MAX PLANCK
COMPUTING & DATA FACILITY
MPCDF

Bits & Bytes

No.211, December 2022

Computer Bulletin of the Max Planck Computing and Data Facility (MPCDF)*
https://docs.mpcdf.mpg.de/bnb

High-performance Computing

Anaconda Python modules

Starting early next year, users will need to specify an explicit version when loading the 'anaconda' Python module, similar to the compiler, MPI, and CUDA modules. This is supposed to avoid common issues due to changes of the default version of a module. In case you need to load the anaconda module for your jobs, please adapt your submit scripts already now and use

```
module load anaconda/3/2021.11
```

to load version 3/2021.11 explicitly, for example. This new behaviour will be enforced with an upcoming maintenance early 2023 which will be announced in due time.

Sebastian Ohlmann

Hints for architecture-specific and optimized CUDA compilation

Compute nodes at the MPCDF that are equipped with Nvidia GPUs typically run a long-term-supported (LTS) major version of the CUDA device driver. At the same time, more recent CUDA SDK versions are provided (e.g. `cuda/11.6`) that rely on the forward-compatibility feature[1] of the device driver. In case you are using the Nvidia CUDA compiler to compile source code, make sure to specify the correct target architecture via the `-arch` flag, e.g. `sm_80` for the Ampere (A100) GPUs on *Raven* or `sm_70` for the Volta (V100) GPUs on *Cobra*:

```
nvcc -arch sm_80 source_file.cu
```

The proper `-arch` flag makes the binary match the actual GPU architecture and avoids PTX-related errors that might occur with most recent CUDA versions on LTS drivers if the architecture is not specified. Moreover, this flag enables compiler optimizations specific to the target microarchitecture which may improve performance.

Klaus Reuter

New Intel C/C++ compilers and associated MPCDF software stack

Intel is replacing their current, so-called "classic" C and C++ compilers (icc and icpc, respectively) by new LLVM-based compilers (icx and icpx, respectively). The classic compilers were already deprecated and will be removed from new OneAPI releases in the course of the second half of 2023, as mentioned in the release notes[2]. The same transition is planned for the classic Fortran compiler (ifort) to be replaced by the new LLVM-based compiler (ifx), but only at a later time when the new compiler matches the functionality and performance of the classic one.

MPCDF will offer the new compilers and a full software stack compiled with icx, icpx and ifort on the HPC systems and clusters starting early next year. The corresponding intel modules are named with a version number ending with "x" (the first will be `intel/2022.2.1.x`). We recommend users to test and adopt this new software stack as soon as possible. In case of issues regarding functionality or performance, please let us know by opening a ticket in the helpdesk or via E-mail to support@mpcdf.mpg.de.

Sebastian Ohlmann

Turbomole license for MPG renewed

The existing licensing agreement of the MPG for the quantum-chemistry software package Turbomole[3] was recently renewed with the new owner of Turbomole, Dassault Systems. The agreement has been negotiated by the Max Planck Digital Library[4] in collaboration with MPCDF. MPCDF maintains latest versions of Turbomole on its HPC systems and provides the software for download by Max Planck Institutes on request. For obtaining Turbomole please open a ticket in the MPCDF helpdesk or send an E-mail to support@mpcdf.mpg.de.

Markus Rampp

*Editors: Dr. Renate Dohmen & Dr. Markus Rampp, MPCDF

[1] https://docs.nvidia.com/deploy/cuda-compatibility/index.html#deployment-consideration-forward
[2] https://www.intel.com/content/www/us/en/developer/articles/release-notes/oneapi-c-compiler-release-notes.html
[3] https://www.3ds.com/products-services/biovia/products/molecular-modeling-simulation/solvation-chemistry/turbomoler/
[4] https://www.soli.mpdl.mpg.de/en/

Apptainer on HPC clusters, the Linux Foundation successor to Singularity

Introduction

Apptainer[5] is an open-source, container virtualization software designed to execute software in a secure, portable and reproducible environment. Just like its predecessor Singularity, Apptainer has been developed with the idea of providing container technologies on HPC systems. The software, for example, gives users an easy way to use different operating systems on the HPC systems while still ensuring that containers run in an established user environment, without a pathway for privilege escalation on the host.

Apptainer was born in 2021, when the Singularity open-source project split into two distinct projects: Apptainer and SingularityCE. The Apptainer branch has joined the Linux Foundation, while the Sylabs' fork of Singularity, dedicated to commercial use, was renamed SingularityCE. While, at least at the beginning, there has been continual alignment between Sylabs' SingularityCE and Apptainer, over time the projects will likely diverge as both continue to mature and new features are included in the releases.

As part of the transition, only open community standard interfaces will be supported in Apptainer. This includes removing the "Library", the Sylabs repository (similar to DockerHub) where you can push your containers to or pull containers from, and the "Remote Builder" support (but see the "Notes on the Sylabs Cloud endpoint" below). In the event these become open community maintained standards (and not corporate controlled), these features might be re-added at a later date.

Apptainer at MPCDF

In its current version, Apptainer provides backwards compatibility offering `singularity` as a command line link. It is also committed to maintain as much of the command-line interface (CLI) and environment functionality available in the old Singularity software as possible. From the user's perspective, very little, if anything, should change and the wrapper around the `singularity` command allows users to run commands like `singularity pull`, `singularity run`, etc. just as before.

On the HPC clusters at MPCDF, an environment module is available in order to load the Apptainer software. For backward compatibility, a Singularity module (singularity/link2apptainer) is also provided and will print a warning message and load the default Apptainer module. Users are encouraged to switch from the old Singularity module to the new Apptainer one, adjusting their scripts as needed. Please, note that support for the old Singularity software has already been discontinued and the new SingularityCE software will not be supported on the HPC clusters at MPCDF.

Notes on the Sylabs Cloud endpoint

As mentioned above, Apptainer removed support for the old library remote endpoint provided by Sylabs Cloud. Practically, this means that commands like

```
apptainer pull library://lolcow
```

would fail with the error message "FATAL: Unable to get library client configuration: remote has no library client". The `apptainer remote` command group allows users to manage the service endpoints Apptainer will interact with for many common command flows. If access to the Sylabs Cloud remote is required, users can follow the instructions here[6] and run the commands

```
apptainer remote add --no-login SylabsCloud \
 cloud.sylabs.io
apptainer remote use SylabsCloud
```

in order to restore the library behaviour of the old Singularity software. It is possible to check the current list of remote endpoints using

```
apptainer remote list
```

Example of common Apptainer commands

In the following, we illustrate some of the most common Apptainer commands (based on version 1.0.3). For more information and a complete description of all the commands, see the Apptainer User Guide[7].

Help command:

```
apptainer help <command> [<subcommand>]
```

Note that this command also shows the available options for each apptainer command.

Pull an image (*lolcow* in the following examples) from an online repository:

```
# Pull from the Sylabs Cloud (see notes above)
apptainer pull ./lolcow.sif \
 library://lolcow

# Pull from the docker registry
apptainer pull ./lolcow.sif \
 docker://godlovedc/lolcow
```

Build locally an image from a recipe:

```
sudo apptainer build lolcow.sif lolcow.def
```

where the recipe file *lolcow.def* contains the following

```
Bootstrap: docker
From: ubuntu:16.04

%post
    apt-get -y update
    apt-get -y install cowsay lolcat

%environment
```

[5] https://apptainer.org
[6] https://apptainer.org/docs/user/1.0/endpoint.html#restoring-pre-apptainer-library-behavior
[7] https://www.apptainer.org/docs/

```
    export LC_ALL=C
    export PATH=/usr/games:$PATH

%runscript
    date | cowsay | lolcat
```

Important: the build command requires `sudo` just as installing software on your local machine requires root privileges. For this reason, users should build images from a recipe on their local machine (where they have sudo privileges) and then transfer the image to HPC clusters in order to run it.

Run the image *lolcow.sif*:

```
apptainer run lolcow.sif
```

This command will run the user-defined default command within a container (i.e. the section `%runscript` in the apptainer recipe file). Some useful flags supported by the `apptainer run` command are:

- `-C/--containall` this option ensures that not only file systems, but also PID, IPC, and environment are completely contained and separated from the user environment on the HPC cluster. In order to access files and directory on the host when this flag is used, you need to explicitly bind them (see the `-B/--bind` option below).
- `-B/--bind /path_outside_container/:/path_inside_container/` is used to bind a user-specified path on the host (path before the : symbol) to a path inside the container (path after the : symbol). Multiple bind paths can be provided using a comma-separated list.
- `--nv` is used in order to instruct the container's environment to use an Nvidia GPU and the basic CUDA libraries to run a CUDA enabled application. See here[8] for details about the GPU support in Apptainer.
- `--net --network=none --network-args "portmap=8080:80/tcp"` maps a port from inside the container to a different port on the host. Note that unprivileged users on the HPC clusters need to include the `--network=none` option.

For example:

In order to mount the `/ptmp/` folder available in the HPC cluster inside the container, so that your containerized code can access data or store results in your `/ptmp/$USER` space, you can use:

```
apptainer run --containall \
 --bind /ptmp/$USER/:/path_inside/ \
  lolcow.sif
```

where `/path_inside/` is the new path inside the container where the `/ptmp/` folder is mounted.

If your code needs access to the GPUs available on the compute node, you can run

```
apptainer run --nv lolcow.sif
```

In order to expose port 80 inside of the container and map it to port 8080 outside of the container, use

```
apptainer run --net --network=none \
 --network-args "portmap=8080:80/tcp" lolcow.sif
```

Execute a specific command (`whoami` in the example) in the container:

```
apptainer exec lolcow.sif whoami
```

Open a shell in the container:

```
apptainer shell lolcow.sif
```

Note that the flags presented above for the `apptainer run` command can also be used for the `apptainer exec` and the `apptainer shell` commands.

Michele Compostella

GitLab Tips & Tricks: Use of Docker Images in GitLab CI

Continous-integration (CI) pipelines are a well-adopted feature of the MPCDF GitLab instance, and many of our users are utilizing CI pipelines in their daily work. To execute these pipelines, MPCDF hosts several servers as so-called "shared GitLab runners" which means that every user of the MPCDF GitLab instance can execute pipelines on these servers. You can find more detailed information about our GitLab runners here[9].

For every defined job in a CI pipeline, a Docker container with a default Docker image is executed on one of the runners. This default Docker image is **python:3.9** on all shared Gitlab runners currently. For other programming languages or if the MPCDF module system is required, you can easily base your pipeline on a different image. The image can be taken from a public container registry like Docker Hub[10] or be a self-built image stored in MPCDF's GitLab instance (here[11] you can find fur-

[8] https://apptainer.org/docs/user/1.0/gpu.html
[9] https://docs.mpcdf.mpg.de/doc/data/gitlab/gitlabrunners.html
[10] https://hub.docker.com
[11] https://docs.gitlab.com/ee/user/packages/container_registry/

ther documentation on how to use the GitLab container registry).

You can specify the Docker image to be used for creating your Docker container by modifying your pipeline definition file `.gitlab-ci.yml`. Insert a line at the top of the file, so that it looks, e.g., like this:

```
image: gitlab-registry.mpcdf.mpg.de/mpcdf/module-image
```

This example would provide an environment that is aligned with the software stack provided by the MPCDF module system on HPC systems and clusters.

As another example, with

```
image: golang:latest
```

the latest Docker image with the environment for the Go programming language will be downloaded from Docker Hub, cached on our GitLab servers, and used as a basis for the container. To achieve reproducibility of the results, we recommend to explicitly specify a version instead of `latest`, because the latter might continuously be overwritten by newer images:

```
image: golang:1.19.1
```

Thomas Zastrow, Tobias Melson

GO-Nexus

This article presents GO-Nexus, an enhanced data transfer and sharing service for MPCDF's Nexus-Posix storage system based on Globus Online (GO).

Nexus-Posix is an IBM Spectrum Scale filesystem which is commonly used by projects in MPCDF's HPC-Cloud. Projects can rent a reservation on Nexus-Posix which can be scaled up as the project grows. The reservations are generally in the range of 10-100 TB and are accessed via mount points on the HPC system *Raven* and/or HPC-Cloud VMs. Until now external access to Nexus-Posix has required either project specific solutions or the use of tunneled SFTP connections. Both of these solutions have significant limitations which lead to extra project overheads, slow and possibly unreliable data transfers, custom-made solutions etc. GO-Nexus is designed to address this by providing a Globus connect server on top of Nexus-Posix.

Globus provides a fast, reliable and user-friendly way to transfer or share large amounts of data. Additionally, Globus can aid projects in publishing findable data for their communities. These qualities make Globus an ideal service to enable access to the large-scale data stored in Nexus-Posix. Combining these two services provides a solution for several core use cases and opens extra possibilities for projects which make use of Nexus-Posix.

Two of the primary use cases are highlighted in Fig. 1, namely: 1. Transfer and Sharing service with mount points on Raven and possibly HPC-Cloud VMs. 2. Standalone Transfer and Sharing service, for data collection, publishing and sharing.

The first use case highlights how projects can expose the Nexus-Posix filesystem mounted on the Raven HPC system and/or HPC-Cloud VMs, with users possibly performing large-scale simulations at MPCDF and then transferring results back to their home institute or even sharing them with colleagues in world-wide collaborations.

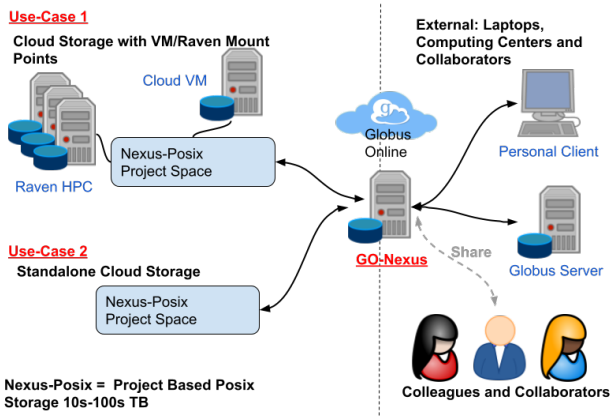

Figure 1: GO-Nexus Example Use-Cases

The second use case shows how standalone storage can be made globally available via GO-Nexus. This could be used when gathering data in the field for processing at a later date and/or for distributed collaborations where GO-Nexus would act as a central datastore, benefiting from the high-speed network connection at MPCDF.

In both cases the reservations can be exposed either as findable or as private data collections via Globus, where users and community members can easily search for the data via the Globus web portal.

In addition to the reliable transfer capabilities GO-Nexus benefits from all the advanced functionality which is available via the MPCDF's globus subscription, enabling actions such as sharing and the use of Globus flows for automation. Several cloud projects have already adopted GO-Nexus for large-scale data transfers and to regularly sync data to and from Nexus-Posix by using "Globus timers", a cron like service offered through the Globus web portal.

John Alan Kennedy

News & Events

Discontinuation of General VPN

MPCDF has been offering a virtual private network (VPN) named "General VPN" for all its users, which allowed the access to the MPCDF network from everywhere. For security reasons this VPN will be discontinued as of February 1, 2023. Users are recommended to employ the VPN of their home institute in order to access the MPG network and to use the MPCDF gateway machines[12] for login or for tunnelling connections to MPCDF services.

Andreas Schott

Meet MPCDF

The monthly seminar series "Meet MPCDF" which was newly launched earlier this year, is receiving great interest by many MPCDF users. So far, the series covered "AI services at the MPCDF" (June), "CMake for HPC" (July), "MPCDF Gitlab Features" (September), "Data Transfer and Sharing" (October), "Going Public with your Code" (November) and "Trends in high-performance computing" (December). The corresponding announcements and slides can be found on the MPCDF training portal[13]. We gratefully acknowledge the attention of so many participants and look forward to continuing the series with talks and discussions next year. After a christmas break the series continues on Thursday, February 2, with a talk on basic debugging tools and strategies. Further topics envisaged for subsequent events (first Thursday of the month, at 15:30) include containers for HPC, strategies for testing software, and the Jupyter ecosystem.
Our users are particularly welcome to suggest topics of their interest and to raise discussions and specific requests to MPCDF during the seminar. Registration is not required, the zoom link is distributed in advance via our all-users mailing list and is also posted on the MPCDF website.

Python for HPC

MPCDF held another issue of its well-established online course on "*Python for HPC*" from November 15 to November 17, which was attended by around 130 participants from several institutes. This annual workshop consists of lectures in the morning and exercise sessions in the afternoon and teaches how to combine the advantages of python for quickly developing new code with techniques for achieving good performance on HPC systems. The next issue is planned for November 2023.

Advanced HPC Workshop

From November 22 to 24, MPCDF organized the annual "Advanced HPC workshop" with talks from experts from MPCDF, Intel and Nvidia about profiling, debugging and porting HPC codes. The material of the talks can be found on the MPCDF training portal[14].

Tilman Dannert

[12] https://docs.mpcdf.mpg.de/doc/computing/gateways.html?highlight=tunnel#gateway-machines
[13] https://www.mpcdf.mpg.de/services/training
[14] https://www.mpcdf.mpg.de/services/training