MAX PLANCK
COMPUTING & DATA FACILITY
MPCDF

Bits & Bytes

No.213, August 2023

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

High-performance Computing

New GPU development partition on *Raven*

A gpu development partition "gpudev" was created on the set of GPU nodes on *Raven* in order to facilitate development and testing of GPU codes. In order to use the "gpudev" partition you have to specify

```
#SBATCH --partition=gpudev
```

in your submit script. The maximum number of nodes available with "gpudev" is 1, the maximum execution time is 15 minutes, and you can choose to use between one and four Nvidia A100 GPUs like for usual GPU jobs.

Renate Dohmen, Mykola Petrov

Memory profiling with heaptrack

The memory profiler heaptrack has recently been installed on *Raven*. It can be used to measure memory usage, find memory allocation hotspots and memory leaks in a C, C++, or Fortran code. Heaptrack traces the memory allocation size and frequency as well as the call stack. More information about heaptrack can be found here[1].

To use heaptrack, you first have to collect the memory profiling data while running your executable. There is no need for recompiling (instrumenting) your executable. Simply run it through `heaptrack` in your SLURM job script as shown below (assuming your executable is named `a.out`):

```
module load heaptrack
srun hpcmd_suspend heaptrack ./a.out
```

A `.gz` file containing the profiling data will be generated in your job submission directory. To view a short analysis of it in the console, run `heaptrack --analyze` on that file. You also have the option to download the data file to your workstation and analyze it locally. This is especially useful if you install the graphical interface for heaptrack which may be possible via your system's package manager. Note that the version of heaptrack used for data collection and analysis should match in this case. Heaptrack does not natively support MPI. However, it can analyze MPI codes and generate a separate output file for each MPI rank.

Tobias Melson

New compilers and libraries: Intel oneAPI 2023.1

Intel oneAPI 2023.1 has been made available on *Raven*, *Cobra*, and other clusters. It provides the compiler module `intel/2023.1.0.x`, the MPI module `impi/2021.9`, the MKL module `mkl/2023.1`, and the corresponding Intel profiling tools. Also the software stack on these clusters has been compiled with this toolchain.

With this oneAPI version, MPCDF follows Intel's recommendation to set the new LLVM-based C and C++ compilers, `icx` and `icpx`, respectively, as the default. They replace the deprecated "classic" compilers `icc` and `icpc`. With this transition, the corresponding MPI wrappers are called `mpiicx` and `mpiicpx`, respectively. The "classic" Fortran compiler `ifort` is still the default, however, the new LLVM-based Fortran compiler `ifx` and its MPI wrapper `mpiifx` are already available and can be tested thoroughly. We propose compiling your Fortran code with both `ifort` and `ifx` to compare performance and unravel possible issues. Please contact us via the MPCDF Helpdesk[2] in case you encounter compiler-related problems .

Tobias Melson

*Editors: Dr. Renate Dohmen & Dr. Markus Rampp, MPCDF
[1] https://github.com/KDE/heaptrack
[2] https://helpdesk.mpcdf.mpg.de/

Using linters to improve and maintain code quality

Linters are static code analyzers that are commonly used for detecting programming errors, code smells[3], compatibility issues, stylistic errors, etc. Some linters also have automatic patch generation or in-place code fixing capabilities. The term linter originates from S. C. Johnson's Lint[4] tool for C source code, released in 1978. In the example section, you can see a short list of open-source linters for various syntaxes. Modern compilers can also be used as linters by activating warnings[5]. Using external linters is a part of OpenSSF best practices criteria[6]. As an example, here's a typical output from pylint[7]:

```
> pylint not_bad.py
************* Module not_bad
not_bad.py:1:0: C0114: Missing module docstring (missing-module-docstring)
not_bad.py:6:0: E0401: Unable to import 'wrongpy' (import-error)
not_bad.py:9:0: C0116: Missing function or method docstring (missing-function-docstring)
not_bad.py:10:4: W0621: Redefining name 'timestep' from outer scope (line 32) (redefinedouter-name)
not_bad.py:18:13: W1514: Using open without explicitly specifying an encoding (unspecifiedencoding)
-----------------------------------
Your code has been rated at 6.40/10
```

Depending on the syntax of your code, you may have multiple options when choosing a linter for your project. For example, in a python project, you can use `isort` and `autoflake` for cleaning up imports, `vulture` for finding dead code, `bandit` for checking security issues, `pyroma` for checking packaging, and `black` for formatting. Your project's size and complexity are also important factors in choosing linters. If your project is large, `ruff` together with a type checker such as `mypy` can be much faster than pylint (all tools can be obtained from pypi.org[8]).

Linters are usually designed to be very flexible and easily controllable using a configuration file. You must keep your linter's configuration together with your code under version control.

One can manually run the linters to detect the issues and fix them. However, to ensure code quality, these linters must be integrated into your workflow to be triggered automatically. This is commonly done by adding them to the build system, integrating them into IDEs, or using git pre-commit[9]. Specialized runners such as lintrunner[10] can simplify setting up linters for complex projects.

To maintain code quality, linters must be added to automated tests, e.g., in CI pipelines. The output of the linters can also be integrated into git forges such as GitLab[11] and inspected in each merge request. As an example, you can see pylint integration in the Code Quality section of this merge request[12].

The online version of this article[13] provides a table with a number of popular open-source linters and online references to all tools as a starting point for your projects.

Meisam Farzalipour Tabriz

HPC-Cloud Object Storage

On July 1st, 2023 a new Petabyte-scale Object Storage system was commissioned at MPCDF as part of the HPC-Cloud. The Object Storage system is based upon CEPH, a software-defined storage solution, and comprises 11 servers with a total of 11 PiB storage.

One of the major advantages of the Object Storage system is that it offers global access, i.e. it may be accessed from MPCDF clusters and servers at Max Planck Institutes as well as user desktops/laptops (see Figure 1).

[3] https://en.wikipedia.org/wiki/Code_smell
[4] https://citeseerx.ist.psu.edu/doc/10.1.1.56.1841
[5] https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html
[6] https://bestpractices.coreinfrastructure.org/en/criteria/0?details=true&rationale=true#static_analysis
[7] https://pylint.readthedocs.io/en/latest/
[8] https://pypi.org/
[9] https://pre-commit.com
[10] https://pypi.org/project/lintrunner/
[11] https://docs.gitlab.com/ee/ci/testing/code_quality.html
[12] https://gitlab.mpcdf.mpg.de/mpcdf/training/pylint/-/merge_requests/2
[13] https://docs.mpcdf.mpg.de/bnb/213.html

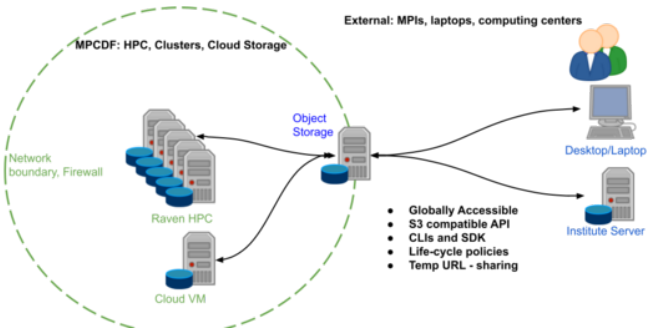

Figure 1: HPC Cloud Object Storage

In addition to global data access other benefits include S3 compatible API, life-cycle policies, temporary data sharing via temp URLs, multiple clients and a flexible python SDK. The SDK allows the storage to be accessed and managed directly from within applications, opening numerous possibilities for projects to automate and encapsulate data access in their workflows and services.

The Object Storage system complements existing storage solutions at MPCDF offering an alternative to standard POSIX-based storage systems. Two possible data storage classes exist, replication and erasure coding, which allow the underlying data storage to be tuned to best suit a project's use-cases.

An example use-case would be using the storage as an output sink for batch-based data production: Temporary access keys could be generated for the batch processing, these may be revoked after the batch processing. Then a set of batch jobs could be run with results being PUT into the object storage. Post processing jobs, at remote MPI clusters and/or desktops, could then GET the data for processing. Final results could again be PUT into object storage and could also be shared with collaborators via temporary access URLs if required.

The Object Storage can be rented by projects in the range of 10s-100s of TB and is primarily designed to support scientific datasets with objects in the multi-MB range with a PUT/GET access pattern. The storage can be used together with cloud compute services or stand-alone as a storage silo. More information about the HPC-Cloud and the rental model can be found here[14].

John Alan Kennedy, Florian Kaiser, Robert Hish

JADE - Automated Slurm deployments in the HPC-Cloud

A growing number of HPC-Cloud projects are deploying complex systems in the cloud to support various use-cases. One such system is a Slurm cluster, either as a backend for a service or to help better utilise cloud resources. To address this need an example solution has been created in the form of JADE. JADE uses Infrastructure as Code solutions including Terraform, Packer and Ansible to provide an automated deployment of Slurm clusters within the MPCDF HPC-Cloud.

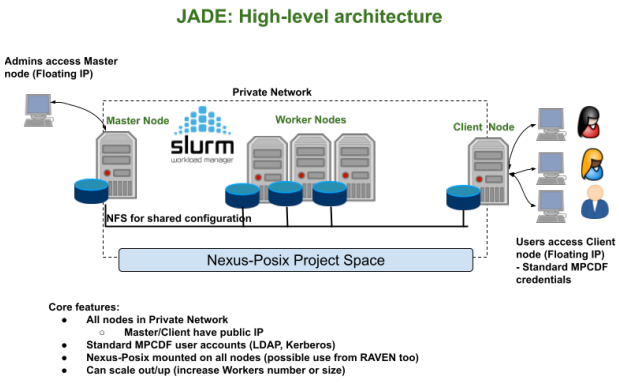

Figure 2: High-level JADE architecture

Terraform allows a JADE deployment to be managed as a complete stack, ensuring that a cluster can be reliably deployed as a whole, but also can be reliably deleted (removing all dependencies). This helps to maintain clean deployments within a cloud project and avoids possible wastes of resources. Packer and Ansible allow JADE images to be generated in a reproducible and well-understood manner. This is a cloud best practice that can be utilised in many other projects.

JADE aims to be simple to deploy, elastic, i.e. it can scale in and out (scale the number of workers) and ephemeral, i.e. a cluster can be destroyed and re-created without loss of persistent data

This is ideal for projects which require small to medium Slurm cluster deployments or which wish to evaluate deploying a batch system within the cloud either for testing or to better utilise cloud resources. Using JADE a Slurm cluster can be deployed within approximately 10 minutes. The standard deployment (see Figure 2) provides a Slurm master and worker nodes, which only cluster admins can log in to, and a UI node for users to submit batch jobs. Cluster admins have complete freedom w.r.t. software installations and can flavour the UI and worker nodes to suit the user communities (a popular choice is the deployment of cluster specific software via modules in a similar fashion to the MPCDF clusters and HPC systems).

[14] https://docs.mpcdf.mpg.de/doc/cloud/index.html

The integration of MPCDF user management systems and Nexus-Posix allows JADE to provide users with an experience similar to standard cluster deployments at MPCDF. Moreover, since Nexus-Posix can be mounted on both Cloud resources and Raven, a hybrid solution can use *Raven* for large-scale HPC processing and a JADE-based cluster for long-running post-processing jobs, sharing the data via Nexus-Posix. In addition to using JADE as a solution projects can use the Terraform-based deployment as an example to help understand how other complex services can be deployed within the HPC-Cloud. Further information about JADE can be found in our gitlab repository[15].

John Alan Kennedy

GitLab: Tips & Tricks

Online editing of source code revisited

Five years ago, in 2018, Bits&Bytes published an article about GitLab's integrated Web IDE[16] and how it can be used to edit code online. Today, GitLab contains a completely different Web IDE which is based on MS Visual Code. If you are already familiar with MS Visual Code on your desktop, you will now find the same behaviour and user experience directly integrated into GitLab. In future versions, GitLab's Web IDE will also support the native MS Visual Code plugins, which can be used to enhance the IDE's functionality in many ways. From any GitLab repository, you can reach the Web IDE via the button "Web IDE" on the repository's start page or from any open file.

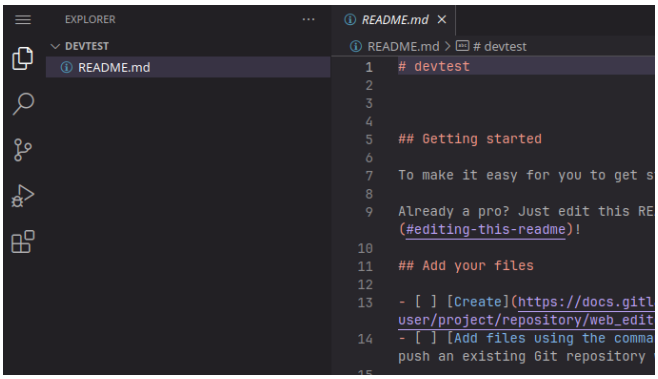

Figure 3: GitLab's new Web IDE

Custom badges

GitLab offers badges[17] to display short pieces of information in a graphical way. Badges are small images, displayed under the header of a GitLab project or group. By default, GitLab supports badges to display information about the current status of the CI Pipeline, test coverage and the latest release.

Figure 4: Pipeline Badge

With custom badges, it is also possible to upload any image and use it as a badge. You can find the badge settings under "Settings / General / Badges" in any GitLab repository. Here, you can create individual badges from uploaded images; for example, upload a logo and use it as eye catcher for your repository:

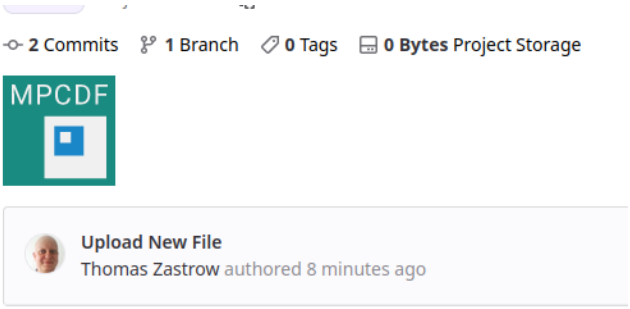

Figure 5: Custom Badge

Security warning

GitLab's image registry is a convenient way of managing Docker images. You can upload and tag self-created Docker images, use them in Continous Integration Pipelines or make them accessible from outside GitLab. To work properly, the software inside the images needs sometimes user credentials, access tokens or other secret information to access services or data somewhere else. Storing these credentials in a Docker image which is publicly available can be a high security risk. In a recently published article by M. Dahlmanns et al.[18], the authors found thousands of private credentials stored in publicly available Docker images. If you store self-created Docker images in GitLab's image registry and make them publicly available, please make sure that none of your usernames, passwords or other access tokens are stored inside the image!

Thomas Zastrow

[15] https://gitlab.mpcdf.mpg.de/mpcdf/cloud/jade
[16] https://docs.mpcdf.mpg.de/bnb/pdf/bits_and_bytes_issue_199.pdf
[17] https://gitlab.mpcdf.mpg.de/help/user/project/badges
[18] https://arxiv.org/abs/2307.03958

New IBM tape library and tape drives installed at MPCDF

This year, the two Oracle SL8500 tape libraries at MPCDF will be taken out of service. To replace them and also enhance capacity and performance, one of the existing IBM TS4500 tape libraries has been expanded. Additionally, a new IBM TS4500 library has been installed, along with a total of 96 new LTO9 tape drives.

The initial installation of the Oracle SL8500 tape library took place at MPCDF back in 2006, more than 17 years ago. Subsequently, it underwent multiple expansion phases, reaching a capacity of 20,000 tapes. Throughout the years, various generations of tapes and tape drives have been utilized, ranging from LTO-3 with a native capacity of 400 GB to LTO-8 with a native capacity of 12 TB. This tape library has served for many years as the primary storage location for backup and archive data generated by users of different Max Planck Institutes, ensuring its retention over an extended period. Due to the library model's discontinuation by Oracle and its prolonged period of operation, a decision was made to seek a suitable replacement.

In 2013, MPCDF (formerly RZG) installed an IBM TS3500 tape library at the Leibniz computing centre (LRZ) with the purpose of storing a second copy of long-term archive data. In the following years, to accommodate the continuously growing volume of data, three additional IBM TS4500 tape libraries were installed at both MPCDF and LRZ. These additional installations were essential in meeting the expanding data storage demands. Now, the old Oracle library gets replaced by expanding one of the existing IBM TS4500 tape libraries with approximately 10,000 tape slots and procuring an additional IBM TS4500 library to further meet growing storage requirements. Furthermore, a total of 96 of latest generation LTO-9 tape drives have been installed across all IBM libraries.

Here are some features of the IBM TS4500 tape library model:

- One base frame and up to 17 expansion frames with a total capacity of over 22,000 LTO tapes per library.
- Up to 128 tape drives per library.
- Dual robotic accessors.
- Automatic control-path and data-path failover.
- Support for multiple logical libraries.
- Tape-drive encryption and WORM media support.
- Persistent worldwide names, multipath architecture, drive/media exception reporting, remote drive/media management.

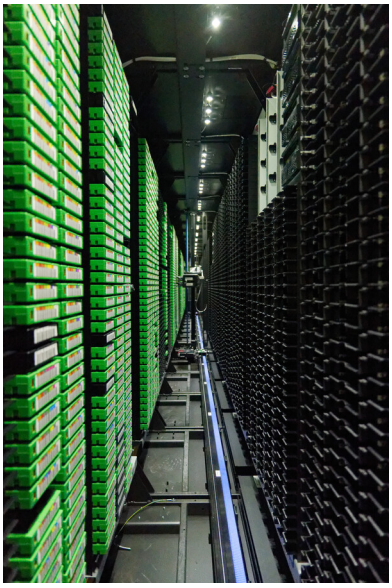

Figure 6: IBM TS4500 tape library

After the installation of the new systems, the current MPCDF tape storage infrastructure looks like this:

- 5 IBM TS4500 tape libraries (3 at MPCDF + 2 at LRZ) with a total capacity of more than 105,000 LTO tape slots, of which about 60,000 are currently in use.
- 200 LTO tape drives, of which 106 are latest-generation LTO-9 tape drives, while the remaining drives consist mostly of LTO-8. A small number of older LTO-7 and LTO-6 drives are still operational.
- The tape drives are all integrated in two Fibre Channel Storage Area Networks (SANs), one at MPCDF and another one at LRZ. These SANs employ 16 and 32 Gb/s capable Broadcom Fibre Channel switches laid out in a multiple-path meshed topology. Over 30 server machines (IBM Spectrum Protect and HPSS servers) have access to these tape SANs to store data on tape.
- Currently, the total data stored on tape, comprising backups in Spectrum Protect and long-term archives in HPSS, amounts to approximately 330 Petabytes.

Manuel Panea Doblado

News & Events

Open positions at MPCDF

The MPCDF currently has three open positions in the Systems, Basic IT-services, and HPC application support division, respectively. Specifically, we are looking for:

- Systems expert - design and operation of complex cloud and storage infrastructures for scientific applications
- Computer Scientist - system and user management development
- HPC application expert - development and optimization of new methods in the electronic-structure software package Octopus

For details and directions to apply, please visit the MPCDF career webpage[19].

Markus Rampp

Meet MPCDF

Our monthly online seminar series "Meet MPCDF" has developed to a well-attended and valued training event. On every first Thursday of the month at 15:30 you can participate in an online seminar with a talk usually given by a member of the MPCDF and subsequent discussion. All material can later be found on our training webpage[20]. The schedule of upcoming talks is:

- September, 7th: The MPCDF Metadata Tools
- October, 5th: To be decided
- November, 2nd: To be decided
- December, 7th: Introduction to the new HPC system Viper

We encourage our users to propose further topics of their interest, e.g. in the fields of high-performance computing, data management, artificial intelligence or high-performance data analytics. Please send an E-mail to training@mpcdf.mpg.de[21].

Tilman Dannert

AMD-GPU development workshop

In preparation for the new supercomputer of the MPG with AMD MI300A GPUs in 2024[22], the MPCDF in collaboration with AMD and Atos offers an online course on AMD Instinct GPU architecture and the corresponding ROCm software ecosystem, including the tools to develop or port HPC or AI applications to AMD GPUs. The workshop will be held as an online event, spanning three afternoons on **November 28-30, 2023**, please save the date. Further details including the agenda and a registration link will be published on the MPCDF training website[23] in due course.

Tilman Dannert, Markus Rampp

[19] https://www.mpcdf.mpg.de/career
[20] https://www.mpcdf.mpg.de/services/training
[21] mailto:training@mpcdf.mpg.de
[22] https://docs.mpcdf.mpg.de/bnb/212.html#new-supercomputer-of-the-mpg-cobra-successor
[23] https://www.mpcdf.mpg.de/services/training