No. 192
April 2016

Bits & Bytes

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

Remote visualization on Hydra

Klaus Reuter

Interactive GPU-accelerated visualization services are now offered on *hydra*. In particular, a number of the *hydra* GPU nodes were upgraded with additional main memory and with local hard disks for temporary user data. Compared to the former visualization cluster *viz*, users now experience increased performance thanks to more recent GPUs (2 x NVIDIA Tesla K20X per node, 6 GB GPU RAM each) and more powerful host systems (2 x 10-core Intel IvyBridge CPUs per node, up to 256 GB RAM). In addition, the integration of the visualization services into the supercomputer enables innovative approaches such as in-situ visualization.

To start, users are required to load the 'remotevis' module and run the script 'setup_remotevis' once. Interactive visualization sessions can be requested via the LoadLeveler batch system, similar to regular compute jobs. A sample submit script is placed in the directory '~/remotevis', which is automatically generated by the 'setup_remotevis' script. As soon as the visualization job starts on *hydra*, the user receives a notification e-mail with instructions. VNC is used to connect remotely to the VNC desktop running on *hydra*. The GNOME desktop environment is installed, and state-of-the-art visualization software is provided. Interactive visualization sessions are limited to 24 hours.

Please note that *hydra* remote visualization is intended to be used only in cases where GPU acceleration is required, e. g. for rendering with VisIt or ParaView. In case only a plain graphical remote desktop is needed, please use a VNC session as described in the following article.

VNC remote desktops on MPCDF Linux systems

Klaus Reuter

While command-line based terminal sessions via SSH should certainly satisfy most requirements of users involved in scientific computing, it is sometimes desirable to have a graphical remote desktop session running on a remote Linux server. To serve that purpose, VNC (Virtual Network Computing) provides a graphical remote desktop system that uses the RFB protocol (Remote Frame Buffer) to control a graphical desktop session on another computer over a network. In particular, the performance of VNC is superior compared to traditional X forwarding, especially when dealing with complex graphical interfaces on wide-area networks.

At the MPCDF, a typical scenario would be to run a graphical tool (e. g. a debugger, a performance analysis tool, or other software such as IDL, Matlab or Python/ipython/matplotlib) interactively on a Linux cluster or on the *hydra* supercomputer in a VNC session. A particularly useful feature of a VNC session is its persistence. Users may disconnect from the VNC session and reconnect later, potentially from different computers at different locations. Programs that run within the session continue to run as long as the VNC session is not shut down or killed.

For plain text applications, the GNU 'screen' tool offers similar persistent functionality at a much smaller resource footprint. Please consider using 'screen' instead of VNC in case you don't need to run GUI applications.

Detailed documentation is provided on the MPCDF web page at Network –⟩ VNC. VNC sessions can be launched on most interactive MPCDF Linux machines including the *hydra* login nodes.

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

BagIt: smart packaging of files

Raphael Ritz, Thomas Zastrow

When it comes to handling very large numbers of files even modern storage systems and networks can run into trouble. Using many small files can slow down file operations and network transmission, eventually causing applications to hang or even to crash. Due to these reasons, it is better to pack large amounts of files into container formats. Formats such as ZIP or TAR archives are widely used and in addition to encapsulating the files, the data itself can be easily compressed.

Specialized container formats like the BagIt specification offer more functionality, going beyond packing and compressing files. **BagIt** was developed by the California Digital Library and the Library of Congress for data exchange. It defines a hierarchical file system structure as well as checksums for every file and additional metadata.

Figure 1 shows the basic setup of a BagIt container:

- The subdirectory `data` contains the so-called payload, the files themselves. An arbitrary depth of nested folders and files is possible.
- The file `manifest-md5.txt` contains checksums for every file in the `data` directory. The file name indicates the algorithm which was used to calculate the checksums (here `md5`).
- `fetch.txt` allows to specify a list of URLs to external resources which can be downloaded to complete the data in the BagIt container.
- A file with the name `bagit.txt` contains information about the BagIt version etc.

For most common languages, there are libraries for easy creation and manipulation of BagIt containers available. Next time you wrap up a simulation or an experiment consider packaging your files in a BagIt container before moving it to an archive. It will pay off many times later on, as the data is better organized and can be validated at any time.

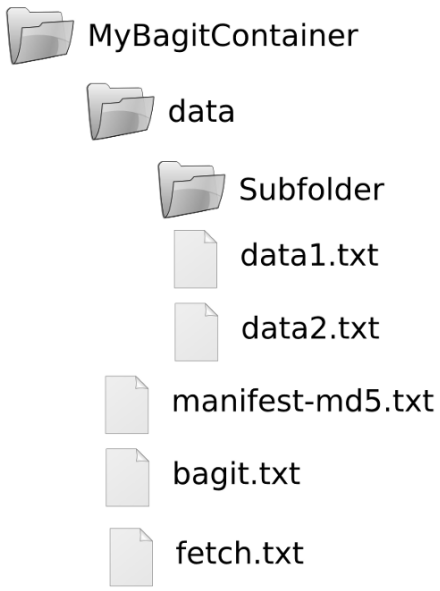

Figure 1: Basic structure of a BagIt container

Gitlab update

Florian Kaiser

Initially launched the end of last year (see Bits&Bytes No. 191), the Gitlab service has been well received and as of April 2016 hosts close to 200 projects. Over the last few months, quite a few features were added such as

Todos A chronological list of things waiting for your input, such as an issue or merge request being assigned to you or somebody @summoning you. A todo will be marked done when the corresponding issue has been resolved or reassigned.

Reply by mail Users can now comment on issues by replying to the e-mail notifications.

Confidential issues Issues can now be reported as confidential, making them visible only to team members.

Move issue Issues can now be moved to another project.

Improved search Search everything – code, issues, wikis and commit messages.

Releases Add a markdown-formatted message to any Git tag, which can include attached files.

Global milestones Milestones can now be created for multiple projects in a group simultaneously. This makes it easier to track activity and progress across groups and projects.

Group visibility level Groups now have a visibility level just like projects. This allows internal groups to be hidden from non-registered users.

Subscribe to label Get notified when a specific label gets added to an issue.

DataShare update

Florian Kaiser

A new release of the DataShare service was rolled out in December based on ownCloud 8.2. The service now keeps track of recent activity such as files being shared with you. This information is available through the web interface ('Activity'), but also in recent desktop and mobile clients. A lot of functionality that was previously only available through the web interface such as sharing files either with other users or via a link has been added to the native clients. We encourage you to update to the newest version (2.1 for Windows/Mac/Linux, 1.9 for Android).

Python infrastructure at the MPCDF

Klaus Reuter

The Python programming language is becoming more and more popular in scientific computing and data analysis. On MPCDF systems, a selection of important Python modules is provided, in particular an optimized build of the NumPy/SciPy stack that is linked against Intel MKL. The language versions 2.7 and 3 are supported. Use the command 'module avail' and see the 'python' section for an up-to-date list of the available software.

For Python, a very large package repository is available on the Internet (see `https://pypi.python.org`). MPCDF users request many of those packages regularly, however it is often not reasonable to provide the particular packages site-wide. To enable users to easily download and install Python software, 'pip' is now installed along with the Python interpreters. The command 'pip install --user PACKAGE_NAME' can be used to automatically install a certain package together with its dependencies locally to a user's home directory. The Python package is then located at '~/.local' and is automatically found by the interpreter. Please note that it is potentially important to load the 'python{27,33,35}/scipy' module before invoking 'pip' in order to satisfy the requirements for NumPy and SciPy that many scientific packages have.

In addition to the software provided by the modules 'python{27,33,35}/*' we have recently rolled out the versions 2 and 3 of Anaconda Python. Anaconda is a Python distribution available for free under a BSD-type license. On MPCDF systems it is accessible via the environment modules 'anaconda/{2,3}'. By default, Anaconda contains numerous packages for scientific computing and data analysis that are updated on a regular basis. Anaconda mitigates the installation effort which is significant for certain packages and enables MPCDF staff to react more quickly to user requests. The aforementioned 'pip' tool is available under Anaconda, as well.

Change in Forcheck

Markus Rampp

Forcheck is a commercial code-analysis tool which is licensed by the MPCDF and is provided on all systems via `module load forcheck`. Forcheck performs static syntax checks of Fortran programs and detects usage of undeclared or uninitialized variables, obsolete language features and other coding issues. A basic introduction to Forcheck was given in Bits & Bytes No. 183.

In order to analyze MPI programs Forcheck provides an MPI library interface file which, until Forcheck version 14.5, was referenced as `forchk -include $(FCKDIR)/share/forcheck/MPI.flb ...`

With the latest version 14.6 an API change was introduced by the vendor which eventually removed the common `MPI.flb` interface library. As a consequence, users are required to adapt their forchk command line by replacing `MPI.flb` by one of the following choices:

- `MPI_2.flb` provides the MPI 2.2 constants and Fortran 90 interfaces.
- `MPI_3.flb` provides the MPI 3.1 constants and Fortran 90 interfaces.
- `MPI_3_f08.flb` provides the MPI 3.1 constants and Fortran 2008 interfaces and uses derived types for several variables instead of integers.

For further documentation see `man forchk` or visit the Forcheck homepage

Backup of desktop systems

Hermann Lederer, Manuel Panea-Doblado

We would like to remind our users of the importance of regular backups of the data residing on desktop systems, so that in case of a virus infection (which may result in an encryption or damage of your files) data can be restored from the backup. Usage of the MPCDF backup service via TSM is highly recommended. This includes daily backups which are kept for 60 days so that also older versions (up to 60) can be restored. For Windows PCs, backups are normally executed during the night, so an office PC should be kept running over night (or at least from time to time). Backups of Windows laptops are done during working hours. In case no backup has been done for more than seven days, an e-mail notification will be sent. Status of backups of Windows systems can be inspected via the MPCDF web page, after login into your personal area. For Linux and Macintosh PCs and laptops, backups are configured only on demand. If you want backups of your Linux or Macintosh machine – which is highly recommended – please open a ticket on the MPCDF helpdesk.

MPCDF archive system – expansion of GHI for projects

Manuel Panea-Doblado

At the MPCDF, the archiving system HPSS is usually accessed through one of the GHI file systems, that is, one of the migrating file systems which automatically transfer files from disk to tape, ensuring that the disk file system always has enough space for new incoming data.

The two main GHI filesystems are /r for general user access from the HPC system *hydra* and from the machine *archive* and /p for specific projects and for Max Planck Institutes all over Germany. This 'projects GHI' /p has seen a remarkable load increase over the last few months. To improve its performance, all existing files were recently moved to a new disk system with better tuned operating parameters, a delicate action which involved much planning and testing and careful execution in order to rule out any possible loss of data. After the new disk system was successfully put into operation, the number of disks it contains was also doubled, giving the system double capacity and better performance (since the load gets spread across more disks).

In another area of the archiving system, one of the tape libraries was recently expanded from 10,000 to 20,000 tape slots. With it, the MPCDF currently has three tape libraries with a combined capacity of about 40,000 LTO tapes.

ELPA: improved eigensolvers for computational materials science

Hermann Lederer

ELPA-AEO has been selected as one of six BMBF projects from the fourth HPC call of the BMBF in the area of 'application-oriented HPC software for high-performance computing'. New, more powerful HPC architectures are associated with increasing energy consumption. ELPA-AEO (EigensoLver for Petaflop Applications: Algorithmic Enhancements and Optimizations) addresses this challenge by improving the application software with respect to both 'time-to-solution' and 'energy-to-solution'. The goal of ELPA-AEO is in particular to increase the efficiency of supercomputer simulations for which solving the eigenproblem for dense or banded symmetric matrices is a significant contribution, as it is the case especially in computational materials science, biomolecular research and structure dynamics. Starting with eigensolvers of the ELPA library (supported through a BMBF grant until 2011), through ELPA-AEO even larger problem sizes shall be addressed, the computing effort for the simulation shall be reduced, and for a given accuracy resource usage and energy consumption shall be reduced while maintaining high scalability. The multidisciplinary team consists of scientists of the Fritz Haber Institute in Berlin (Theory Dep.), the University of Wuppertal (Lehrstuhl für Angewandte Informatik), the Technical University of Munich (TUM, Lehrstuhl für Informatik V and Lehrstuhl für Theoretische Chemie), as well as the MPCDF as coordinating institution. The project started in February 2016 with a planned duration of three years.

Vagrant – Virtual Development Environments made easy

John Alan Kennedy

In this short article we would like to draw your attention to Vagrant, a powerful tool for virtual development environments for desktops, that can be installed and configured individually by advanced users (currently without support by the MPCDF).

Vagrant acts as a higher-level wrapper around virtualization software such as VirtualBox, VMware, KVM and Linux Containers (LXC). Moreover, Vagrant has native integration support for configuration management software such as Ansible, Chef, Salt, and Puppet. What this gives the end-user is a versatile tool for deploying and testing services and/or application software in well-defined, self-contained sandbox VMs.

Once a user installs Vagrant and Virtualbox the generation of a virtual development environment is trivial:

```
user> vagrant init <vagrant-box>
user> vagrant up
```

The 'vagrant init ⟨vagrant-box⟩' command can be used to create the initial configuration (Vagrantfile) and define the base VM (box) which it should be built with. Running 'vagrant up' will download a pre-existing VM from a web repository and start it. Now the VM is up and running a user can access it via:

```
user> vagrant ssh
```

Users can then install and configure the VM as they wish and also produce their own box from this image to share with others.

To start the development cycle from scratch (following a mistake or to test an automated deployment process) users can simply destroy the VM and re-create it as follows:

```
user> vagrant destroy
user> vagrant up
```

A typical Vagrant workflow is depicted below:

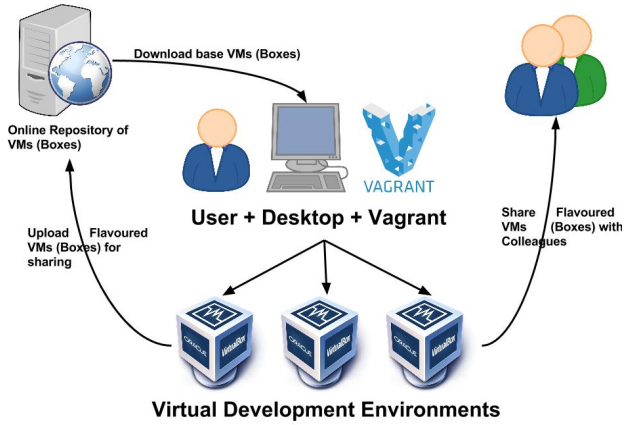

Figure 2: Typcial Vagrant workflow

Once a user 'flavours' a development VM they can directly share it with colleagues and also upload it to the central repository for sharing.

Vagrant puts the end-user in control of the environment in which they develop and test. This freedom leads to fast turn-around, simplifies the development process and the fact that it occurs in self-contained VMs means you don't dirty your desktop with dozens of packages etc. needed for development.

To learn more see: http://www.vagrantup.com