2022年4月13日 更新

Type IIサブシステム向け、Singularityの利用方法

名古屋大学
NAGOYA UNIVERSITY

更新履歴など

- 2020年9月3日の医用画像研究会の発表を聴講した方へ
  - 発表時間に収まるように基本的な部分だけ説明させていただいたつもりでしたが、情報を削りすぎて一部間違った情報を提供してしまいました。
  - 本資料の情報の方が正確ですので注意してください。
  - 利用例2と利用例3が混ざったような情報をお話ししてしまいました。大変申し訳ありませんでした。
- 2020年10月08日
  - 利用例3に注意点（SSD上でビルドする必要がある）を追加、その他細かい修正
- 2020年10月12日
  - 利用例5を修正
- 2020年10月23日
  - 利用例4にコメント追加、その他細かい修正
- 2021年4月15日
  - 2021年度システム向けに更新（Singularityのバージョンについての追記など細かい修正）
- 2021年4月30日
  - singularity 3.5.3→3.7.1の注意点を追記
- 2022年2月28日
  - singularityのバージョンのデフォルトを3.7.4に更新
  - 共有コンテナ置き場についての情報を追記
  - openmpi_cudaのバージョンを4.0.5から4.0.4に変更（バージョンを合わせるため統一、4.0.5でも問題は発生しない）
- 2022年4月13日
  - 2022年度の環境向けに更新。Singularityのデフォルトバージョンは3.9.5に上がっています。

「不老」におけるSingularityの利用方法

- Type IIサブシステムではSingularityが利用可能
  - SingularityでDockerコンテナを利用可能、コンテナからGPUを利用可能
  - Docker Hubから構築済みのコンテナイメージを持ってきたり、NVIDIAのNGCやClaraを使って素早く機械学習や医用画像処理の研究を行うことが可能
  - ノード間は高速なInfiniBand EDRで接続されているため、多数のノードを用いた大規模分散実行でも高い性能が期待できる

- スーパーコンピュータの利用はバッチ処理が一般的だが、コンテナ環境の導入・調整はインタラクティブジョブな処理ができないと不便
  → 「不老」はインタラクティブな処理（会話型バッチ）にも対応しているため、会話型バッチで実行環境の準備とテスト実行を行った後にバッチジョブを実行するという使い方を想定している

「不老」Type IIにおけるSingularityを用いた作業の流れ（利用想定）

- 想定しているいくつかの利用方法を紹介する
  - 利用例1：リポジトリで公開されているコンテナイメージをそのまま利用する
  - 利用例2：コンテナイメージを元に環境構築して利用する
  - 利用例3：コンテナイメージ自体をカスタマイズして利用する
  - 利用例4：.defファイルでコンテナイメージを作成して利用する
  - 利用例5：手持ちのDockerコンテナから変換して利用する

  - 「jobenv=singularity」の設定はどの利用例でも重要になるので忘れずに指定してください

Singularity 3.5.3→3.7.xにおける注意点など（1/2）

- 2020年度はmodule load singularityを実行するとsingularity/3.5.3がロードされましたが、2021年度ではsingularity/3.7系がロードされます。基本的には同じように使えますが、いくつか異なる挙動があるため注意してください。
  - 実行時のオプションに関すること
    - Singularityの仕様の更新によりいくつか実行時のオプション等に違いが生じています。オプションを追加して実行していた場合はマニュアル等で違いがないか確認してください。
  - 実行時のディレクトリに関すること
    - 3.7系では/data/group1側でコンテナを起動するとコンテナ起動時のカレントディレクトリがホームディレクトリになってしまうことが確認されています。（3.6での仕様変更の影響のようです。）オプションを追加すれば従来通り使えます。
    - 追加するオプションの例：--bind /data/group1/$USER:/data/group1/$USER
      - /data/group1/以下のユーザディレクトリがsingularity内からも同じように見えるようになる。
    - 以前紹介していたオプションの例：--bind $PWD:$PWD
      - singularityコマンドを実行するパスがsingularity内からも同じように見えるようになる。singularity内からカレントディレクトリより上位のパスを参照する必要がある場合はこれでは足りない点に注意が必要。

Singularity 3.5.3→3.7.xにおける注意点など（2/2）

- OpenMPIとの組み合わせに関すること
  - OpenMPIと組み合わせて利用する、特にmpirunでsingularityを呼び出す使い方でsingularityを使いたい場合は、cuda/11.2.1 と openmpi_cuda/4.0.4 と singularity/3.7.4（または3.7.1）のmoduleをloadして使ってください。~~openmpi/4.0.4 と singularity/3.7.1 だけloadすると singularity/3.5.3が使われてしまいます。~~ loadできるmoduleの組み合わせを見直し、この組み合わせのloadはできなくなりました。（singularity/3.5.3の提供は終了しました）

Singularity 3.9.5では？

- 基本的にSingularity 3.7系と同様の使い方となります
  - /data/group1　以下でコンテナを利用する場合は--bindオプションが必要です
  - --bind /data:/data　としてしまうのが一番手っ取り早いようです（何か弊害があるのかどうかはわかっていません）

Singularity + MPI（Singularity 3.7.4版）

```
[a49979a@flow-cx04 20220413]$ pjsub --interact -L rscgrp=cx-workshop,jobenv=singularity
[INFO] PJM 0000 pjsub Job 688440 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 688440 started.
[a49979a@cx051 20220413]$ module load gcc/8.4.0
[a49979a@cx051 20220413]$
[a49979a@cx051 20220413]$ module load openmpi
openmpi        openmpi/4.0.4  openmpi/4.1.1  openmpi/4.1.2
[a49979a@cx051 20220413]$ module load cuda/11.6.2
[a49979a@cx051 20220413]$
[a49979a@cx051 20220413]$ module load openmpi
openmpi             openmpi/4.0.4       openmpi/4.1.1       openmpi/4.1.2       openmpi_cuda        openmpi_cuda/4.1.2
[a49979a@cx051 20220413]$ module load openmpi_cuda
openmpi_cuda        openmpi_cuda/4.1.2
[a49979a@cx051 20220413]$ module load openmpi_cuda/4.1.2
[a49979a@cx051 20220413]$
[a49979a@cx051 20220413]$ module load singularity
singularity        singularity/3.7.1  singularity/3.7.4  singularity/3.9.5
[a49979a@cx051 20220413]$ module load singularity/3.9.5
[a49979a@cx051 20220413]$ module list
Currently Loaded Modulefiles:
  1) gcc/8.4.0            2) cuda/11.6.2          3) openmpi_cuda/4.1.2   4) singularity/3.9.5
[a49979a@cx051 20220413]$
```

gcc/8.4.0をload後にmodule load openmpiまで入力してTABで補完するとopenmpi系のmoduleが見える（OpenMPIのバージョン違いが複数用意されている）

同様にcuda/11.6.2をload後はopenmpi_cuda系のmoduleが見える

gcc/8.4.0 cuda/11.6.2 openmpi_cuda/4.1.2 singularity/3.9.5
の組み合わせが基本となります

参考（旧版）：Singularity 3.5.3→3.7.1における注意点など（実行例）

```
[a49979a@flow-cx02 work]$ pjsub --interact -L rscgrp=cx-workshop,jobenv=singularity
[INFO] PJM 0000 pjsub Job 447211 submitted.
[INFO] PJM 0081 .connected.
[INFO] PJM 0082 pjsub Interactive job 447211 started.
[a49979a@cx094 work]$ module load gcc/8.4.0 cuda/11.2.1 openmpi/4.0.5 singularity/3.7.1
[a49979a@cx094 work]$ mpirun -n 2 which singularity
/home/center/opt/x86_64/apps/singularity/3.7.1/bin/singularity
/home/center/opt/x86_64/apps/singularity/3.7.1/bin/singularity
[a49979a@cx094 work]$ mpirun -n 2 singularity --version
singularity version 3.5.3
singularity version 3.5.3
[a49979a@cx094 work]$ module purge
[a49979a@cx094 work]$ module load cuda/11.2.1 openmpi_cuda/4.0.5 singularity/3.7.1
[a49979a@cx094 work]$ mpirun -n 2 which singularity
/home/center/opt/x86_64/apps/singularity/3.7.1/bin/singularity
/home/center/opt/x86_64/apps/singularity/3.7.1/bin/singularity
[a49979a@cx094 work]$ mpirun -n 2 singularity --version
singularity version 3.7.1
singularity version 3.7.1
[a49979a@cx094 work]$ logout
[INFO] PJM 0083 pjsub Interactive job 447211 completed.
```

gcc/8.4.0をloadしないとopenmpi/4.0.5はloadできない

OpenMPIのビルド時オプションの都合により、この使い方では実際には singularity/3.5.3が実行されてしまう

cuda/11.2.1をloadしないとopenmpi_cuda/4.0.5はloadできない

openmpi_cuda/4.0.5をロードすればsingularity/3.7.1が実行される

以上の都合により、2021年度のシステムでは
`module load cuda/11.2.1 openmpi_cuda/4.0.5 singularity/3.7.1`
のmoduleの組み合わせを利用し、singularityのexecやshell実行時に`--bind $PWD:$PWD`を指定するという使い方を推奨します。（ただし、OpenMPIのより良い（高い通信性能が得られる）ビルドオプションなどが判明し新たなmoduleを用意した際にはまた変わるかもしれません。ご了承ください。）

利用例1：リポジトリで公開されているコンテナイメージをそのまま利用する 1/2

- Docker Hubなどで公開されているリポジトリを使ったプログラム実行やコンテナイメージの作成は数行で可能

- バッチジョブで実行する例
  - 以下を記述したファイルを用意、pjsubコマンドでジョブを投入

```
#!/bin/bash
#PJM -L rscunit=cx
#PJM -L rscgrp=cx-debug
#PJM -L node=1
#PJM -L elapse=10:00
#PJM -L jobenv=singularity
#PJM -j
#PJM -S

module load singularity
singularity exec --nv docker://<イメージの在処>/ コンテナ上で実行したいコマンド
```

ジョブの設定

--nvでGPUが利用可能になる

- 会話型バッチで実行する例　※紫の文字列が入力するコマンド

```
$ pjsub --interact -L rscgrp=cx-interactive,jobenv=singularity
（会話型ジョブが起動される）
$ module load singularity
$ singularity shell --nv docker://<イメージの在処>/
Singularity> あとはコンテナ上で実行したいコマンドを実行していく
```

- 初回実行時はコンテナイメージのダウンロードに時間がかかるが、~/.singularity/cache/以下にキャッシュされるため二回目からは起動時間が短縮される

利用例1：リポジトリで公開されているコンテナイメージをそのまま利用する 2/2

- もちろん、コンテナイメージを明示的に保存して再利用することも可能
  - 基本的な手順

```
$ pjsub --interact -L rscgrp=cx-interactive,jobenv=singularity
 (会話型バッチが起動される)
$ module load singularity
$ singularity build 保存先イメージファイル名 docker://<元となるイメージの在処>/
$ singularity shell --nv 保存先イメージファイル名
```

  - 具体的な例

```
$ singularity build tmp.sif docker://tensorflow/tensorflow:latest-gpu
INFO:    Starting build...
 (途中省略)
INFO:    Creating SIF file...
INFO:    Build complete: tmp.sif
$ singularity shell --nv ./tmp.sif
Singularity> nvidia-smi
 (結果省略)
```

↑例として、TensorFlowのGPU対応コンテナを手元のファイルにイメージとして保存
←tmp.sifに保存された
←singularity shellで呼び出して利用
←GPU対応コンテナなので--nvオプション付きでsingularity shellを起動すれば計算ノードに搭載された4つのGPUが利用可能

利用例2：コンテナイメージを元に環境構築して利用する 1/3

- 「普通に」コンテナを起動するとコンテナ内のファイルシステムがread onlyになってしまい幾つかの操作に支障をきたすが、Singularityからホットストレージにアクセスできるため、そのままでも様々な作業が可能
- 例：pipでアプリをインストールしようとするとホットストレージ上のホームディレクトリにインストールされる

- 例：Docker Hubのイメージを手元に持ってくる（この時点では前ページと全く同じ状態）

```
$ singularity build tmp.sif docker://tensorflow/tensorflow:latest-gpu
INFO:    Starting build...
（途中省略）
INFO:    Creating SIF file...
INFO:    Build complete: tmp.sif
$ singularity shell --nv ./tmp.sif
Singularity> nvidia-smi
（結果省略）
```

↑例として、TensorFlowのGPU対応コンテナを手元のファイルにイメージとして保存
←tmp.sifに保存された
←singularity shellで呼び出して利用
←GPU対応コンテナなので--nvオプション付きでsingularity shellを起動すれば計算ノードに搭載された4つのGPUが利用可能

– 次のページへつづく

利用例2：コンテナイメージを元に環境構築して利用する 2/3

```
Singularity> python                                    ← Pythonを起動するとPython 3.6.9が使えるが……
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd                                ← 例えばPandasを使おうとしてもインストールされ
Traceback (most recent call last):                     ておらず使うことができない
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'pandas'
>>>
Singularity> pip3 install pandas                       ← pipでPandasをインストール
Defaulting to user installation because normal site-packages is not writeable
Collecting pandas
  Downloading pandas-1.4.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.7 MB)
 (途中省略)
Successfully installed pandas-1.4.2 python-dateutil-2.8.2 pytz-2022.1
Singularity>                                           ↑（--userを付けなくてもuserレベルで）
                                                       インストールできたようだ
```

利用例2：コンテナイメージを元に環境構築して利用する 3/3

```
$ pjsub --interact -L rscgrp=cx-interactive,jobenv=singularity
 (起動時のメッセージは省略)                                    ↑ 改めて会話型バッチを起動
$ module load singularity                                    ← moduleをloadして
$ singularity shell --nv ./tmp.sif                           ← コンテナを起動
Singularity> python
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd                                      ← インストールしたPandasが使える状態であること
>>>                                                          を確認
```

- 実はこの例のpip3 installはコンテナ外でpip3 --userによるインストールを行ったのと同じであり、デフォルトで ~/.local/lib/python3.8/site-packages/ にインストールされる。
- そのため、コンテナ外で ~/.local/lib/python3.8/site-packages/ を参照したり、pip3 list | grep pandas を実行すればpandasの存在が確認できる。コンテナ外で pip3 uninstall pandas を実行すれば、コンテナからもpandasが使えなくなる。
- これで十分な場合もあるが、コンテナ自体を更新したいことも多いだろう。利用例3ではコンテナ自体を更新する場合の例を紹介する。

利用例3：コンテナイメージ自体をカスタマイズして利用する 1/2

- 他の環境へ持っていきたいときや他のユーザに渡したいときなどは、コンテナイメージ自体を更新した方が便利である
- コンテナイメージ自体を更新したい場合は少しだけ手間をかける必要がある
- 手順
  1. sandboxコンテナを作成する
  2. 書き込みできるオプションを付けてコンテナを起動する
  3. 改めてコンテナイメージに変換する（変換せずに使い続けても良い）

1. sandboxコンテナを作成する
   - -s (--sandbox)オプションを付けると、1ファイルにまとまっていないディレクトリ型のコンテナイメージを作成することができる

   Docker Hubから入手してそのままsandboxコンテナにする例
   ```
   $ singularity build -s ./tmp2 docker://tensorflow/tensorflow:latest-gpu
   ```
   既存のコンテナイメージからsandboxコンテナに変換する例
   ```
   $ singularity build -s ./tmp2 ./tmp1.sif
   ```

利用例3：コンテナイメージ自体をカスタマイズして利用する 2/2

2. 書き込みできるオプションを付けてコンテナを起動
   - -w (--writable)で書き込み可能に、-f (--fakeroot)でroot権限が必要な処理を解決

   オプション付きでsandboxコンテナを起動
   ```
   $ singularity shell -w -f ./tmp2
   ```
   ←sandboxを作成する場所に制限がある、詳しくは次ページ
   コンテナが起動するので、アプリケーションのインストールなどを行い、コンテナから抜ける。
   tmp2内のファイルが更新されることでコンテナ自体が更新される。

3. 通常のコンテナイメージに変換して利用する
   - （変換せずにディレクトリのまま使い続けても特に問題があるわけではない）

   sandboxコンテナから通常のコンテナイメージに変換する例
   （変換にはある程度時間がかかる）
   ```
   $ singularity build ./tmp2.sif tmp2
   INFO:    Starting build...
   INFO:    Creating SIF file...
   INFO:    Build complete: ./tmp2.sif
   ```
   変換後は通常のコンテナイメージとして利用可能
   ```
   $ singularity shell --nv ./tmp2.sif
   ```

利用例3：注意点 1/2

- コンテナに対して書き込みを行いたい場合は、sandboxディレクトリを作成する場所に制限がある
  - ホームディレクトリ（/home/ユーザID）やdataディレクトリ（/data/group1/ユーザID）はNG
    - Luster, FEFS, NFSではダメ
  - ローカルSSD上であれば問題なく実行できる
    - 「NVMeSSD6.4TB利用可能」なリソースグループ（cxgfs系以外）では$PJM_LOCALDIRにSSDがマウントされている
- うまくいかない例（/data/group1/ユーザID以下で実行）

※ インタラクティブジョブを起動した直後の状態を想定

```
$ module load singularity
$ singularity build -s ubuntu docker://ubuntu:latest
$ singularity shell -f -w ubuntu
Singularity> apt update
4 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: chown to _apt:root of directory /var/lib/apt/lists/partial failed - SetupAPTPartialDirectory (1: Operation not permitted)
W: chown to _apt:root of directory /var/lib/apt/lists/auxfiles failed - SetupAPTPartialDirectory (1: Operation not permitted)
W: Download is performed unsandboxed as root as file
'/var/lib/apt/lists/partial/archive.ubuntu.com_ubuntu_dists_focal_InRelease' couldn't be accessed by user '_apt'. -
pkgAcquire::Run (13: Permission denied)
```

ubuntuを拾ってきてaptのパッケージ情報を更新しようとしたが、エラーしてしまった。

利用例3：注意点 2/2

- うまくいく例（$PJM_LOCALDIRに移動して実行）

※ インタラクティブジョブを起動した直後の状態を想定

```
$ cd $PJM_LOCALDIR
$ pwd
/local/33618                    ← /local/ジョブIDにSSDがマウントされている
$ module load singularity
$ singularity build -s ubuntu docker://ubuntu:latest        ←すでに入手済のsifファイルからbuildしても良い
 （この時点でlsコマンドを使うとubuntuディレクトリが見える）
$ singularity shell -f -w ubuntu
Singularity> apt update
 （通常のapt実行時の出力が得られる）
Singularity> apt install -y vim
 （元々はインストーされていなかったvimがインストールされる）
Singularity> exit
exit
$ singularity build -f ubuntu.sif ubuntu
```

← これでsifファイルが完成。SSD上に置いてあるので/data/group1/ユーザID 側にコピーしておかないとジョブ終了時に失われてしまう点に注意。-f の付け忘れにも注意。（付けないとエラーする。）

```
$ singularity build -f /data/group1/userid/ubuntu.sif ubuntu
```

← 直接/data/group1/ユーザID 側にビルドすると失われなくて安心

利用例4：.defファイルでコンテナイメージを作成して利用する

- Singularityでは.defファイルを用いたコンテナの作成方法もよく用いられる
  - DockerにおけるDockerfileのようにインストール手順を書いたもの
  - これまでの例で使っていたtensorflowのコンテナをそのまま持ってくるだけの最低限の例

```
Bootstrap: docker
From:tensorflow/tensorflow:latest-gpu
```

  - 実行例

```
$ singularity build -f ./test.sif ./test.def
INFO:    Starting build...
Getting image source signatures
 (省略)
INFO:    Creating SIF file...
INFO:    Build complete: ./test.sif
$ singularity shell --nv ./test.sif
```

-fオプションは必要だが、SSDで行う必要はない。
.defファイル内でapt-getなどを走らせる場合も
走らせない場合も同様。

- さらに%postにapt/yumやpipなどを書いて
ソフトウェア環境を揃えたコンテナを作る
という使い方が一般的

```
Bootstrap: docker
From:tensorflow/tensorflow:latest-gpu

%post
apt-get update -y
apt-get install -y emacs
pip3 install pandas
```

利用例5：手持ちのDockerコンテナから変換して利用する

- 既に別環境で構築済みのDockerコンテナ（イメージ）を持っている場合、変換すればSingularityで利用できる

既存環境にてDockerコンテナをイメージファイルに保存する
```
$ docker save 既存のイメージ > tmp.tar
```
保存したファイルをscp/sftpなどで「不老」にコピーする

コピーしたファイルをsingularity buildで変換する
```
$ singularity build container.sif docker-archive:tmp.tar
```
変換後は通常のコンテナイメージとして利用可能
```
$ singularity shell --nv ./container.sif
```

FAQ

- Q. ファイルはどのように見えるのか、コンテナ内からホットストレージ上のデータは見えるのか？
- A. SingularityはDockerと異なりコンテナ外のファイルに簡単にアクセスできます。むしろ、コンテナ内のファイルを参照するつもりだったのにホットストレージのホームディレクトリ上に置いてあるファイルを参照してしまって躓くことがあるため注意してください。また、--bindオプションでコンテナ起動時のディレクトリのマウントができるので活用してください。

- Q. Dockerコンテナを持ち込む際に注意することは？
- A. コンテナを実行する際のオプション設定に気を付けてください。環境変数設定やディレクトリのバインド設定の書式が異なります。詳細はマニュアルを参照。

あらかじめダウンロード（pull）してあるコンテナの提供について

- 大容量のコンテナを外部からダウンロード（pull）してsifファイルを構築するにはある程度長い時間が必要
- 手間と時間がもったいないためいくつかのコンテナを共有ディレクトリにおいておきます

- 共有ディレクトリ
- /home/center/local/doc/singularity

- 共有して欲しいコンテナがあればご連絡ください（必ず対応できるとは限りません）