1506
UNIVERSITÀ
DEGLI STUDI
DI URBINO
CARLO BO

PH.D. **R**esearch **Me**thods in **S**cience and **T**echnology

Executing applications on RemestHPC

Usage of Singularity containers in a shared environment to access computation resources.

The software has grown in complexity over the years making it difficult sometimes to install, update and run it. Moreover, ReMeST is a strongly interdisciplinary PH.D., and the number of software that can be used by students/teachers is potentially high. Containers address this problem by storing the software and all its dependencies (including a minimal operating system) in a single image so that there is nothing to install. This makes the software both shareable and portable while the output becomes reproducible.

1. Accessing to the remestHPC server

The remestHPC server can be reached only through a VPN connection which needs to be activated by Uniurb ICT staff. Once activated, it will be possible to connect to the server using a VPN client (the use of **FortiClient VPN** is recommended). The VPN configuration is the follow:

```
remote gateway:      vpnssl.uniurb.it
port:                443
client certificate:  none
username:            your uniurb account
password:            your uniurb password
```

Once the VPN connection is active, you need to use an RDP client to access the remote desktop which is reachable at the following address: **172.19.2.250** with the default RDP port (**3389**).

Alternatively, a normal browser can be used to take advantage of the web access offered by the **Guacamole** application which is listening on the 8383 port of the same address **(172.19.2.250:8383)** using the following credential:

```
username:            remestHPC
password:            remestHPC
```

2. Singularity as a Secure Alternative to Docker

Docker images are not secure because a malicious user can gain root access, with a *privilege escalation* mechanism, to the system they are running on. For this reason, Docker is unavailable on the **RemestHPC** (neither is nvidia-docker). To overcome the problem, RemestHPC offers **Singularity**, an alternative to Docker that is both secure and designed for high-performance computing. Singularity is compatible with all Docker images and it can be used with several types of GPUs applications.

3. Running a Singularity container

Singularity is a container platform specifically for high-performance computing. It runs container images that are compressed on `.sif` files.

RemestHPC offers several ready-to-use containers accessible to the read-only directory `/software` of the main storage.

Running:

To run the default command within the Singularity image use, for example:

```
$ singularity run -p -B /run/user/$UID /software/remest_octave_6.2.0.sif
```

To execute the `octave` command with the argument `--gui` within the container use:

```
singularity exec -p -B /run/user/$UID /software/remest_octave_6.2.0.sif octave --gui
```

Notice that, the `-B /run/user/%UID` option force the locale directory `/run/user/$UID` to be mounted on the same path within the container. This is needed to allow `dconf` within the container to write shell or GUI configuration into the user space. A running container automatically mounts these paths:

```
1. /home/$USER  #your Linux home on remestHPC
2. /tmp
3. #the directory from which the container was run
```

To save the processing results it will therefore be sufficient to move within the file system of the container by identifying the mount point of your home on the remstHPC server.

Get a shell within the container:

Use the `shell` command to run a shell within a container:

```
$ singularity shell -p -B /run/user/$UID /software/remest_octave_6.2.0.sif

Singularity> cat /etc/os-release

Singularity> cd /

Singularity> ls -l
```

```
Singularity> exit
```

Environment Variables:

Singularity by default exposes all environment variables from the host inside the container. Use the `--cleanenv` argument to prevent this:

```
$ singularity run -p --cleanenv -B /run/user/$UID /software/remest_octave_6.2.0.sif
```

4. Accessing GPUs with Slurm

To administer the GPUs utilization, remestHPC uses the Slurm scheduler which offers an automatic allocation policy and a jobs enqueuing mechanism. Using Slurm to access the GPUs is mandatory because remestHPC hides these resources for the tasks executed outside of Slurm.

Requesting a GPU and running a task:

RemestHPC staff created a script to simplify the start of a singularity container through Slurm execution called `cstart`.

To execute a task and request one of the three available GPUs, for a maximum time of 4 hours use:

```
$ cstart -g 1 -h 4 <path to the container .sif file>
```

For instance, to request two GPUs and run the Matlab container with a time limit of 6 hours you can use:

```
$ cstart -g 2 -h 6 /software/remest_matlab_r2022b_gpu.sif
```

Notice that the maximum time you are allowed to lock a GPU is 48 hours, if you need more time please send an e-mail to remest@uniurb.it

You can also start a container using the native `srun` command where you need to specify the `--deadline=xxxx` slurm parameter to declare the maximum allowed time, and the `--nv` singularity parameter to allow it to mount Nvidia drivers. For instance, to specify a deadline of two hours you have to write: `--deadline=now+2hours`

```
$ srun --gpus=1 --pty --deadline=now+2hours singularity run -p --nv -B
/run/user/$UID /software/remest_matlab_r2022b_gpu.sif
```

Running a task in an interactive mode:

`srun` does not release the shell so if you need to write commands within the container use the option `--pty` to enable interactive mode.

```
$ srun --gpus=1 --pty  --deadline=now+2hours singularity run -p --nv  -B
/run/user/$UID /software/remest_matlab_r2022b_gpu.sif
```

If the requested CPUs are not available because already locked by other tasks, the scheduler enqueued the job until these are unlocked. So you have just to wait.

```
$ srun --gpus=2 --pty  --deadline=now+2hours singularity run -p --nv -B
/run/user/$UID /software/remest_matlab_r2022b.sif

srun: job 197 queued and waiting for resources
```

Schedule a batch execution:

Slurm support also the batch execution by means of `sbatch` command with an appropriate batch script.

Below is a Slurm script appropriate for a GPU code such as TensorFlow:

```
#!/bin/bash
#SBATCH --job-name=myjob          # create a short name for your job
#SBATCH --gpus=1                  # number of gpus per node
#SBATCH --time=00:05:00           # total run time limit (HH:MM:SS)
#SBATCH --mail-type=begin         # send email when job begins
#SBATCH --mail-type=end           # send email when job ends
#SBATCH --mail-user=<YourNetID>@uniurb.it
singularity exec -p -B /run/user/$UID --nv
/software/remest_tensorflow_22.10.1-tf2-py3_gpu.sif python3
/home/<user>/myPythonScript.py
```

How to kill a Slurm job:

The normal method to kill a Slurm job is:

```
$ scancel <jobid>
```

You can find your jobid with the following command:

```
$ squeue -u $USER
```

If the the job id is 123 then to kill the job:

```
$ scancel 123
```

5. Containers available on RemestHPC

RemestHPC offers the following container in `.sif` format located inside the `/software` read-only directory:

```
remest_gromacs_2022.3_gpu.sif

remest_matlab_r2022b_gpu.sif

remest_octave_6.2.0.sif

remest_openMM_gpu.sif

remest_qiime_core_2022.8.sif

remest_tensorflow_22.10.1-tf2-py3_gpu.sif
```

These original Docker containers have been modified to allow no-root execution within Singularity.

remest_gromacs_2022.3_gpu:

This allows the execution of GROMACS molecular dynamics analysis software using Nvidia GPU acceleration.

Use `cstart` to start the container with 2 hours of running limit and access one GPU:

```
cstart -g 1 -h 2 /software/remest_gromacs_2022.3_gpu.sif
```

remest_matlab_r2022b_gpu:

This allows the execution of Matlab2022b using both CPU and Nvidia GPU acceleration. You will be prompted to insert a valid Matlab license during the first run. See https://www.uniurb.it/ateneo/servizi-ict/utilita/matlab to configure your academic license.

Use `cstart` to start the container with 2 hours of running limit and access one GPU:

```
cstart -g 1 -h 2 /software/remest_matlab_r2022b_gpu.sif
```

remest_octave_6.2.0:

This is a Linux container whit installed **GNU Octave** and **Gnuplot** tools. It is NOT Nvidia accelerated container but it can still take advantage of the great performance of Xeon processors. This means that you do have not to request GPUs to run it.

Octave can be run both in the command line and in GUI environments (`--gui` as a running option).

To execute the `octave` in the command line environment with 4 hours of running limit; then start octave on the Singularity shell:

```
cstart -h 4 /software/remest_octave_6.2.0.sif

Singularity> octave
```

To execute it with GUI add `--gui` option:

```
cstart -h 4 /software/remest_octave_6.2.0.sif

Singularity> octave --gui
```

remest_openMM_gpu:

This is a Linux container whit installed the high-performance toolkit for molecular simulation called **openMM**.

To get a shell on the container with 4 hours running limit type:

```
cstart -h 4 /software/remest_openMM_gpu.sif

Singularity>
```

remest_qiime_core_2022.8:

This container is built starting from a Linux **Qiime** core container in which **Cutadapt** and **Gprename** have been installed. This is NOT an Nvidia accelerated container but it can still take advantage of the great performance of Xeon processors. This means that you do have not to request GPUs to run it.

Start the container with an execution limit of 4 hours by:

```
cstart -h 4 /software/remest_qiime_core_2022.8.sif
```

Start Qiime:

```
Singularity> qiime
```

Start Cutadapt:

```
Singularity> cutadapt
```

Start Gprename:

```
Singularity> gprename
```

remest_tensorflow_22.10.1-tf2-py3_gpu:

The Tensorflow container is an Nvidia accelerated container whit **python3** and **TensorFlow_v2.10** pre-installed. The `pip` command is also available within the container so you can download your preferred python libraries. The current version of Jupyter Notebook is also available to execute notebooks (*.pynb*) scripts.

Use `cstart` to start the container and access with 12 hours of running limits and access to one GPU; then start using python on the Singularity shell:

```
cstart -g 1 -h 12 /software/remest_tensorflow_22.10.1-tf2-py3_gpu.sif

… …

Singularity> python
```

To run Jupyter Notebook and mount your home inside of it use:

```
Singularity> jupyter notebook --notebook-dir=/home/<user>/  --port=8888

[I 17:16:59.705 NotebookApp] Jupyter Notebook 6.4.10 is running at:

[I 17:16:59.705 NotebookApp]
http://hostname:8888/?token=5c3f836a805fb41627d26200434b50dd613c210357f3678c

[I 17:16:59.705 NotebookApp] Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).
```

Once started you can access the notebook by opening the link reported on the shell, together with the generated token, by means of the browser. Notice that, you are not allowed to do port remapping between the container and the host machine so, to select a different port on which to launch Jupyter, you have to use the `--port` option on the jupyter launch command.

6. Get your own Singularity container

Obtaining the Image using the pull Command:

Some software is provided as a Singularity image with the .sif or .simg file extension. More commonly, however, a Docker image will be provided and this must be converted to a Singularity image. For instance, you can download and convert a Docker image to a Singularity image with:

```
$ singularity pull docker://hello-world
```

This will produce the file `hello-world_latest.sif` in the current working directory.

When looking for containerized software try these repositories:

- Docker Hub (https://hub.docker.com/)
- NVIDIA GPU Cloud (https://ngc.nvidia.com/catalog/containers)

- Singularity Cloud Library (https://cloud.sylabs.io/library)
- Singularity Hub (https://singularityhub.github.io/singularityhub-docs/)
- Quay.io (https://quay.io/)
- BioContainers (https://biocontainers.pro/)