No. 202
December 2019

Bits & Bytes

Max Planck Computing & Data Facility (MPCDF)*
Gießenbachstraße 2, D-85748 Garching bei München

High-performance Computing

Michele Compostella, Lorenz Hüdepohl, Sebastian Ohlmann, Markus Rampp, Klaus Reuter, Luka Stanisic, Ingeborg Weidl

Software upgrades on Cobra and Draco

On the HPC cluster Cobra and on the Draco cluster, some major software upgrades were done in October/November 2019. Along with the operating system upgrade a completely new set of software modules was deployed. With this, we provide updated versions of the Intel Compiler, MPI and MKL, which are also now set as the default:

Cobra: Intel Compiler 19.0.4, MPI 2019.4, MKL 2019.4

Draco: Intel Compiler 18.0.5, MPI 2018.4, MKL 2018.4

The deprecated compatibility modules intel/17.0 and intel/18.0 were finally removed. The SLURM batch system was upgraded from version 17.11 to 18.08 on both clusters.

RVS on Cobra

The MPCDF has recently extended the web-based Remote Visualization Service to Cobra. HPC users can now select the "COBRA" machine in the submission page at https://rvs.mpcdf.mpg.de and run their remote visualization session on a dedicated visualization node of the Cobra cluster.

Each remote visualization session has access to all visualization software and data stored on the HPC systems. A remote session on Cobra uses half of a visualization node (i. e. a single RTX5000 GPU, 20 cores and 92 GB of main memory) and can extend up to a maximum running time of 24 hours. All steps required to initialize and submit a remote visualization session on the Cobra cluster via the web interface are described at the remote-visualization-service webpage and are the same as needed for the Draco cluster.

We remind users who require access to the OpenGL libraries for rendering purposes, that, in order to use the OpenGL libraries in your remote visualization session, the visualization command should be prefixed by "vglrun" (e. g. "vglrun visit", "vglrun paraview", "vglrun python", etc.). This will prevent error messages stating the GLX extension is missing from the display of the host.

Module environment

Since the start of their usage on Cobra last year, hierarchical environment modules have also been installed on Draco and several dedicated clusters. You can read more about hierarchical environment modules on the modules web page. However, the fact that not all modules are directly visible upon login, created minor problems for a few users. Therefore, the MPCDF decided to address this issue with the following tool.

In case you know the name of the module you wish to load, but you are not sure about the available versions or what dependencies need to be loaded first, you can try to use the 'find-module' command. This tool searches for the MODULENAME string through a list of all installed modules

```
~> find-module MODULENAME
```

You can then choose the desired module version, use the output of the command to determine the correct order to load dependencies, and finally load the module itself, e. g.

```
~> find-module horovod
horovod/cpu/0.13.11     (after loading anaconda/3/2019.03 tensorflow/cpu/1.14.0)
horovod/cpu/0.15.2      (after loading anaconda/3/2019.03 tensorflow/cpu/1.14.0)
horovod/gpu/0.13.11     (after loading anaconda/3/2019.03 tensorflow/gpu/1.14.0)
horovod/gpu/0.15.2      (after loading anaconda/3/2019.03 tensorflow/gpu/1.14.0)
~> module load anaconda/3/2019.03 tensorflow/cpu/1.14.0 horovod/cpu/0.13.11
```

*Tel.: +49(89) 3299-01, e-mail: benutzerberatung@mpcdf.mpg.de, URL: https://www.mpcdf.mpg.de/
Editor: Renate Dohmen

GPU profiling tools

Nvidia has recently introduced a new suite of tools to profile applications running on Nvidia GPUs, the Nsight tools (https://developer.nvidia.com/tools-overview). These tools supersede nvprof which is discontinued. The two most interesting tools for profiling HPC applications are "Nsight systems" and "Nsight compute", which are available as modules on our HPC systems.

"Nsight systems" is a tool that helps in understanding and optimizing the workflow of the full application. It offers a timeline view that precisely shows kernel launches, memory transfers, stack traces at certain points, and more information. For the kernel launches, e. g., the duration and launch parameters can be viewed; for the memory transfers, the size, speed, direction, and duration of the transfers are available. Moreover, "Nsight systems" supports NVTX (Nvidia tools extension), an API for defining ranges and events in the application code which facilitates associating the timeline to certain ranges in the source code. This also allows one to easily see which parts of the code run on the GPU and which parts are still executed on the CPU.

An application can be profiled by using

```
~> module load cuda nsight_systems
~> nsys profile -t cuda,nvtx srun ./application
```

which generates a file named 'report1.qdrep' that can be opened in the GUI (to be started with 'nsight-sys'). It is advised to profile only short runs (a few minutes maximum) to avoid generating trace files which are too large.

"Nsight compute" is a tool which allows much more detailed profiling of kernels. It provides very detailed information on the kernels, such as the utilization of the streaming multiprocessors, the memory bandwidth, and overview of the memory hierarchy and caching, thread scheduling, and much more.

An application can be profiled by using

```
~> module load cuda nsight_compute
~> nv-nsight-cu-cli --kernel-id \
      ::kernel_name:2 -o output ./app
```

which will profile the second invocation of the kernel with the name 'kernel_name'. This creates a file named 'output.nsight-cuprof-report' that can be opened in the GUI (to be started with 'nv-nsight-cu'). Because "Nsight compute" runs each kernel to be profiled several times, one should limit the profiling to a small number of representative kernel launches.

Python on HPC systems

Klaus Reuter

Introduction

In addition to the traditional HPC applications compiled from C/C++ and Fortran code we're witnessing an increasing workload of Python-based code running on the HPC systems. Being an interpreted and dynamically-typed language, plain Python is not a language suitable per se to achieve high performance. Nevertheless, with the appropriate packages, tools, and techniques the Python programming language can be used to perform numerical computation in a very efficient manner, covering both aspects, the program's efficiency and the programmer's efficiency. The aim of this article is to provide some advice and orientation to the reader in order to use Python correctly on the HPC systems and to take first steps towards basic Python code optimization.

Performance

The key to achieve good performance with Python is to move expensive computation from the interpreted code layer down to a compiled layer which may consist of compiled libraries, code written and compiled by the user, or just-in-time compiled code. Below, three packages are discussed for such use cases.

NumPy

NumPy is the Python module that provides arrays of native datatypes (float32, float64, int64, etc.) and mathematical operations and functions on them. Typically, mathematical equations (in particular, vector and matrix arithmetic) can be written with NumPy expressions in a very readable and elegant way, which has several advantages: NumPy expressions avoid explicit, slow loops in Python. In addition, NumPy uses compiled code and optimized mathematical libraries internally, e. g. Intel MKL on MPCDF systems, which enables vectorization and other optimizations. Parts of these libraries use thread-parallelization in a very efficient way by default, e. g. to perform matrix multiplications. In summary, NumPy provides the de-facto standard for numerical array-based computations and serves as the basis for a multitude of additional packages.

Cython

Cython is a Python language extension that makes it relatively easy to create compiled Python modules written in Cython, C or C++. It integrates well with NumPy arrays and can be used to implement time-critical parts of an algorithm. Moreover, Cython is very useful to create interfaces to C or C++ code, such as legacy libraries or native CUDA code. Technically, the Cython source code is translated by the Cython compiler to intermediate C code which is then compiled to machine code by a regular C compiler like GCC or ICC.

Numba

Numba is a just-in-time compiler based on the LLVM framework. It compiles Python functions at runtime for the datatypes these functions are being called with. Moreover, Numba implements a subset of NumPy's functions, i. e. it is able to compile NumPy expressions. Functions are declared via a simple decorator syntax to be suitable for jit-compilation, hence, Numba is minimally intrusive.

Parallelization

While Python does implement threads as part of the standard library, these cannot be used to accelerate computation on more than one core in parallel because the standard cPython implementation serializes the execution of Python byte code. A global interpreter lock is used to ensure that only one instruction can be executed at a time from all threads belonging to a Python process. Nevertheless, Python is suitable for parallel computation. In the following, two important packages for intra-node and inter-node parallelism are addressed.

multiprocessing

The multiprocessing package is part of the Python standard library. It implements building blocks such as pools of workers and communication queues that can be used to parallelize data-parallel workloads. Technically, multiprocessing forks subprocesses from the main Python process that can run in parallel on multiple cores of a shared-memory machine. Note that some overhead is associated with the inter-process communication. It is, however, possible to access shared memory from several processes simultaneously. A typical use case are large NumPy arrays.

mpi4py

Access to the Message Passing Interface (MPI) is available via the module mpi4py. It enables parallel computation on distributed-memory computers where the processes communicate via messages. In particular, the mpi4py package supports the communication of NumPy arrays without additional overhead. On MPCDF systems, the environment module mpi4py provides an optimized build based on the default Intel MPI library.

I/O

NumPy implements efficient binary I/O for array data that is useful, e. g., for temporary files. A better choice with respect to portability and long-term compatibility are HDF5 files. HDF5 is accessible via the h5py Python package and offers an easy-to-use dictionary-style interface. For parallel codes, a special build of h5py with support for MPI-parallel I/O is provided via the environment module h5py-mpi.

The Python software ecosystem

In addition to the packages discussed above, there is a plethora of well-established packages for scientific computation and data science available, covering, e. g., numerical libraries (SciPy), visualization (matplotlib, seaborn), data analysis (pandas), and machine learning (TensorFlow, pytorch), to name only a few.

Software installation

Often, users need to install special Python packages for their scientific domain. In most cases, the easiest and quickest way is to create an installation local to the user's home directory. After loading the Anaconda environment module, the command "pip install --user PACKAGE_NAME" would download and install a package from the Python package index (PyPI), or similarly, the command "python setup.py install --user" would install a package from an unpacked source tarball. In both cases, the resulting installation is located below "~/.local" where Python will find it by default.

Summary

The software recommended in this article is available via the Anaconda Python Distribution (environment module "anaconda/3") on MPCDF systems. Note that for some packages (mpi4py, h5py-mpi), the hierarchical environment modules matter, i. e., it is necessary to load a compiler (gcc, intel) and an MPI module (impi) in addition to Anaconda in order to get access to these depending environment modules. Some examples of SLURM job scripts are available at https://www.mpcdf.mpg.de/services/computing/software/languages-1/python.

The application group at the MPCDF has developed an in-depth course on "Python for HPC" which covers all the topics touched in this article in more detail on two days. It is taught one to two times per year and announced via the MPCDF web page.

Finally, it should be pointed out that Python 2 reaches its official end-of-life on January 1, 2020. Consequently, new Python modules and updates to existing ones will not take Python 2 compatibility into account in the future. Users who are still running legacy code are strongly encouraged to migrate to Python 3.

Cluster hosting

Hermann Lederer

The MPCDF traditionally hosts central Max Planck supercomputers and data infrastructures. In 2015, when the RZG was renamed to MPCDF, midrange-cluster hosting became a further official mission. Until 2000, only a few systems from Garching institutes (IPP, MPA and MPE) had been hosted at the RZG. In 2002, when the new machine room became available, cluster hosting at the RZG also started for external institutes, the first ones being the Fritz Haber Institute in Berlin, and the MPI for Polymer Research in Mainz, together with other Garching MPIs. Since 2007, when the old machine room was refurbished, more and more institutes started to host their servers at the RZG and to replace them on-site after typically about five years of operation. In the meantime, more than 25 Max Planck institutes have their compute or data systems hosted at the MPCDF, with a total power consumption of around 1.5 MW. Key advantages of hosting clusters at the MPCDF include: the availability of groundwater cooling, the proximity to the Max Planck supercomputers for pre and postprocessing, the mass storage system for long-term archival, high-bandwidth (WAN) network connections, and the availability of a comprehensive, unified and maintained software stack as well as application support.

The MPCDF SelfService: Guest-User and Self-Management for GitLab and DataShare

Amazigh Zerzour, Thomas Zastrow, John Alan Kennedy

A new self and guest-administration platform

The newly introduced SelfService platform lets MPCDF users easily manage their subscriptions to MPCDF services and allows them to invite external guests to those services (please note: you need a regular MPCDF account before you can login to the MPCDF SelfService). The SelfService thereby replaces the previous subscription service for administrating guest accounts (subscriptions.rzg.mpg.de). A redesign of the previous system was undertaken to give users more control over their accounts. In particular, the goals were to enable users to subscribe to specific MPCDF services and to introduce a more fine-grained privilege management for guest accounts.

The new platform extends the functionality of the subscription service by multiple aspects:

- Users can grant themselves access to the MPCDF services GitLab and DataShare (more to come).
- Access for guests can be granted for each service individually.
- Access for guests can be withdrawn entirely or per service.

The SelfService thereby makes the services offered by the MPCDF more accessible to all MPCDF users and gives users the chance to specify exactly which services they would like to use. Additionally, guest-account management is now more flexible since access can be granted and revoked for each service individually. The platform can be found at `https://selfservice.mpcdf.mpg.de`.

Self-administration for MPCDF users

The services that users can subscribe themselves to currently include GitLab (code versioning) and DataShare (online data storage). After logging in with the credentials of their "Primary MPCDF account/Erstkennung" users can use the navigation bar at the top of the page to navigate to *My Account* ⟩ *Services* (Fig. 1) and see which services they are currently using and which they can additionally subscribe to. **Note:** It may take up to 20 minutes until users can log in to a service after subscribing to it on the SelfService. At the bottom of the page users can see which service attributes are assigned to them. Attributes are service-specific settings such as the storage quota for DataShare which can only be altered by the MPCDF support. It is not possible to unsubscribe oneself from a service since this would generally lead to orphaned user data. If users would like to stop using a service they need to contact MPCDF support to decide what should happen with their data.

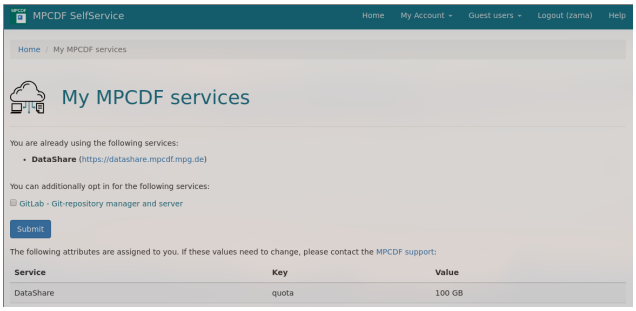

Figure 1: Managing subscribed services

Additionally, users can view their data such as telephone or room number via *My Account* ⟩ *My Data*. Currently the data shown here is read-only.

Guest management

Users can invite persons who have no MPG account to collaborate with them via one or more of the services provided by the MPCDF. However, please note, that people who already have an MPG account cannot be invited as guests and should apply for their own MPCDF account instead. A new guest may be invited via *Guest users* ⟩ *New invitation* (Fig. 2). Invitors can provide the full name and e-mail address of the guest. They may also write a short message to the guest and specify why the guest was invited (such as a project name or contract id). The guest then has seven days to use the link in the invitation e-mail to register at the SelfService. The inviter is informed by e-mail as soon as this happens. Guest accounts are valid for two years by default. Guests who were granted access to DataShare have no quota of their own; they use the storage quota of the users which share links with them. Please remember, that when you share a link with a guest, they will use your data quota.

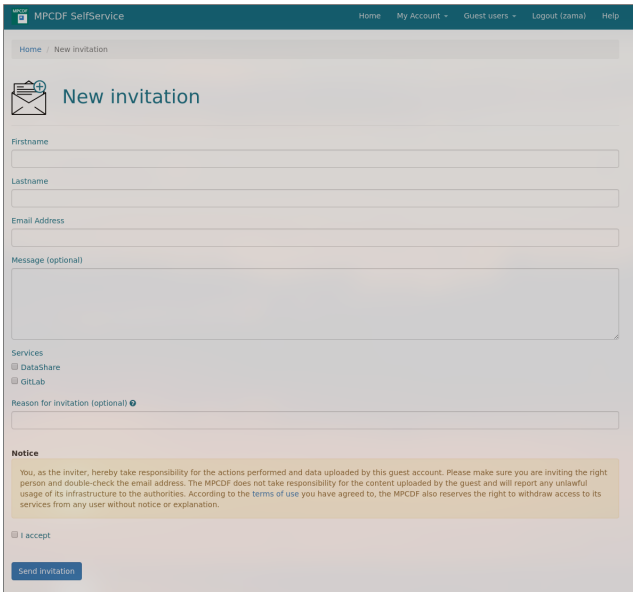

Figure 2: The invitation form

Existing guests can be managed under *Guest users* ⟩ *List of guests* (Fig. 3). Next to each guest there is an edit button that allows the user to grant and withdraw access to specific services, deactivate the account completely, prolong it, or send out a password reset mail (Fig. 4). These functionalities get restricted when the inviter deactivates the account and fully disabled if an administrator deactivates it. The inviter may choose to reactivate the guest account and change access rights on a per-service basis at any point provided the account has not been deactivated by an administrator. Again, changed access rights may take up to 20 minutes to manifest.

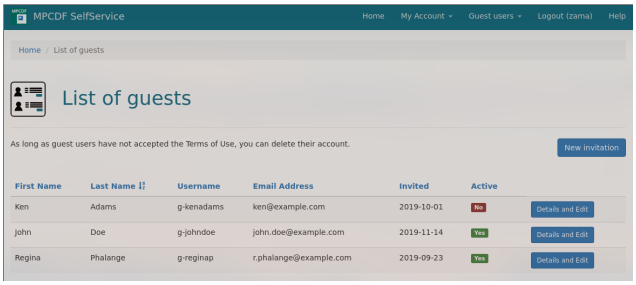

Figure 3: Overview of invited guests

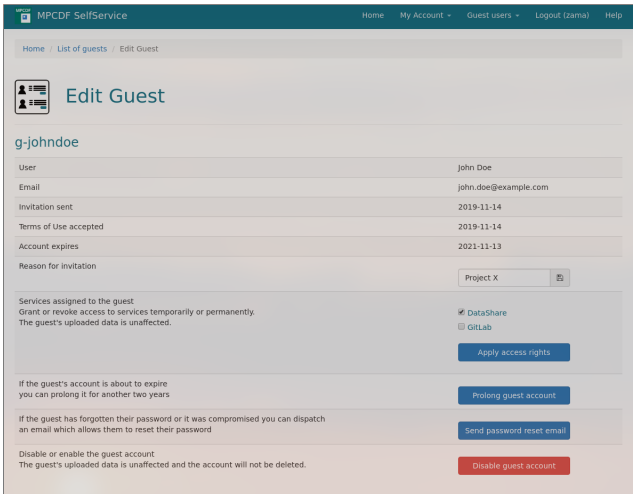

Figure 4: Editing a guest

Self-administration for guest users

Guests are currently not able to log in to the SelfService platform. However, they may use it to accept invitations or request a password reset e-mail. When accepting an invitation, guests have the chance to choose a username and are asked to set a password for their new account. For the safety of the users the MPCDF password policy disallows the reuse of a password that is in use with any other third-party service to prevent password-reuse attacks.

Further information and future plans

The SelfService help page contains more detailed information as well as FAQs and troubleshooting tips. The SelfService is envisioned to provide more options for self-administration as well as a login for guests in the future. The list of available services for self-subscription is also expected to grow.

Archival

Hermann Lederer

Data management with the HSM system HPSS

The Hierarchical Storage Management (HSM) system HPSS, introduced at the RZG in 2011, proves to be increasingly essential for managing the archival requirements of many Max Planck Institutes from all three scientific sections. The largest needs, however, in meantime arise from Life Sciences. Since late 2018, when the amount of data stored in HPSS at the MPCDF surpassed the 100 PB threshold, we have faced an increase of more than 36 PB. In the list of publicly disclosed HPSS deployments, as published in http://www.hpss-collaboration.org/customersT.shtml, the MPCDF continues to belong to the top 10 scientific data centers worldwide and remains on rank 1 within Germany.

Safety of Archival Data

In 2006, the president of the Max Planck Society, Prof. Gruß, requested the RZG to ensure that important data can be stored safely for at least 50 years. From the very beginning of mass storage at the RZG with an automated tape library in 1980, a CDC8500 with 2000 8-MB cartridges, the challenge we had to master was to provide bit preservation across multiple generations of archiving technologies, w.r.t both hardware and software. The standard solution has been to have two tape copies for archival data (for backup, the standard is only one tape copy). The MPCDF has started to further improve safety of particularly precious archival data which fulfill the following criteria: MPCDF is the master site, and there are no officially maintained data copies at other sites, and in the unlikely case of loss, the data would be unrecoverable or it would require immense efforts to recover the data. For such data, a third tape copy, using different software technology, is now hosted in Berlin on request. This procedure takes into account the updated recommendation for geo-redundancy which requires a distance larger than 200 km.

News & Events

Tilman Dannert, Hermann Lederer

HPC Workshop November 2019

For the second time an Advanced HPC Workshop has been held at the MPCDF from November 11 to 13, 2019. Over 40 participants from 18 Max Planck Institutes from diverse research directions attended the first 1.5 days. Experts from the application group and from Intel gave 15 lectures about different topics ranging from software engineering over (parallel) debugging and profiling up to overview talks about new hardware and GPU programming models. The final day was dedicated to "hands-on" sessions on a set of user codes, which were analyzed with the support of the experts from Intel and the application group of the MPCDF.

HPC Summer School 2020

The 11th International HPC Summer School 2020 is going to take place from July 12 to 17 in Toronto, Canada, hosted by the University of Toronto. Graduate students and postdoctoral scholars from institutions in Canada, Europe, Japan and the United States are invited to apply. The summer school is organized by XSEDE for the US, PRACE for Europe, R-CCS for Japan and the SciNet HPC Consortium for Canada. Interested students are invited to apply by January 27, 2020. School fees, meals and housing will be covered for all accepted applicants, as well as intercontinental flight costs. Further information and application: http://www.ihpcss.org.