PacificTeck
HPC and Machine Learning Experts

Singularity による アプリケーション環境

2021/04/12 , 14

ABCI ミニキャンプ（オンライン）

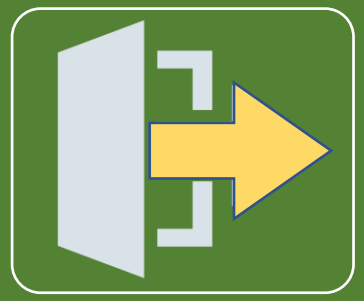
背景

- 多彩なアプリケーションとその実行環境が高頻度に更新されるAI分野では、全要件を包含するようなシステム構築は非現実的。
- コンテナを用いると、システムに手を入れることなく実行環境が用意できる。
- 複数システムの併用が必要な場合、再現性を担保するには相応のスキルが必要。
- 多くのコンテナ実装がある。(Docker, Singularity, Podman, Shifter…)
- Singularity は、米国Lawrence Berkeley National Laboratoryのスパコンで生まれ、スパコン利用者が直感的に使えるつくり。
- アプリケーションを直接実行できるため、ジョブスケジューラとの連携が容易。

Singularityの際立った特徴

シングルファイル

- 独自のSIFフォーマットでコンテナを１ファイルに集約しており、
- 通常のファイルコピーだけでハンドリング可能
- 電子署名によりイメージの同一性を担保、改変防止と高い再現性



ホストのリソースやネームスペースを共有

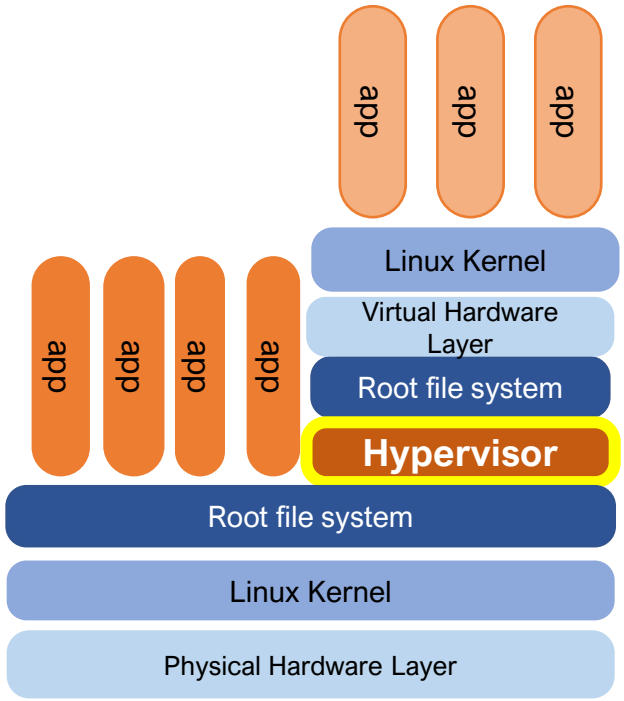
- 起動したユーザーと同一 ID でアプリケーションが稼働
- プロセス空間とデバイス・ネットワークを完全共有
- 事前のデータ転送やネットワーク設定が不要

Root権限と無縁の運用が可能

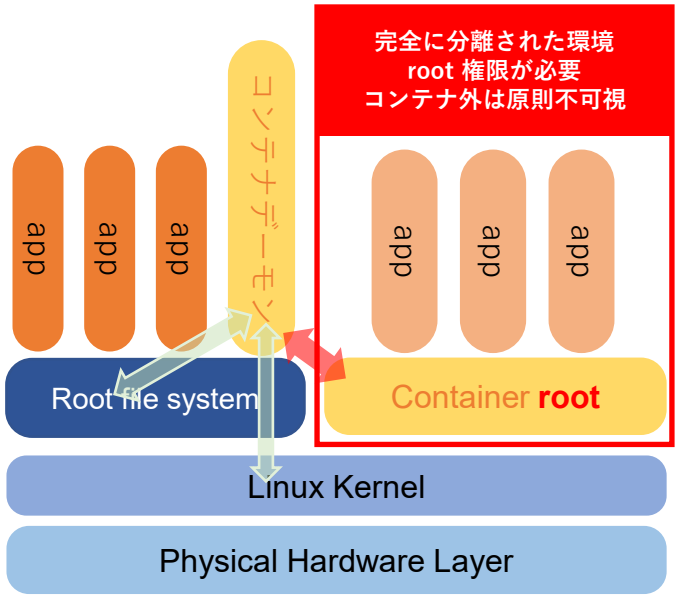
- コンテナ実行やコンテナ内でもroot権限が必要にならない運用が可能
- イメージ作成にもroot権限が不要（OS対応/RemoteBuilderが必要）
- イメージの保存にシステムのスプールディレクトリを不使用

Singularityと他の仮想環境との比較

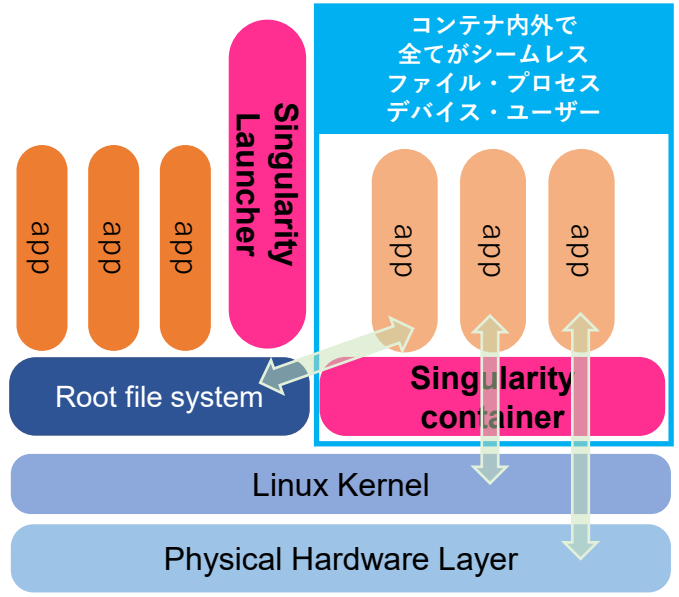
ハイパーバイザ型仮想環境



Docker コンテナ型仮想環境



Singularity コンテナ型仮想環境



- コンテナ型はハイパーバイザがないため、仮想化によるオーバーヘッドがない
- Dockerはコンテナの内外が分断され独立した環境となるが、Singularityは同一ユーザーで稼働できる
- Singularityはホストのリソース(ファイル・ハードウエア)を直接利用できる

HPC向け仮想コンテナーSingularity

Singularity CE

SingularityPRO

Singularity Enterprise

SingularityCE・SingularityPRO・Singularity Enterprise

SingularityCE

- オープンソースで提供・有償サポートなし。
- Sylabs社のパブリッククラウドサービス(RemoteBuilder, KeyStore, Library)が限定的に利用可能
- 最新機能はいち早く搭載。セキュリティ対応やバグ修正は最新版にのみ反映

SingularityPRO

- Sylabs社による検証・署名済みバイナリを提供
- 有償・長期サポート(リリースから2年)・機能追加・プラグイン開発のリクエスト
- 脆弱性情報の公開前に対応済みバイナリを提供し、バックポートにも対応
- Sylabs社のパブリッククラウドのフルサービスが利用可能

Singularity Enterprise

- SingularityPROを提供
- Kubernetesで稼働するプライベートクラウドサービスのRemoteBuilder, KeyStore, Libraryを完全セットで提供
- 自社・自サイトの認証基盤を統合可能(パブリッククラウドでは不可)

Singularity Enterprise を構成するサービス

Remote Builder

- root権限なしでコンテナイメージを作成できる外部サービス
- WebUIならびにSingularityコマンドから接続して利用
- AWS、GCP等の外部リソースを作成環境として指定可能

Library

- コンテナイメージのリポジトリサービス
- 作成したイメージをタグ情報を付与して管理
- AWS S3等の外部オブジェクトストレージを接続可能

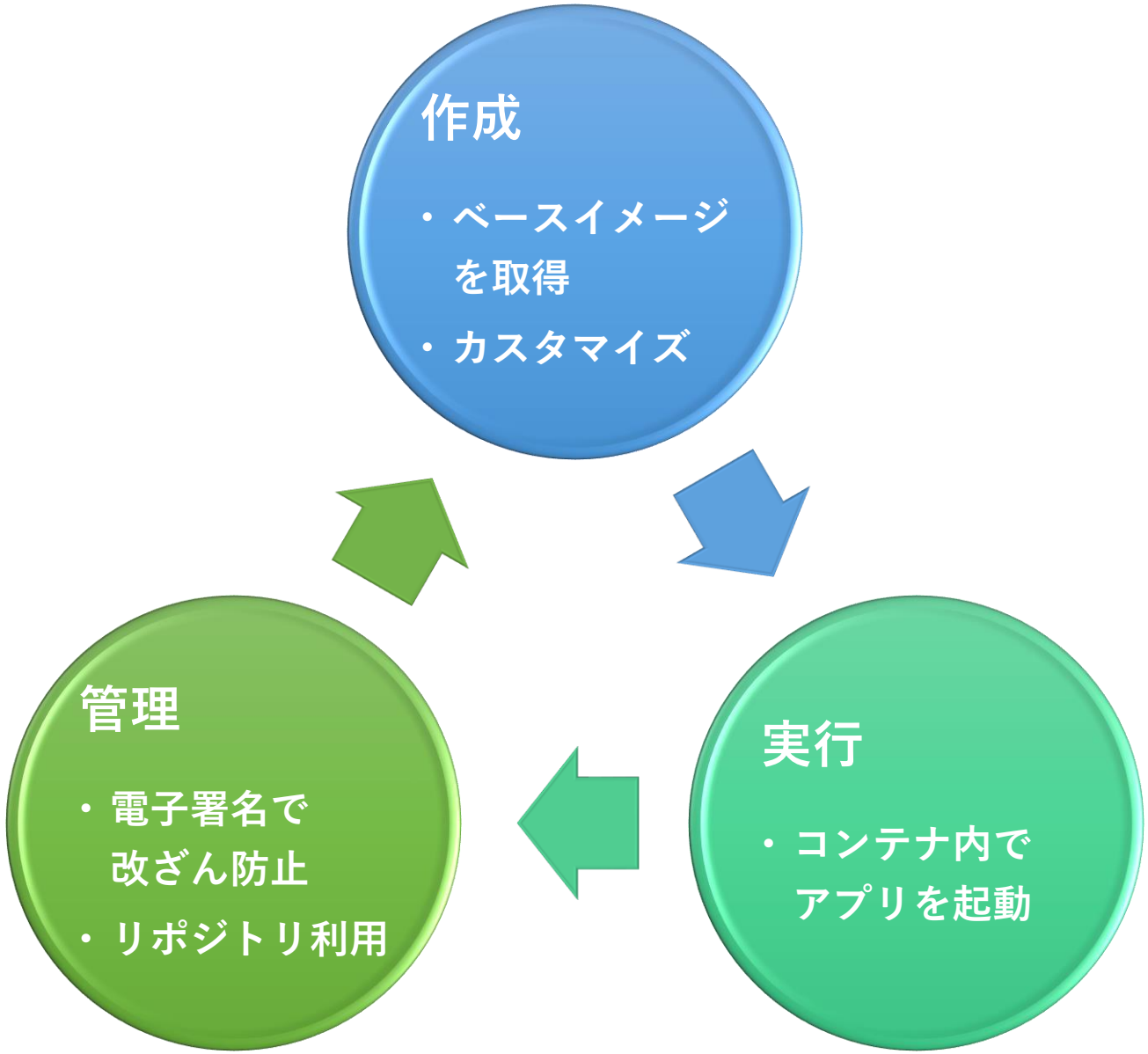
Key Store

- イメージに付与された電子署名のベリファイを行う公開鍵の管理サービス
- 共有されたイメージを実行前に検証して、改変防止しつつ再現性を担保
- 自身が署名したイメージを、誰もが検証できる公開鍵を安全に管理・共有

ドキュメント類

- 公式マニュアル
  - User Guide — Singularity User Guide 3.7 documentation (sylabs.io)
    - https://sylabs.io/guides/3.7/user-guide/
- ABCI マニュアル
  - 9. Linuxコンテナ - ABCI User Guide
    - https://docs.abci.ai/ja/09/

Singularity利用のサイクル



イメージの取得とカスタマイズ

Singularityのイメージ形式は SIF イメージファイルと、イメージを手元のファイルシステムに展開したSandboxイメージの2種類です（他のイメージを使うこともできます）。

イメージの作成にあたりベースとなるイメージを再利用するのが手っ取り早いです。Singularityが使えるイメージの主な取得先としては、上記のSingularityイメージの他、以下があります。

- Docker
  - Docker Hub および互換リポジトリ（Redhat UBI, NVIDIA NGC等）の直接利用が可能です。
- Library
  - SIngularity Enterprise Library = ABCI 内
  - Sylabs 社クラウドサービス
- OCI Registry
  - コンテナイメージの標準規格。Azure Container Registry 等が利用可能です。

イメージの取得と実行　-from Docker-

```
$ module load singularitypro/3.7
$ singularity pull docker://ubuntu:21.04
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
Getting image source signatures
＜＜中略＞＞
INFO:    Creating SIF file...
$ ls -l *.sif
-rwxr-x---   1 acd13367mq acd13367mq 28446720 Apr 27 17:12 ubuntu_21.04.sif
```

Let's TRY

```
$ singularity exec ubuntu_21.04.sif cat /etc/os-release  # OS 確認
$ singularity exec ubuntu_21.04.sif ls -l                # ホームディレクトリ
$ singularity shell ubuntu_21.04.sif                     # シェル起動
Singularity> cat /etc/passwd (or /etc/hosts)             # UID の共有を確認
Singularity> ps ef                                  # プロセス空間の共有を確認
Singularity> uname -r                               # カーネルがホスト側と同一
```

イメージの取得と実行　ーABCI Libraryー

```
$ singularity remote list
Cloud Services Endpoints
========================
NAME          URI                    ACTIVE   GLOBAL   EXCLUSIVE
ABCI          cloud.se.abci.local    YES      YES      NO
SylabsCloud   cloud.sylabs.io        NO       YES      NO
＜＜後略＞＞
$ singularity search mycamp
Found 1 container images for amd64 matching "mycamp":
        library://acd13367mq/c/mycamp:mcamp21
        library://acd13367mq/c/mycamp:sd

$ singularity pull mycamp.sif library://acd13367mq/c/mycamp:sd
```

Let's TRY

```
$ singularity inspect mycamp.sif          # イメージの情報
$ head -n 1 ./mycamp.sif                  # イメージの先頭行を確認
$ ./mycamp.sif --center welcome to MiniCamp21
                                          # イメージの直接実行
```

イメージの取得と実行　–from NGC-

```
$ singularity pull ngctf.sif docker://nvcr.io/nvidia/tensorflow:21.04-tf1-py3
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
Getting image source signatures
Copying blob a70d879fa598 done
Copying blob c4394a92d1f8 done
<<後略>>
sha256:a1e08a896305b05ac040e4021b65c94947188fb9d9e3d84bd27ad1b341377062
INFO:    Creating SIF file...
$ ls –lh ngctf.sif
-rwxr-x--- 1 acd13367mq acd13367mq 5.2G May 10 19:42 ngctf.sif
```


```
$ singularity exec ngctf.sif cat /etc/os-release  # OS 確認
$ singularity exec ngctf.sif mpirun --version     # ホスト側にない MPI
$ ./ngctf.sif python                              # イメージの直接実行
```

イメージのカスタマイズ　–def ファイル-

直接実行時にコンテナ内で設定される環境変数や、実行される処理を記述して埋め込むことができます。

```
$ cat sd.def
Bootstrap: localimage
From: mycamp.sif
%post
    mkdir -p /opt/minicamp/bin
    apt update; apt install -y curl
    curl https://raw.githubusercontent.com/fumiyas/home-commands/master/echo-sd >
/opt/minicamp/bin/echo-sd
    chmod 755 /opt/minicamp/bin/echo-sd
    apt remove –y curl; apt clean all
%environment
    export PATH=/opt/minicamp/bin:$PATH
%runscript
    echo-sd $*

$ singularity build –-fakeroot sd.sif sd.def
```

ベースイメージ等の指定

ベースイメージを展開し、
コンテナとして起動後に実行され、
イメージのカスタムを行う

実行時の環境やコマンドの埋め込み

イメージのカスタマイズ例　–def ファイル-

以下は、Redhatの公式コンテナイメージをベースにして、PyTorchをインストールし、イメージの実行時には直接Pythonを起動するようにした例です。

```
$ cat pt.def
Bootstrap: docker
From: registry.access.redhat.com/ubi8/python-38
%post
   /opt/app-root/bin/pip install torch torchvision –f ¥¥
    https://download.pytorch.org/whl/torch_stable.html
%runscript
    python $*

$ singularity build –f (or –remote) pt pt.def
$ ./pt tutorial.py
```

同様にアプリケーションの起動コマンドやオプションを埋め込んでおくことが可能です。

```
%runscript
    NP=$1; shift
    mpirun –np $NP python $*
```

イメージのカスタマイズ例　–sandboxイメージ経由-

Sandboxイメージは、コンテナイメージをディレクトリに展開したものです。SIFイメージは編集できませんが、Sandboxイメージは書き換えが可能です。buildコマンドで相互に変換できます。

```
$ singularity build –sandbox mysandbox mycamp.sif
$ ls –l mysandbox
total 60
lrwxrwxrwx  1 acd13367mq acd13367mq    7 May 11 09:53 bin -> usr/bin
drwxr-xr-x  2 acd13367mq acd13367mq 4096 Apr 19 16:26 boot
drwxr-xr-x  2 acd13367mq acd13367mq 4096 May 11 09:53 dev
＜＜後略＞＞
$ tar xzpf ~/prebuild-application.tar.gz –C mysandbox/usr/local
$ singularity build –f mycamp-appl.sif mysandbox
```

sandboxイメージからコンテナを起動して root での操作が必要な場合、以下のようにします。

```
$ singularity shell –f -w mysandbox
```

コンテナ内で pip を使う際に-wだけで起動すると、ホスト側と共有しているユーザーディレクトリへインストールされてしまい、イメージ内に残らないので注意してください。

GPUの利用

Singularity はカーネルから /dev, /proc, /sys, /tmp をデフォルトでコンテナ内にマウントします。
そのため、何もしなくてもデバイスそのものは見えています。

これを利用するためのCUDA等ランタイムライブラリの取り扱いについて 2 つの方法があります。

1. **コンテナイメージにCUDAをインストールしてしまう**
   特定バージョンのものがどうしても必要になる場合に有効です。ただしイメージがかなり大きくなることと、カーネルに組み込まれたデバイスドライバのバージョンとの整合性に注意が必要です。

2. **--nv オプションの利用**
   ホスト側にインストールされているライブラリをコンテナ内に取り込んで使えるようにします。取り込まれるライブラリのリストは /etc/singularity/nvliblist.conf にあります。実行環境によって内容が変わることを意味しており、アプリケーションが要求するバージョンとの整合性に注意する必要があります。

```
$ qrsh –g グループ名 -l rt_AF=1 -l h_rt=1:00:00
$ singularity exec –nv ngctf.sif python cnn_mnist.py
```

追加：留意事項など

- singularity buildを実行すると、ベースとなるイメージを$TMPDIRに展開してsandboxとします。
  それをコンテナ起動してroot権限により%postの処理を行うという流れになっています。
  そのため、コンテナ内限定でroot権限を行使できる—fakerootオプションが必須となります。
  ただし、単純にsandboxイメージを SIFイメージに変換するだけであれば不要です。
- fakerootはカーネルのユーザーネームスペースという機能を用いて、独立したUID空間を作り、IDのマッピングを行って疑似的にコンテナ内だけで有効なrootを生成します。そのためコンテナ外からは、起動したユーザーが処理を行っているように見えます。
- マッピングは当該マシンのカーネルの支配下でのみ有効で、これをノード間で共有することができません。これはストレージアクセスで問題となります。すなわちLustreやGPFS、NFSといった共有ファイルシステム（すなわちホームディレクトリ）を構成するストレージサーバーではオーナー情報との整合性が取れなくなることを意味し、I/Oが失敗します。これを回避するには、展開先をローカルストレージとする必要があります。$TMPDIRがそういった共有ストレージ内を指している場合はこれを変更するか、変えられない場合は$SINGULARITY_TMPDIRを別途設定してください。
- ネットワークブート等によりローカルストレージがないマシンの場合は、iSCSI等でストレージを供給するか、/dev/shmのようなtmpfsを使う方法があります。（「富岳」がこの運用です）
- SIFの実体はsquashfsという圧縮ファイルシステムで、ローカルのメモリ上で展開します。そのため、ホームディレクトリ内に展開したライブラリをimportしまくるよりもストレージの負荷が低く、環境によってはアプリケーションの起動等が有意に速いことがあります。特に多数のジョブを同時展開する場合は有利かもしれません。

Thank you

sales@pacificteck.com