No. 205
December 2020

Bits & Bytes

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

High-performance Computing

Mykola Petrov, Renate Dohmen, Markus Rampp, Sebastian Ohlmann

HPC System Raven: interim system in full production

The first CPU segment of the new Max Planck supercomputer Raven was taken into production operation in September 2020. This interim system provides 516 compute nodes based on the Intel Xeon CascadeLake-AP processor (Xeon Platinum 9242) with 96 cores per node (4 "packages" with 24 cores each), and 384 GB of main memory (24 memory channels) per node. The Raven nodes are interconnected with a Mellanox HDR InfiniBand network (100 Gbit/s) using a non-blocking fat-tree topology over all nodes. The machine has been stable and fully loaded from the very beginning and is routinely used by scientists from more than thirty Max Planck Institutes.

The HPC cluster Raven can be accessed either from MPG networks or via MPCDF gateway systems by using ssh (users always have to provide their Kerberos password and 2FA token on the login nodes, SSH keys are not allowed):

- ssh raven.mpcdf.mpg.de

The login nodes (raven is an alias for raven11i or raven12i) are mainly intended for editing, compiling and submitting parallel programs. Running parallel programs interactively in production mode on the login nodes is not allowed. Jobs have to be submitted to the Slurm batch system.

Raven uses the Slurm workload manager. To run production or test jobs, submit a job script to Slurm, which will find and allocate the resources required for your job (e. g. the compute nodes to run your job on). The number of jobs each user can submit and run at the same time is limited. By default, the job run limit is set to 8, the job submit limit is 300. If batch jobs can't run independently from each other, one can use job steps or contact the helpdesk on the MPCDF web page.

To test or debug codes interactively on the interactive nodes raven-i.mpcdf.mpg.de (raven13i, raven14i), one can use the command:

- srun -n ⟨#cores⟩ -p interactive -t ⟨time⟩ ⟨prog⟩

Interactive usage is limited to only a small number of cores, 60000 MB of memory and 2 hours runtime per job.

As on the HPC cluster Cobra, also on Raven a job submit filter is used which automatically chooses the right partition and job parameters from the resource specification of a submitted batch job. If not all necessary resources are specified or if there are inconsistencies, the job will be rejected at submission with warning message.

Note that exactly the same version of the Intel MPI library and runtime must be used during building and running an MPI binary, otherwise errors will occur. Also there are no preloaded modules. One has to specify the needed modules with explicit versions at login and also in the batch submission script, e. g.:

- intel/19.1.3 impi/2019.9 mkl/2020.2

AFS as well as the remote file systems /draco/u and /draco/ptmp are available only on the login nodes raven.mpcdf.mpg.de and on the pool of interactive nodes raven-i.mpcdf.mpg.de, and not on the Raven compute nodes. Please be aware that automatic cleanup of /raven/ptmp has been activated as of Dec 15, i. e. files in /raven/ptmp that have been not accessed for more than 12 weeks will be removed automatically. So, please archive your data regularly, e. g. into the migrating file system /r.

More information about Raven can be found on the Raven home page.

HPC system Raven: notes on upcoming GPU resource

As announced earlier this year, the current HPC interim system Raven will be replaced by the final extension stage during the first half of 2021. The new Raven machine will provide roughly twice the total CPU application performance delivered by the interim system, based on ca. 1350 dual-socket nodes with the new Intel Xeon IceLake-SP CPU. In addition, Max Planck scientists will have a very powerful GPU resource at their disposal: Raven will

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

comprise a total number of 768 Nvidia A100 GPUs. Each GPU delivers a nominal peak performance of 9.7 TFlop/s for 64-bit floating point (FP64) calculations, plus another 9.7 TFlop/s by the new FP64-capable tensor cores, and provides 40 GB of high-bandwidth memory capable of sustaining a bandwidth of up to 1.6 TB/s. A single Raven GPU node (with 2 IceLake host CPUs) hosts four A100 GPUs, with Nvlink (gen 3) connecting a GPU with any of its three peers at a bandwidth of 100 GB/s (per direction). PCI express (gen 4) links the GPU complex to the host CPUs at a total bandwidth of 128 GB/s (per direction).

The Raven GPU partition will support all relevant programming tools, including the latest Nvidia SDK (e. g. with CUDA and OpenACC-capable compilers, numerical libraries, etc.), machine-learning tools and other GPU-capable software.

Users are advised to begin with the porting as early as possible in order to be able to take advantage of the significant GPU computing capacity provided by the new Raven machine early on. Although the A100 GPUs will mark another step in the evolution in GPU hardware and software technology, the preparation of applications and in particular the porting of CPU-only codes can start right away, e. g. by using the GPU partition of the MPCDF Cobra system with 128 Nvidia V100 GPUs and corresponding software tools. The course material of the advanced HPC workshop organized by the MPCDF in November 2020 may serve as a good starting point for exploring details of the A100 GPU hardware architecture and software environment, GPU programming tools, as well as general strategies and specific hints for porting HPC codes to GPUs. Requests for application support for porting to GPUs are welcome (contact: Markus Rampp).

Intel MKL: new module for using only parts of its functionality

The Intel Math Kernel Library (MKL, `https://software.intel.com/en-us/mkl`) is a comprehensive numerical library that offers lots of functionality optimized for x86 CPUs – BLAS, LAPACK, ScaLAPACK, FFT and much more. However, its implementation of FFTW functionality is incomplete which can lead to problems for codes using both, a generic FFTW library (e. g. for features not supported by MKL) and other parts of the MKL, such as BLAS and LAPACK. To overcome this problem, parts of the MKL functionality can be bundled in specific shared objects using the builder tool, as described in Bits and Bytes 203 (April 2020).

The MPCDF now offers these shared objects as modules, specific for each compiler (‘mkl_parts‘) and compiler + MPI combination (‘mkl_parts-mpi‘). Each module exports ‘$MKL_PARTS_HOME‘ as a path to the base directory and offers several shared objects. The most important ones are ‘libmkl_blas.so‘, ‘libmkl_lapack.so‘, and ‘libmkl_cluster.so‘ for BLAS, LAPACK, and ScaLAPACK support. ScaLAPACK support is only available for the ‘mkl_parts-mpi‘ modules. All shared objects have been created correctly for the corresponding compiler and MPI library such that the link line does not have to be adapted (as is the case for the normal MKL link line). These libraries support threading; in addition there are also sequential variants with ‘_sequential‘ appended to the name.

For linking these libraries, you can use the following link lines:

BLAS: ‘-L$MKL_PARTS_HOME/lib -lmkl_blas
-Wl,-rpath=$MKL_PARTS_HOME/lib‘

LAPACK: ‘-L$MKL_PARTS_HOME/lib -lmkl_lapack
-Wl,-rpath=$MKL_PARTS_HOME/lib‘

ScaLAPACK: ‘-L$MKL_PARTS_HOME/lib
-lmkl_cluster
-Wl,-rpath=$MKL_PARTS_HOME/lib‘

If your code does not use OpenMP, you can also use the sequential variants.

Using these libraries, one can use, both, FFTW and MKL, without interference between those two libraries.

FAQ 4 2FA

Andreas Schott, Amazigh Zerzour

For accessing HPC systems and gateways, a stronger authentication mechanism has been applied since November. The introduction of this so-called two-factor authentication (2FA) has been successful and well accepted by the users.

Their valuable feedback has even led to further optimizations of the method, and additional new features could be implemented like the re-synchronization in case of a clock-skew on the mobile phone. The possibility to generate TAN lists will become available soon.

An FAQ for 2FA has been set up for sharing solutions for common problems. This document will be updated continuously to cover new questions and provide improved solutions for known issues around 2FA at the MPCDF. It can be found here: `https://docs.mpcdf.mpg.de/faq/2fa.html`

Charliecloud: containers for HPC

Michele Compostella, Andreas Marek

Introduction

In recent years, containers have become a popular tool to package software and all its dependencies into an isolated environment that is compatible and portable between different host systems. Among the large number of container engines available nowadays, **Charliecloud** (https://hpc.github.io/charliecloud/) provides a user-defined software stack specifically developed for High Performance Computing (HPC) centers, with a particular attention towards security, minimal overhead and ease of use.

Charliecloud runs on Linux systems and isolates the image environment using Linux *user namespaces*. Contrary to other container engines, it does not require privileged operations or daemons at runtime, can easily access the hardware available on the node (e. g. GPUs) and can be configured to run on multiple computing nodes (for example using the OpenMPI library). A Charliecloud container image can be created from any Docker image locally available in the user's workstation producing a tarball file that can be transferred to the HPC cluster where it is intended to run. See https://hpc.github.io/charliecloud/command-usage.html for a list of commands. Note that for the creation of a Docker image, root permissions are required on the local machine.

Installation

Charliecloud has minimal software requirements:

- Recent Linux kernel (recommended version 4.4 or higher)
- Bash (version 4.1 or higher)
- C compiler and standard library
- Docker (version 17.03 or higher)

and can be conveniently installed from one of the software tarballs (available at https://github.com/hpc/charliecloud/releases) using a standard 'configure-make-make install' procedure. Alternatively, Charliecloud is also available for local installation from several package managers.

Once the software has been installed, any Docker image previously built on the local system can be converted into a Charliecloud image using:

```
ch-builder2tar <image_name>:<tag> .
```

This command creates a *.tar.gz* file in the local folder containing everything that is needed to run the container on the HPC host.

In order to facilitate the access to Charliecloud for researchers of the Max Planck Society, Charliecloud has been deployed on the HPC clusters Draco, Cobra and Talos via the module system and can be provided on institute's clusters on request. The software on the host can be loaded by simply using

```
module load charliecloud
```

Once the container image has been transferred to the HPC cluster (e. g. using `scp`), the `.tar.gz` file can be decompressed to a folder with the command

```
ch-tar2dir <charliecloud_image.tar.gz> \
     <destination_folder>
```

and the container can be launched in a standard Slurm script using `ch-run`, as shown in the following example:

```
srun ch-run <image_dir> -- echo \
     "I'm in a container"
```

Multiple options are available, like, for example, the possibility to bind-mount local folders inside the container to access or store software and data available on the host system.

Performance

In order to assess the overhead introduced by the container engine and to test the performance of Charliecloud against software running on the 'bare metal', we run the Tensorflow 2 synthetic benchmark on 1 to 96 Tesla Volta V100 GPUs on the Talos machine-learning cluster. We focus on the following combination of software and libraries:

- `gcc 8`
- `openmpi 4.0.5`
- `CUDA 10.1`
- `cudnn 7.6.5`
- `nccl 2.4.8`
- `TensorFlow 2.2.0`
- `Horovod 0.20.2`

The resulting total number of images processed by the benchmark for the Charliecloud and the bare metal runs are shown in Fig. 1, together with the ideal scaling normalized to a single GPU. The overhead introduced by the container engine is minimal and the performance is almost identical to the bare metal run.

Conclusion

Charliecloud has been deployed at the MPCDF and is available via the module system, providing an additional platform to run containers at our HPC facility. The software is fully compatible with images created with Docker and grants effortless access to accelerated hardware and multi-node computation, while maintaining a strong focus on security and ease of use. Running Charliecloud containers can be efficiently integrated into any Slurm submission script for the HPC clusters available at the MPCDF without requiring any special privilege on the machine.

On their local workstation, users are entrusted with the preparation of the Docker source image and the installation of software into the container.

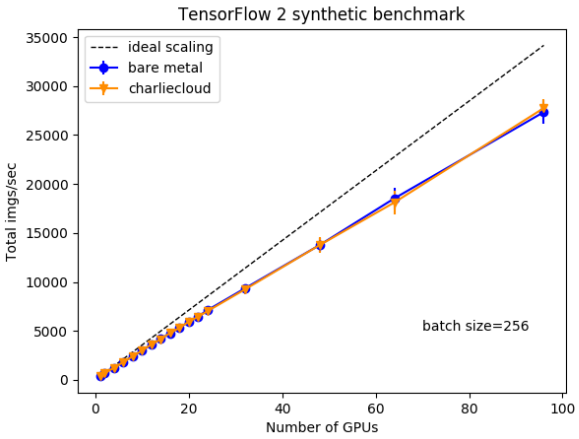

Fig. 1: Tensorflow 2 synthetic benchmark comparison between bare metal and Charliecloud runs. Performances are measured as total number of images per second processed by TensorFlow for an increasing number of GPUs.

repo2docker: Running Jupyter Notebooks via Docker

Thomas Zastrow

The tool repo2docker can be used to easily execute Jupyter Notebooks on your local machine. The only local requirement is an up-to-date and active Docker installation. The notebooks and any additional files need to be stored in a publicly available Git repository, for example the MPCDF GitLab instance.

repo2docker is a Python tool and can be installed via pip:

```
pip install repo2docker
```

After installation, a Git repository with Jupyter notebooks can be transformed into an active Docker container:

```
repo2docker https://gitlab.mpcdf.mpg.de/thomz/pandastutorial
```

The command will produce some output on the console. The last lines should be similar to this:

```
To access the notebook, open this file in a browser:
        file:///home/tom/.local/share/jupyter/runtime/nbserver-1-open.html
    Or copy and paste one of these URLs:
        http://127.0.0.1:36223/?token=d20b552e296f655d68882ae
```

Opening up the address in a browser, you can see the content of the cloned repository and execute any Jupyter notebook it contains.

Behind the scenes: repo2docker clones the Git repository and – if the file "requirements.txt" is present – the necessary requirements. The base image used by repo2docker contains an Anaconda installation based on Python 3.7.6 (current version of repo2docker is 0.11.0).

Warning: repo2docker is meant to run existing Jupyter notebooks. Its intention is not on developing or editing Jupyter notebooks. Any changes you do in the Jupyter notebooks or in other files are not automatically committed and sent back to the external Git instance! If you shut down the Docker container, your changes are gone. To avoid data loss, you can login manually to the Docker container while it is still running and execute Git commands on the shell.

News & Events

Raphael Ritz, Klaus Reuter, Thomas Zastrow

RDA-Deutschland Tagung 2021

Next year's conference of the German-speaking part of the Research Data Alliance will be an online event running from February 22 to 26, 2021. Topics covered will range from new developments within the alliance to related activities such as the European Open Science Cloud (EOSC) and the Nationale Forschungsdaten Infrastruktur (NFDI). Participation is free of charge but registration will be required to attend the sessions. Details will be made available at the RDA-DE Website.

New website with Frequently Asked Questions

The MPCDF has launched a new website at `https://docs.mpcdf.mpg.de/` to provide a collection of Frequently Asked Questions. The platform enables both, novice and expert users, to quickly find essential information on MPCDF systems and services and will be continuously updated and extended. Users are kindly asked to check that FAQ before submitting a support ticket.