Vol.2018-MPS-118 No.47
Vol.2018-BIO-54 No.47
2018/6/15

生命情報解析分野における コンテナ型仮想化技術の動向と性能検証

青山 健人[1,2,a)]　大上 雅史[1]　秋山 泰[1]

概要：軽量かつ高速なコンテナ型仮想化はアプリケーションの可搬性を担保し，研究の再現性を向上させるとして生命情報解析の研究でも普及しはじめている．本報告では，特に HPC 分野で主要なコンテナ型仮想環境として Docker, Shifter, Singularity を取り上げ，生命情報科学の研究ワークフローへの導入の実例を示す．また，各コンテナ型仮想環境でのタンパク質間相互作用予測ソフトウェア（MEGADOCK）の実行性能の検証を通して，計算律速な生命情報科学アプリケーションに適したコンテナ型仮想環境の選択を模索する．

キーワード：コンテナ型仮想化, Docker, Singularity, Shifter, MPI, GPU, バイオインフォマティクス, MEGADOCK

Survey and Performance Comparison of Container Technologies in Bioinformatics Field

Kento Aoyama[1,2,a)]　Masahito Ohue[1]　Yutaka Akiyama[1]

Abstract: Container Virtualization, the lightweight and fast virtualization technology, has been spreading in the research field of bioinformatics to improve the portability of application and reproducibility of research. In this study, we focused on the container virtualizations (Docker, Singularity, and Shifter) used especially in the HPC and bioinformatics field. We performed a survey of use-cases of container virtualization in bioinformatics research works and tried to seek a best-practice through the performance evaluation of MEGADOCK, the compute-intensive application of protein-protein interaction prediction software.

Keywords: Container Virtualization, Docker, Singularity, Shifter, MPI, GPU, Bioinformatics, MEGADOCK

1. 導入

計算機性能の向上により我々研究者が利用可能な計算機環境の選択肢は多様化し，現代の研究では研究用アプリケーションは個人用ノートブックから研究機関の持つ並列計算環境や商用クラウド環境など様々な環境における動作が要求されるようになり，様々な計算機環境におけるアプリケーションの可搬性の需要が高まっている．特に生命情報解析においてはデータの形式に応じて様々なアプリケーションの利用が要求されることが多く，必要なライブラリやツール等の依存関係が複雑になりやすいため，可搬性は研究の再現性に繋がる重大な要素の 1 つである [1]．

一方，近年の計算科学の基盤領域においては，依存する環境ごとアプリケーションを仮想化するコンテナ型仮想化技術が導入されはじめている．コンテナ型仮想化は，アプリケーションに必要なライブラリ等の依存関係をホストのファイルシステムから隔離し，実行することができる．隔

[1] 東京工業大学 情報理工学院 情報工学系
Department of Computer Science, School of Computing, Tokyo Institute of Technology
[2] 産業技術総合研究所 産総研・東工大 実社会ビッグデータ活用オープンイノベーションラボラトリ
Real World Big-Data Computation Open Innovation Laboratory (RWBC-OIL), National Institute of Advanced Industrial Science and Technology (AIST)
[a)] aoyama@bi.c.titech.ac.jp

離された環境は他のシステム上でも実行可能な仮想イメージとして共有可能であり，仮想マシン等と比較してイメージのサイズが軽量，実行性能への悪影響が少ないといた特性を持つことから，利用者の利便性向上のためのソフトウェア基盤として徐々に取り入れられている [2].

コンテナ型仮想化は生命情報解析と関連のある計算機センターやクラスタに徐々に導入されはじめており，2018年に稼働予定の産業技術総合研究所の AI 橋渡しクラウド（AI Bridging Cloud Infrastructure, ABCI）では，コンテナ型仮想環境として，スケジューラに統合された Docker[3] および Singularity[4] の導入が予告されている．また，スペインの Centre for Genomic Regulation においても計算機クラスタに Docker 環境が提供されており [5]，国内では東京大学医科学研究所ヒトゲノム解析センターのスーパーコンピュータ SHIROKANE で Singularity が導入されたほか，理化学研究所のバイオインフォマティクス研究開発ユニットではローカルクラスタと商用クラウド環境上の両方でコンテナ型仮想化を利用した解析ワークフローが現場に取り入れられている [6].

以上のように，生命情報解析分野においてもコンテナ型仮想化を利用した事例が増加している．一方で，HPC 環境で導入されるコンテナ型仮想環境には主に Docker, Singularity, Shifter が挙げられるが，計算機センターの方針によってユーザーが利用可能な環境が異なることもあって，生命情報科学の研究者が何のコンテナ環境から手を付けるべきかが明らかではないのが現状である．

本報告では，まず現在の計算基盤に導入されることが多い主要なコンテナ型仮想環境を紹介し，生命情報科学や計算基盤における利用例を例示する．次に，各コンテナ型仮想環境において生命情報科学の実アプリケーションを利用して実行性能の比較を行い，生命情報科学における計算機基盤のユーザーと管理者の視点から本報告を総括する．

2. コンテナ型仮想化

2.1 コンテナ型仮想化の技術概要

一般的に広く使われる仮想マシンとコンテナのそれぞれを対象とした仮想化の概要を図 1 に示す．仮想マシンはハードウェア，オペレーティングシステム（Linux kernel）を含めて様々な仮想化を行うため，実行性能への影響が大きくなりやすい．一方でコンテナ型仮想化では，コンテナを実行するプロセスに見せる名前空間（プロセス空間，ネットワーク，ファイルシステム等）を制限することでコンテナ内のプロセスに対して仮想環境を提供するため，イメージサイズが小さく済み，また仮想マシンに比べて実行性能の低下が小さい利点がある [2].

コンテナの仮想イメージは大抵は異なる環境の動作を想定しており，最も普及する Docker[3] の場合にはビルド済みの仮想イメージを登録する DockerHub[7] にデータを

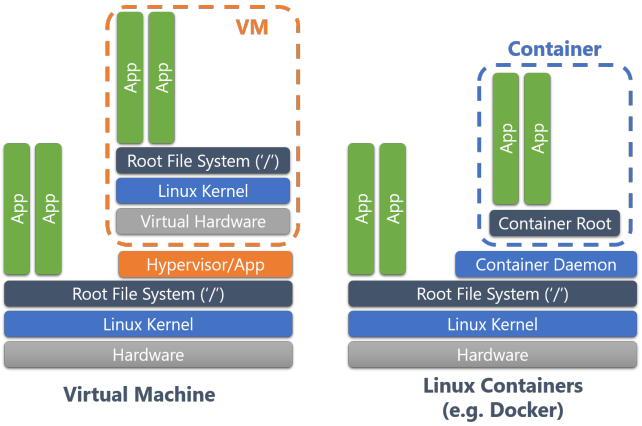

図 1 仮想化の概要
Fig. 1 Virtualization Overview

アップロード（Push）しておくことで，商用クラウド環境や個人用ノートブックにダウンロード（Pull）したあと，即座に実行できる．これは仮想イメージの持つファイルシステムがアプリケーションに必要な依存ライブラリ等をすべて保持しているためであり，この特性がアプリケーションの可搬性の向上に大きく貢献している．

2.2 コンテナ型仮想環境を提供するソフトウェア例

本節では現在利用される主要なコンテナ型仮想環境を提供するソフトウェアを例示する（表 1）.

Docker

Docker[3] は現在最も普及しているコンテナ型仮想環境である．ハイパーバイザが仮想マシンを管理していたように，Docker では管理者権限で動作するデーモンが各コンテナのシステムを管理する．デーモンはコンテナイメージ管理だけでなく，ネットワーク機能やシステムリソースの制限など多数の機能を有しており，一般ユーザーからは機能的にはほぼ仮想マシンのように扱うことが可能である．

また，Docker の最大の特徴は仮想イメージ，つまりコンテナ内のファイルシステムを差分管理できることにある [10]．例えば，ある仮想イメージにタグ付けされたアプリケーションを更新して新しい仮想イメージを作成すると，更新前の仮想イメージからの差分だけを新しくファイルシステムに作成する．これにより，ユーザはアプリケーションの過去の状態にいつでも戻ることができる．

Docker は作成したコンテナの仮想イメージを共有するレジストリサービス DockerHub[7] を提供しており，Ubuntu, CentOS などの公式 OS ベースイメージの配布から，個人や研究者らが作成した特定のアプリケーション（BLAST, BWA, etc.）の共有など様々な仮想イメージを利用できる．仮想イメージをプライベートに設定したり，ローカル環境にプライベートなレジストリサービスを構築することも可能であり，扱いに注意が必要なデータやアプリケーション

表 1　コンテナ仮想化を提供するソフトウェア例
Table 1　Examples of container technologies

	Docker[3]	Singularity[4]	Shifter[8]
仮想化の実現方法	デーモン (root) による管理	SUID/ユーザー名前空間	SUID
仮想イメージの共有方法	DockerHub	SingularityHub	ImageGateway (local)
仮想イメージの差分管理	レイヤー管理	N/A	N/A
Docker イメージとの互換性	-	○	○
GPU 対応	○ [9]	○	○
MPI 対応	○	○	○
実行性能の影響	小（設定に依存）	微小	微小
導入難易度	易しい（e.g. `apt`, `yum`, etc.）	易しい	難しい

を含む場合にも利用できる．

ここに挙げる機能はあくまで代表的な機能の例であり，Docker はさらに複数ノード間の実行やサービスの負荷分散に応じた機能，またオープンに制定されたコンテナ規格により保障された互換性 [11] など多くの標準化された機能を持つ，コンテナ型仮想環境の標準実装となりつつある．一方で高機能であるためコンテナの実行には実質的な管理者権限が要求されるなどの課題があり，HPC 基盤での運用では公認イメージのみの実行を許可するか，または機能が制限されるが管理者権限が不要なコンテナ仮想環境が利用されることが多い．

Singularity

Singularity[4] は，ローレンスバークレー国立研究所（LBNL）で開発された HPC 環境向けのコンテナ仮想環境である．Docker における管理者権限で動作するデーモンに対するセキュリティ上の懸念を回避し，ネットワーク機能やシステムリソース制限などの機能による実行性能への影響を取り除くために，複数の機能を取り除いて HPC 環境に特化したものである．Singularity では実行時のみ管理者権限で動作する SUID フラグのみが必要であり，常駐型のデーモンは不要となる．また，特別な権限を持たない一般ユーザーでも（一部機能を除く）コンテナの実行と操作が可能であり，多数の一般ユーザーによる利用が想定されている．HPC 向けの MPI ベンチマーク等でも実システム環境と遜色のない実行性能を示しており [12]，Docker と比較してシンプルな実装と HPC 向けの周辺ツールセットの充実性から動作が安定しているため，複数の研究機関の HPC 基盤で採用が進んでいる．

Shifter

Shifter[8] は，ローレンスバークレー国立研究所（LBNL）の国立エネルギー研究科学計算センター（NERSC）にて開発された HPC 基盤向けコンテナ仮想環境である．コンテナ型仮想環境の実現方法は Singularity と近いがさらにシンプルな実装であり，プロセスに見せるファイルシステムだけを隔離して（`chroot` による）コンテナ環境を実現している．Singularity と同じく，システムからは通常のプロセスと同じように見えるため，既存のスケジューラーによるプロセス管理が容易に行える．欧州にある Swiss National Supercomputing Centre（CSCS）にて開発が継続されており [13]，同センターが運用するスーパーコンピュータ PizDaint など，複数の研究機関の HPC 基盤で利用可能である．

3. 生命情報科学等におけるコンテナ型仮想化の採用例

本節では生命情報科学の研究へのコンテナ型仮想化の導入例を示す．

3.1 Centre for Genomic Regulation（CGR）の Docker の運用

スペインのバルセロナに所在する Centre for Genomic Regulation (CGR) では，同機関が提供する HPC システムで Docker 環境を提供している [5]．Docker 環境は HPC システム上の Univa Grid Engine のスケジューリングに統合されており，計算ノード上でジョブを実行する際に現行のシステム環境だけでなく，ユーザーが指定した仮想イメージ環境（指定バージョンのゲノム解析パイプライン，過去のシステム環境等）を利用してジョブを実行できる．これは細かなアプリケーションの設定コストが抑えられるだけでなく，ライブラリのバージョンの違い等によって解析結果に違いが生じた場合にも，過去のシステム環境を利用して同一のソフトウェア環境を再現可能とすることにより，研究成果の再現性を向上に貢献している．同機関の開発するゲノム解析パイプライン Nextflow[14] も Docker によるコンテナ仮想環境で公開および配布されており，早期からコンテナ型仮想化を生命情報科学の研究ワークフローに取り入れた例と言える．

3.2 理化学研究所 生命機能科学研究センター バイオインフォマティクス研究開発ユニットの Docker 運用と BioDevOps

国内では理化学研究所のバイオインフォマティクス研究開発ユニットがいち早く Docker を導入，運用している [6]．

こちらも同じく Univa Grid Engine 統合の Docker を利用しており，計算ノード上でコンテナ化された様々な解析ツールが利用できる．同ユニットでは研究室の PC クラスタから商用クラウド環境にジョブを投入するワークフローも存在するが，DockerHub を介してコンテナの仮想イメージをダウンロードすることで，システム環境が異なる商用クラウドなどでも解析アプリケーションを実行できる利点がある．同ユニットはバイオインフォマティクス解析，IT インフラ，アプリケーション開発を一体化した BioDevOps[15] を提言しており，コンテナ型仮想化や仮想マシンの運用を含めた再現性のあるバイオインフォマティクス環境の構築を推進している．

3.3 産業技術総合研究所 ABCI の Singularity 運用

産業技術総合研究所が 2018 年に稼働を予定している AI 橋渡しクラウド（AI Bridging Cloud Infrastructure, ABCI）では，Singularity を核としたコンテナ仮想環境の提供を予定している [16]．特に近年の人工知能研究で興隆する深層学習では大規模な GPU を利用した計算機環境が必要とされているが，ABCI は HPC 分野とクラウド分野の技術領域の橋渡しを狙うものであるため，実システム環境にも最適化された深層学習向けのコンテナ仮想イメージを DockerHub で公開しており，一般に利用可能である．ユーザーは計算ノード上では Singularity を利用したコンテナ仮想環境でユーザーが用意したアプリケーション，コードを実行可能であり，既存のアプリケーションをスーパーコンピュータ上で容易に実行可能とすることが期待されている．既に代表的なユースケースに関してはベンチマーク測定等を実施済みであり，実システム環境に近い実行性能を示すことを確認している [16]．

4. 評価実験

本実験では，前章までに挙げたコンテナ型仮想環境を提供する複数のソフトウェアを用いた際の実行時間について，実際の生命情報科学のアプリケーションを対象に評価実験を行い，コンテナ仮想化による実行性能への影響を調査する．

4.1 対象アプリケーション

Ohue らにより開発された大規模並列計算環境を想定したタンパク質間相互作用予測ソフトウェア MEGADOCK[17] を用いる．MEGADOCK は OpenMP，MPI，GPU による高速化に対応する計算律速のアプリケーションであり，東京工業大学のスーパーコンピュータ「TSUBAME 2.5/3.0」，理化学研究所のスーパーコンピュータ「京」などの大規模並列環境上の実績を持つ．また，Microsoft Azure 上の HPC 向け仮想マシンにおける大規模並列計算の実績 [18], [19] がある．

4.2 実験条件

測定条件，並列化設定等

MEGADOCK のアプリケーションを実システム環境（Native）と仮想コンテナ環境（Docker，Singularity，Shifter）で用意し，ドッキング計算の実行時間を比較する．その際，MEGADOCK の MPI/OpenMP により並列化された CPU 版と，CUDA により GPU 並列化された GPU 版のそれぞれについて実行時間を計測する．実行時間の計測は `time` コマンドを用いて各ケースに対して 5 回ずつ行い，全体の中央値を採用する．CPU 版の実行では，MPI は 1 ノードあたり 4 プロセス，1 プロセスあたり 10 スレッドとして設定して計測を行った．

仮想イメージの作成

仮想コンテナ環境で利用する仮想イメージは Docker で作成した仮想イメージ（Docker イメージ）をベースとする．その際，仮想イメージは可能な限り実システム環境（Native）に近い構成になるように留意する．Singularity/Shifter については，Docker イメージを仮想イメージの共有サービス（DockerHub）にアップロードしたあとそれぞれの持つ互換機能（`singularity pull`，`shifter pull`）を用いて対応する形式に変換したものを利用する（**図 2**）．

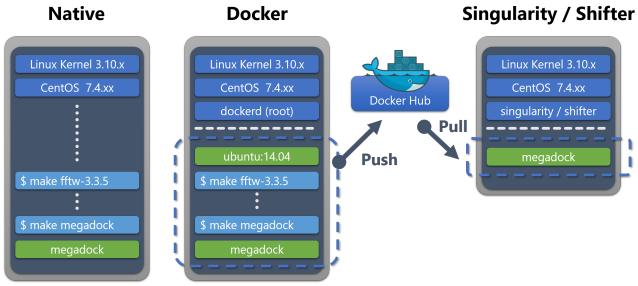

図 2 仮想イメージの構築の流れ
Fig. 2 Workflow of building container images

データセット

Protein-Protein Docking Benchmark version 1.0 [20] の複合体構造データセットを用いた．データセットには二量体の複合体構造が含まれており，二量体の各構成タンパク質がレセプターとリガンドとして区別されている．本研究では，レセプターとリガンドのすべての組み合わせからランダムに 100 ペアの組み合わせを選択してドッキング計算を行う．

4.3 計算機環境

ハードウェア仕様

実験環境のハードウェア仕様について，**表 2** に示す．本報告では，ローカル環境のマシン 1 台ですべての実験を実施する．

表 2　計算機環境
Table 2　Hardware specification

CPU	Intel Xeon Gold 6130 CPU 2.10 [GHz], 16 cores
Memory	512 [GB]
GPU	NVIDIA Tesla V100 × 3 cards

ソフトウェア環境

実験環境上のソフトウェア環境，および仮想イメージの環境について，**表 3** に示す．

4.4　実験結果

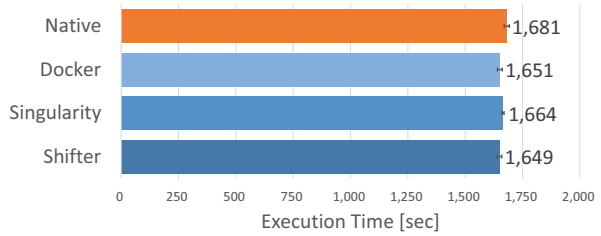

図 3　CPU 版 MEGADOCK によるドッキング計算（100 ペア）の実行時間
Fig. 3　Execution time of 100 pairs of docking calculation by MEGADOCK (CPU)

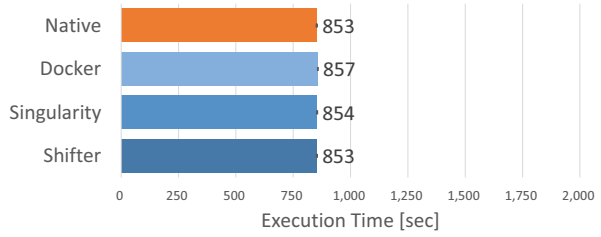

図 4　GPU 版 MEGADOCK によるドッキング計算（100 ペア）の実行時間
Fig. 4　Execution time of 100 pairs of docking calculation by MEGADOCK (GPU)

図 3，**図 4** に示された実験結果より．システム環境上の実行時間とコンテナ環境上の実行時間の両方において大きな差は見られなかった．個別に実行時間を見ると，CPU 版の場合は Native 環境に対する各環境の実行時間の差は 2%以内におさまった．Docker，Singularity の環境ではノード間通信や CPU ソケット間の通信が発生する場合に若干性能が低下する（数%程度）ことが複数の先行研究 [2], [12] で報告されているが，今回の実験は単一ノード上かつソケット内通信のみの環境で行われており，大きな差には繋がらなかったと考えられる．

GPU 版の場合には，CPU 版よりもさらに差が小さく，Native 環境に対する各環境の実行時間の差は 1%未満におさまった．GPU の利用による全体実行時間の短縮に応じて各環境における実行時間の差も小さくなったと考えられる．

一方，実験環境は NVIDIA Tesla V100 の GPU カードが 3 枚搭載された GPU 偏重のマシン構成であったが，CPU 版に対する GPU 版の速度向上は全体的に約 1.95 倍に留まった．今回の実験では 100 ペアの PDB ファイルのそれぞれ 1 ペアに対して 3 枚の GPU カードを用いてドッキング計算を実施したが，データセット内で最も大きなサイズの PDB ファイルのペアのドッキング計算（**図 5**）において GPU による処理にかかる時間（docking）は全体の約 47.8%であり，GPU による実装がなされていない部分の実行時間が支配的になっていた．ドッキングの前処理を含めて GPU で処理を行う，または 1GPU による処理を複数実行するデータ並列化を採用することで全体の実行効率を向上させられると考えられる．

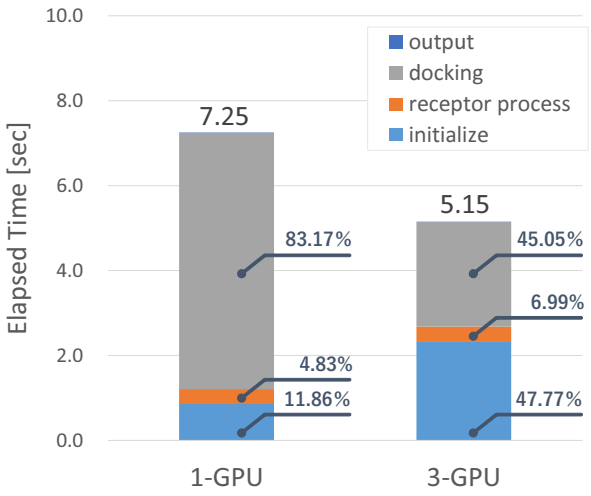

図 5　GPU 版 MEGADOCK 実行時間のプロファイル例
Fig. 5　Example of runtime profile of docking calculation by MEGADOCK (GPU)

5.　結論

本報告では，コンテナ型仮想化の主要なソフトウェアについて概要を示し，生命情報科学におけるコンテナ型仮想化技術の動向を探った．また，評価実験では生命情報科学の実アプリケーションとしてドッキング計算によるタンパク質間相互作用予測ソフトウェア MEGADOCK を対象に主要なコンテナ型仮想環境で単一ノード上の実行性能を測定した．評価実験の結果，CPU 版/GPU 版のいずれのコンテナ型仮想環境でも著しい実行性能の差はみられず，全体を通して 2%以下の差であり，実行性能においてはいずれのコンテナ型仮想環境を利用しても大きな影響がないことを確認した．

ここまでの報告を以下の 2 つの視点で総括する．

生命情報科学アプリケーションの開発者・利用者として

表 3　ソフトウェア環境（システム環境，コンテナ環境）
Table 3　Software Specification (Native, Containers)

	Native	Container (CPU)	Container (GPU)
Base Image	-	centos/7.4.1708	nvidia/cuda:9.1-devel-centos7
Linux Kernel	3.10.0-693.21.1	-	-
OS	CentOS 7.4.1708	CentOS 7.5.1804	CentOS 7.5.1804
GCC	4.8.5	4.8.5	4.8.5
FFTW	3.3.5	3.3.5	3.3.5
OpenMPI	1.10.6	1.10.7	-
NVCC	9.1.85	-	9.1.85

は，コンテナ型仮想化に着手する場合には Docker を利用して仮想イメージを作成することが無難である．これは以下の理由によるものである．

- Docker イメージは最も普及しているコンテナ仮想イメージの形式であり，多くの場合に他の環境への互換性があるため，汎用性が高い．
- 最も活発に開発されており，規格化・標準化が進められていることから，商用クラウド環境など様々な計算機環境で動作する見込みが高い．
- Linux だけでなく Windows/Mac 等の他の OS をサポートするツールセットが存在するため，可搬性と利便性の面でも優れる．

HPC システムによっては Singularity, Shifter を利用した環境を利用する必要があるが，多くの利用方法は Docker のコンテナワークフローと互換性があるため，まずはローカルな環境で Docker を利用してアプリケーションを管理することを推奨する．

一方，生命情報科学の研究者が利用する計算機クラスタの管理者としては，個人用ノートブックや利用者が限定された小規模な計算機クラスタの場合には Docker の利用を推奨する．そうでない場合，多数のユーザーが利用する共用の計算機資源の場合などでは，Singularity の利用（またはスケジューラ統合の Docker）を推奨する．これは以下の理由による．

- Docker は最も高機能で普及しているが特権ユーザの権限を要求する．個人用ノートブックや小規模クラスタであれば問題は小さいが，多数のユーザーの利用を想定する場合にはセキュリティ上の懸念となる．
- Docker の運用では仮想イメージにマウントする領域の設定，不要な仮想イメージやプロセスの破棄など，共用の計算機上では適切な設定と運用ポリシーが必要となる．
- Singularity は一度管理者が設定を行えば通常ユーザでも利用が可能であり，セキュリティ問題が完全に解消されるわけではないが，Docker よりも相対的に問題の影響が小さい．
- Singularity では仮想イメージが通常のファイル形式で出力されるため，管理者が意識する必要がない．デフォルトの設定でほとんど問題なく動作する．
- Singularity はユーザが作成した Docker イメージとの互換性がある．イメージの作成は手元の環境で行い，解析は計算機クラスタで行うなどのワークフローが実現可能である．

Shifter については実行性能と機能上の問題はないものの，仮想イメージ管理サービスの運用コスト，導入コストが課題である．ただし，この点については現行で Shifter の開発を継続する CSCS が前述の問題を解決した別ブランチを公開予定であり，懸念が解消される見込みがある [21]．

コンテナ型仮想化はアプリケーションの可搬性を向上させるだけでなく，ソフトウェア環境の仮想化により研究成果の再現性の向上に寄与する．生命情報科学は生命と情報の境界領域にある学問であり，情報科学のソフトウェア開発や IT インフラに関わる知見を利用して研究活動の効率化が可能である．今後も情報科学の進歩に適応して研究ワークフローそのものをアップデートする必要があるだろう．

謝辞　本研究の一部は，JSPS 科研費（15K16081, 17H01814, 18K18149），JST CREST「EBD: 次世代の年ヨッタバイト処理に向けたエクストリームビッグデータの基盤技術」（JPMJCR1303），JST リサーチコンプレックス推進プログラム，文部科学省地域イノベーション・エコシステム形成プログラム，AMED BINDS (JP17am0101112)，Microsoft Business Investment Funding，リバネス研究費の支援を受けて行われた．

参考文献

[1] B. K. Beaulieu-Jones and C. S. Greene, “Reproducibility of computational workflows is automated using continuous analysis,” *Nature Biotechnology*, vol. 35, no. 4, pp. 342–346, 2017.
[2] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, “An updated performance comparison of virtual machines and Linux containers,” *IEEE International Symposium on Performance Analysis of Systems and Software*, pp. 171–172, 2015.
[3] “Docker.” https://www.docker.com/.
[4] G. M. Kurtzer, V. Sochat, and M. W. Bauer, “Singular-

ity : Scientific containers for mobility of compute," *PLoS ONE*, vol. 12, no. 5, e0177459, pp. 1–20, 2017.

[5] A. Paolo, D. Tommaso, A. B. Ramirez, E. Palumbo, C. Notredame, and D. Gruber, "Benchmark Report : Univa Grid Engine , Nextflow , and Docker for running Genomic Analysis Workflows," *Univa White Paper*, 2015.

[6] 松嶋明宏, "科学技術計算用クラスタへの Docker 導入と運用," **第 *1* 回 *HPC OPS* 研究会**, 2018.

[7] "DockerHub." https://hub.docker.com/.

[8] S. R. Canon and D. Jacobsen, "Shifter : Containers for HPC," *Cray User Group*, pp. 1–8, 2016.

[9] "nvidia-docker." https://github.com/NVIDIA/nvidia-docker, 2016.

[10] "Use the Device Mapper storage driver." https://docs.docker.com/storage/storagedriver/device-mapper-driver/.

[11] "Open Container Initiative." https://www.opencontainers.org/, 2015.

[12] X. Lu and D. K. Panda, "Is Singularity-based Container Technology Ready for Running MPI Applications on HPC Clouds?," *Proceedings of the 10th International Conference on Utility and Cloud Computing (UCC '17)*, pp. 151–160, 2017.

[13] L. Benedicic, F. A. Cruz, A. Madonna, and K. Mariotti, "Portable, high-performance containers for HPC," *arXiv*, no. 1704.03383, 2017.

[14] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nature Biotechnology*, vol. 35, pp. 316–319, 2017.

[15] 二階堂愛, "BioDevOps による再現性のあるバイオインフォマティクス環境の構築." 第 31 回 DDBJing 講習会, 2015.

[16] Hitoshi Sato, "Building Software Ecosystems for AI Cloud using Singularity HPC Container," *5th Accelerated Data Analytics and Computing (ADAC5) Workshop*, 2018.

[17] M. Ohue, T. Shimoda, S. Suzuki, Y. Matsuzaki, T. Ishida, and Y. Akiyama, "MEGADOCK 4.0: An ultra-high-performance protein-protein docking software for heterogeneous supercomputers," *Bioinformatics*, vol. 30, no. 22, pp. 3281–3283, 2014.

[18] 大上雅史，山本悠生，秋山泰, "Microsoft Azure 上でのタンパク質間相互作用予測システムの並列計算と性能評価," **情報処理学会研究報告 バイオ情報学（*BIO*）**, vol. 2017-BIO-4, no. 4, pp. 1–3, 2017.

[19] 山本悠生，大上雅史，秋山泰, "クラウド上の分散 GPU 環境におけるタンパク質間相互作用予測計算フレームワークの開発," **情報処理学会研究報告 バイオ情報学（*BIO*）**, vol. 2018-BIO-5, no. 8, pp. 1–8, 2018.

[20] R. Chen, J. Mintseris, J. Janin, and Z. Weng, "A protein-protein docking benchmark," *Proteins*, vol. 52, no. 1, pp. 88–91, 2003.

[21] A. Madonna, L. Benedicic, F. A. Cruz, and K. Mariotti, "Shifter at CSCS - Docker Containers for HPC," *HPC Advisory Council Swiss Conference*, 2018.