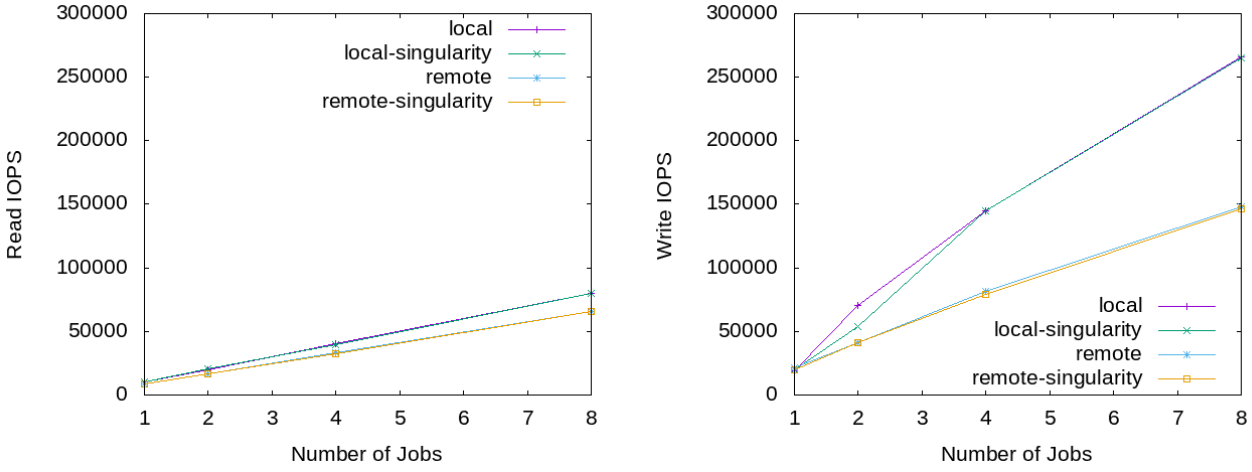

図 6 ジョブ数を増やしたときの IOPS
（左）Random Read（右）Random Write

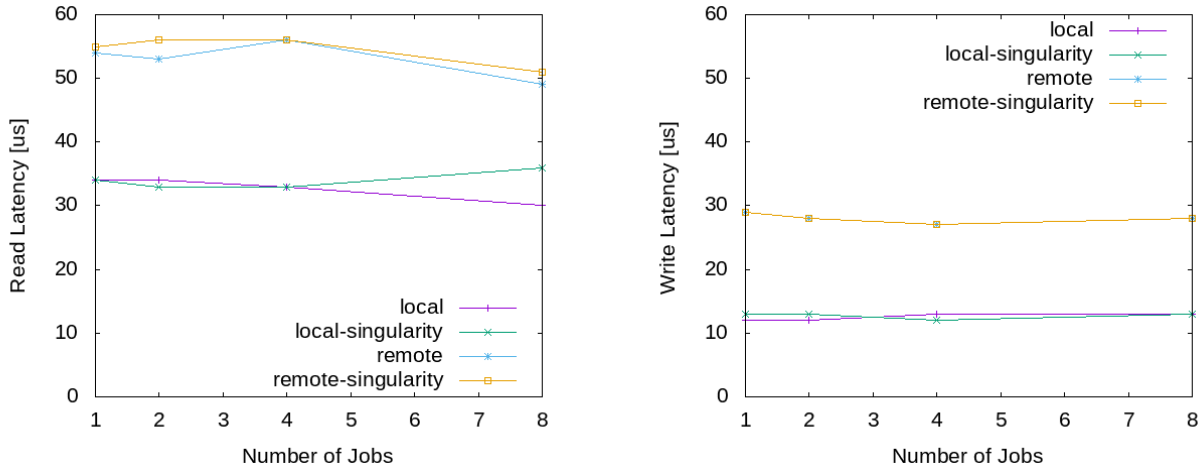

図 7 ジョブ数を増やしたときのレイテンシ
（左）Random Read（右）Random Write

CREST JPMJCR1414，JSPS 科研費 17H01748 および富士通研究所との共同研究の助成を受けたものです．

参考文献

[1] NVM Express: NVM Express scalable, efficient, and industry standard, NVM Express, Inc. (online), available from ⟨https://nvmexpress.org/⟩ (accessed 2018-11-05).
[2] NVM Express: NVMe over Fabrics Overview, NVM Express, Inc. (online), available from ⟨https://nvmexpress.org/wp-content/uploads/NVMe_Over_Fabrics.pdf⟩ (accessed 2018-11-06).
[3] Kim, H.-J., Lee, Y.-S. and Kim, J.-S.: NVMeDirect: A User-space I/O Framework for Application-specific Optimization on NVMe SSDs, *8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16)*, Denver, CO, USENIX Association, (online), available from ⟨https://www.usenix.org/conference/hotstorage16/workshop-program/presentation/kim⟩ (2016).
[4] Schmuck, F. B. and Haskin, R. L.: GPFS: A Shared-Disk File System for Large Computing Clusters., *FAST*, Vol. 2, No. 19 (2002).
[5] Shepard, L. and Eppe, E.: SGI® InfiniteStorage Shared Filesystem CXFS™: A High-Performance, Multi-OS Filesystem from SGI, *White Paper*, Vol. 2691, pp. 08–21 (2003).
[6] Docker: Enterprise Container Platform | Docker, Docker Inc. (online), available from ⟨https://www.docker.com/⟩ (accessed 2018-11-06).
[7] Kurtzer, G. M., Sochat, V. and Bauer, M. W.: Singularity: Scientific containers for mobility of compute, *PloS one*, Vol. 12, No. 5, p. e0177459 (2017).
[8] Sochat, V. V., Prybol, C. J. and Kurtzer, G. M.: Enhancing reproducibility in scientific computing: Metrics and registry for Singularity containers, *PloS one*, Vol. 12, No. 11, p. e0188511 (2017).
[9] Canon, R. S. and Jacobsen, D.: Shifter: containers for HPC, *Proceedings of the Cray User Group* (2016).
[10] Benedicic, L., Cruz, F. A. and Schulthess, T. C.: Shifter: Fast and consistent HPC workflows using containers.
[11] Priedhorsky, R. and Randles, T.: Charliecloud: Unprivileged Containers for User-defined Software Stacks in HPC, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '17, New York, NY, USA, ACM,

pp. 36:1–36:10 (online), DOI: 10.1145/3126908.3126925 (2017).

[12] Mellanox Technologies: NVMe-oF Configuration, Mellanox Technologies (online), available from ⟨https://vimeo.com/222075518⟩ (accessed 2018-11-05).

[13] ophirmaor: HowTo Configure NVMe over Fabrics (NVMe-oF) Target Offload, Mellanox Interconnect Community (online), available from ⟨https://community.mellanox.com/docs/DOC-2918⟩ (accessed 2018-11-10).

[14] ophirmaor: Simple NVMe-oF Target Offload Benchmark, Mellanox Interconnect Community (online), available from ⟨https://community.mellanox.com/docs/DOC-2906⟩ (accessed 2018-11-10).

[15] 佐藤仁，小川宏高：AI クラウドでの Linux コンテナ利用に向けた性能評価，情報処理学会研究報告，Vol. 2017-HPC-162, No. 23 (2017).

[16] Gorgolewski, K. J., Alfaro-Almagro, F., Auer, T., Bellec, P., Capotă, M., Chakravarty, M. M., Churchill, N. W., Cohen, A. L., Craddock, R. C., Devenyi, G. A. et al.: BIDS apps: Improving ease of use, accessibility, and reproducibility of neuroimaging data analysis methods, *PLoS computational biology*, Vol. 13, No. 3, p. e1005209 (2017).

[17] Gorgolewski, K. J., Auer, T., Calhoun, V. D., Craddock, R. C., Das, S., Duff, E. P., Flandin, G., Ghosh, S. S., Glatard, T., Halchenko, Y. O. et al.: The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments, *Scientific Data*, Vol. 3, p. 160044 (2016).

[18] Axboe, J.: Flexible I/O Tester, N/A (online), available from ⟨https://github.com/axboe/fio⟩ (accessed 2018-11-07).

[19] ophirmaor: HowTo Configure NVMe over Fabrics, Mellanox Interconnect Community (online), available from ⟨https://community.mellanox.com/docs/DOC-2504⟩ (accessed 2018-11-05).