High Performance Computing Documentation

Smithsonian Institution High Performance Computing Wiki.

This Wiki holds information for the use of HPC resources at the Smithsonian.

Central to the SI HPC resources is the Smithsonian Institution High Performance Cluster (SI/HPC), named Hydra.

- High performance computing is administered by the Office of Research Computing within OCIO.
- The OCIO Herndon Data Center in Herndon, VA houses the high performance computing cluster, Hydra.
- The documentation is organized as follows:
  - Overview;
    - Citing/Acknowledging Hydra;
  - What's New;
  - Policies;
  - Hydra Training;
  - Quick Start guide;
  - Reference pages;
  - FAQs;
  - Cluster Upgrades

Quick Links [1]

- Cluster Status Page at SAO
- Cluster Status Page at OCIO
- QSub Generator

Note

- If you have any questions or encounter any problems, email:

`SI-HPC-Admin@si.edu`	for sys-admin related issues,
`SI-HPC@si.edu`	for non-SAO users who need help,
`hpc@cfa.harvard.edu`	for SAO users who need help.

- Past news with more details can be found at the What's New page.

<- Last updated: Thu May 16 17:07:03 2024 -> SGK

[1] Some of the pages are only accessible from trusted machines: within SI or SAO networks, or using VPN.

High Performance Computing

Welcome the Smithsonian Institution High Performance Computing Wiki.

This Wiki holds information for the use of HPC resources at the Smithsonian.

Central to the SI HPC resources is the Smithsonian Institution High Performance Cluster (SI/HPC), named Hydra.

- High performance computing is administered by the Office of Research Computing within OCIO.
- The OCIO Herndon Data Center in Herndon, VA houses the high performance computing cluster, Hydra.

- The documentation is organized as follows:
  - Overview;
    - Citing/Acknowledging Hydra;
  - What's New;
  - Policies;
  - Hydra Training;
  - Quick Start guide;
  - Reference pages;
  - FAQs;
  - Cluster Upgrades
- The complete documentation can be downloaded as a PDF document (180+ pages).

What's New

- **May 7, 2024** - Hydra has been upgraded to Linux Rocky 8.9 and 15 new compute nodes have been added.
  - See the "2024 Cluster Upgrade to Hydra-7" page for details.
  - *Please take the time needed to read these pages before contacting us for support.*

- If you have any questions or encounter any problems, email:

SI-HPC-Admin@si.edu	for sys-admin related issues,
SI-HPC@si.edu	for non-SAO users who need help,
hpc@cfa.harvard.edu	for SAO users who need help.

- Past news with more details can be found at the What's New page.

Quick Links

- Cluster Status Page at SAO
- Cluster Status Page at OCIO*
- QSub Generator *

*: these pages are only accessible from trusted machines: within SI or SAO networks, or using VPN.

Reminder

- References to `Hydra` (publications, proposals, etc.) should mention the cluster as:

> Smithsonian Institution High Performance Computing Cluster. Smithsonian Institution. https://doi.org/10.25572/SIHPC.

- or as an acknowledgment:

> "(Some of) [Tt]he computations in this paper were conducted on the Smithsonian High Performance Cluster (SI/HPC), Smithsonian Institution. https://doi.org/10.25572/SIHPC".

Last updated 03 May 2024 SGK/MPK.

Search this documentation

Recently Updated Pages

FAQs
3 minutes ago • updated by Korzennik, Sylvain
• view change
How to Use Containers

5 minutes ago • updated by Korzennik, Sylvain • view change

How to Use GPUs

11 minutes ago • updated by Korzennik, Sylvain • view change

Additional Tools

about 2 hours ago • updated by Korzennik, Sylvain • view change

Bioinformatics Software and Modules

about 3 hours ago • updated by Korzennik, Sylvain • view change

Overview

The Smithsonian Institution High Performance Cluster (SI/HPC), informally known as "Hydra" is the primary high performance computing resource at the Smithsonian Institution.

- It is housed at the Herndon Data Center, in Herndon VA and is supported by OCIO and SAO.
- Hydra is a Linux distributed cluster consisting of
    - some 5,900 CPU cores,
    - distributed over 80 compute nodes,
    - totaling over 43TB of total RAM,
    - all interconnected using 10Gbps Ethernet and 100Gps InfiniBand fabrics
- Some 4PB of storage is available using a 3-tier architecture:
    - high availability disk space (NetApp, via NFS),
    - high performance disk space (GPFS, via IB), and
    - near-line low cost disk space (NAS/ZFS, via NFS), only on a subset of nodes.

Citing Hydra and its DOI

If you have used Hydra for work that will be a part of a scholarly or professional publication, please cite it.

Here is a citation example:

> "*Smithsonian Institution High Performance Computing Cluster. Smithsonian Institution. https://doi.org/10.25572/SIHPC*"

or as an acknowledgment:

> "*(Some of) [Tt]he computations in this paper were conducted on the Smithsonian High Performance Cluster (SI/HPC),* Smithsonian Institution. *https://doi.org/10.25572/SIHPC*".

We also would very much appreciate being notified of such citations, so please email us (hpc@cfa.harvard.edu for SAO users, SI-HPC@si.edu for others) the reference to the citation, and if you can think of one, add a nice illustration with a short caption in layperson's terms.

Last updated 03 May 2024 SGK.

Citing/Acknowledging Hydra and Its DOI

If you have used Hydra for work that will be a part of a scholarly or professional publication, please cite it.

- We have a DOI for Hydra: https://doi.org/10.25572/SIHPC, This means that you are now able (and encouraged) to cite Hydra in papers, presentations, or posters whenever research has benefited from its use and add a DOI link to that citation or acknowledgment.
- If you have used Hydra for work that will be a part of a scholarly or professional publication, we ask you to mention it in the acknowledgments section of papers, adding the DOI will allow us to track citations and thus have better statistics for HPC scientific output, which in turn will help us lobby for resources for the cluster.

Here is a citation example:

"*Smithsonian Institution High Performance Computing Cluster. Smithsonian Institution. https://doi.org/10.25572/SIHPC*"

Or as an acknowledgment:

"*(Some of) [Tt]he computations in this paper were conducted on the Smithsonian High Performance Cluster (SI/HPC), Smithsonian Institution. https://doi.org/10.25572/SIHPC*".

We also would very much appreciate being notified of such citations, so please email us (hpc@cfa.harvard.edu for SAO users, SI-HPC@si.edu for others) the reference to the citation, and if you can think of one, add a nice illustration with a short caption in layperson's terms.

Last updated 03 Apr 2020 SGK.

What's New

- **May 7, 2024 -** Hydra was upgraded to Linux Rocky 8.9 and 15 new compute nodes were added.
    - See the "2024 Cluster Upgrade to Hydra-7" page for details.
    - *Please take the time needed to read these pages before contacting us for support.*
    - **Hardware Changes**

  We added 15 new compute nodes (2 nodes with 192 CPUs and 1.5TB of memory, 12 nodes with 128 CPUs and 1.0TB of memory, and 1 node with 4 GPUs - NVIDIA L40S, 48GB).

    - **Software Changes**

  Hydra's OS was updated from CentOS 7.9 to Rocky 8.9 to support the new compute nodes and the latest software offerings.

  Rocky 8 is the successor to CentOS 7. Both Linux distributions are based on Red Hat Enterprise Linux and share many similarities. We will also upgrade various packages to the most recent versions, including the job scheduler (i.e., the Grid Engine), and many of the modules. We will no longer support old versions of some software packages.

  We are updating the documentation on the Wiki and update the "2024 Cluster Upgrade to Hydra-7" page with details on what has changed including new module versions.

  As always we are striving to make this transition as smooth as possible, while leveraging the opportunities and challenges of using a new version of the OS.

- **November 27, 2023**
    - We have completed the FY23 hardware purchases and started work on upgrading the cluster's OS from CentOS 7 to Rocky 8.
    - We will add 15 new compute nodes, including a quad GPU server, and retire the oldest compute nodes.
    - The 2024 Upgrade page details the planned changes and will describe them once completed.
    - We are in the process of testing Rocky 8 and are currently aiming to transition Hydra to this OS in the end of January - beginning of February 2024 time-frame.
        - We anticipate the cluster being shut down for about 10 days for this work during which time no jobs will be running and there will be no access to the files stored on Hydra.
- **September 27, 2023**
    - **Software updates**
        - The latest version of Python available from Anaconda, namely version 3.11, has been installed. It can be accessed via the `tools/python/3.11` module.
            - The default version, `tools/python`, remains Anaconda's distribution version 3.8.
        - The latest version of IDL (8.9.0) has been installed as well as their latest license manager. IDL is accessed via the `idl` module.
            - The default version remains 8.8.1, you can access the more recent versions via the `idl/8.8.2` or `idl/8.9.0` modules. Note the oldest versions, i.e.. 8.6, will no longer work after Nov 30, 2023.
        - The latest versions of MATLAB runtime has been installed on Hydra: R2022b, R2023a and R2023b. They are accessible via the `matlab/2022b` and `matlab/2023[ab]` modules.
            - The default version remains 2021b.
        - The latest version of Julia has been installed on Hydra, namely version 1.9.3, via the `tools/julia/1.9.3` module.
            - The default version remains 1.6.3.
        - The latest NVIDIA compilers, formerly PGI, have been installed: versions 23.5 and 23.7, and are accessed via the `nvidia/23.5` and `nvidia/23.7` modules.
            - Note that all flavors of MPI for these new versions are not working on Hydra.
            - The default version remains 21.9, the newer available versions are 22.[1239] and 23.[357], with full MPI support up to 23.3.
        - We are unable to install the latest INTEL compilers, since INTEL only releases new versions for Rocky 8, and no longer for CentOS 7.
            - The latest INTEL compiler versions are 2022.[12], the default version remains 2021.4.
    - **Upgrade of Hydra to Rocky 8**
        - We will update Hydra's OS from CentOS 7 to Rocky 8, to fully support the latest hardware and software offerings.
            - The new compute nodes we are in the process of purchasing (see below) require Rocky 8 as well as various new software packages.
        - Note that the transition to Rocky 8 might not be as transparent as previous CentOS upgrades. We will strive to make it as smooth as possible.
        - We do not have yet any estimate of when we will transition to Rocky 8, so stay tuned.
        - As usual, we will give at least four weeks notice for any scheduled downtime.
    - **Hardware updates**
        - We plan to refresh 14 compute nodes with high end servers fitted with the latest AMD processors (Zen4), with 128 or 192 cores per server and 1 or 1.5TB of memory each (12 & 2 respectively).
        - We also plan on adding one GPU server with four A100 GPUs.
        - We will retire our oldest compute nodes (compute-43-xx). We thank Deron Burba, SI CIO, for contributing additional funding to make these purchases possible.
        - We have recently evaluated various storage options and anticipate expanding our storage on Hydra next year, adding more `/scratch` space and if possible adding a somewhat smaller but very fast different storage system.
    - **Personnel changes.**
        - After more than 7 years at the Office of Research Computing (ORC), Rebecca Dikow will leave ORC and move on to her next endeavor.
        - The ORC is putting in place a transition plan so as to avoid any disruptions this might cause. Please use si-hpc@si.edu or si-hpc-admin@si.edu for communicating with us, instead of emailing Rebecca directly for issues/questions related to Hydra.

- **November 15, 2022**

The migration of the 'disks' /home and /share/apps to a hybrid aggregate (a more performant set of disks that combines SSD and HDDs) was completed this past weekend. This will speed up access to the files stored under /home and /share/apps.

    - As a result, the total size of /home is slightly smaller, while the sizes of /data/sao and /data/genomics have been increased.
    - Quotas have not been changed.
    - If /home fills up too quickly we may have to reduce the quota on /home.
    - Users who want to use /data need to request access (up to 2TB of un-scrubbed space); non-SAO users should contact Rebecca, SAO users should contact Sylvain,

Please refer to Disks Space and Usage on the Wiki's Reference pages as to what disks to use and what to store where.

- **November 7, 2022**
    - 21 new compute nodes were added to Hydra as compute-65-xx (Dell R6515: AMD EPYC CPUs, 64 cores, 512GB memory).
    - All compute nodes with an AMD EPYC CPU have their "cpu_arch" set to "zen."
    - All the old compute-81-xx nodes (Dell R815) have been retired.
- **May 9, 2022**
    - The latest versions of the NVIDIA and Intel compilers (22.1, 22.2, 22.3 & 2022.1, 2022.2 respectively) have been installed,
    - The latest version of IDL (8.2.2) and MATLAB runtime (R2022a) have been installed,
    - The required modules are available, the default version have not yet been changed,
    - The examples have yet to be expanded to the new versions (the required changes should be obvious).
    - We plan to change the default versions some time in early June.
- **January 6 2022**

    - Since we have increased the cluster capacity, the maximum number of CPUs (slots) a user can use concurrently has been increased from 640 to 840.
- **December 17 2021**

    - Eight new compute nodes have been added to Hydra, bringing the totals to 5,408 CPUs for 98 nodes, and 42TB of memory.
    - We have noticed a read performance problem on the GPFS and are working with the vendor to resolve it as soon as possible.
- **December 2, 2021**
    - Look of the status pages has been update, and URL can now take up to 3 arguments.
    - New hardware (8 servers and 56 GPFS disks) has been delivered and will be deployed soon.
- **November 29, 2021**
    - The default version for 7 modules has been updated as announced in the Nov 22 update (see below.)
- **Nov 22 2021**
    - The documentation has been updated and reorganized to reflect the most recent changes.
    - New versions of the compilers and several tools have been installed,
        - default versions will get shortly updated, details are here.
- **Sep 25 2021**
    - IDL 8.8.1 is available, use: `module load idl/8.8.1`
    - Loading idl/8.8 will still load 8.8.0 for a little while. Also the IDL licensing method has changed, you will now see the message:

      ```
      License: 100554-5516875-BUF
      License expires 30-Nov-2021.
      ```

      which is normal (similar to SAO/CfA/CF's installation).
- **Sep 14 2021**
    - The next major upgrade of the SI/HPC cluster is completed.
    - We have made every effort to set up the new configuration as backward compatible as possible, although a few things have changed.
    - Please look at the 2021 Cluster Upgrade page for details.
- **Jun 24 2021**
    - The next major upgrade of the SI/HPC cluster, Hydra, will take place from **August 30th through September 14th, 2021.**

        - During the upgrade, Hydra will be inaccessible to users, and
        - as of 9am EDT on Monday August 30th any running jobs will be killed and any queued jobs will be deleted.

      While we are making every effort to set up the new configuration as backward compatible as possible, there will be changes. We hope to have Hydra back up before September 8th, but that decision won't be made until the upgrade work is completed. During the downtime, access to files stored on Hydra will be limited, and at times unavailable, although none of your files will be deleted.
    - Please look at the 2021 Cluster Upgrade page for additional details.
- **Sep 15 2020**
    - Scrubbing on /scratch has resumed: files older than 180 days are scrubbed
    - The GPFS s/w is in the process of being upgraded from v4 to v5, on a rolling basis and transparent to the users
    - IDL v 8.8[.0] is available
- **Apr 3 2020** - Hydra DOI
    - We have setup a DOI for Hydra (https://doi.org/10.25572/SIHPC)
      You are now able and encouraged to cite Hydra whenever research has benefited from its use and add a DOI link to that citation or acknowledgment.
- **Mar 24 2020** - Hydra status while teleworking
    - Hydra remains up and running.
    - We will address problems that require on-site staff as fast as possible.
    - We will answer people's questions and requests as promptly as possible.
    - Access to Hydra via VPN:
        - users are asked to limit the strain on the institutional VPN resources when ever appropriate.
    - Hydra can be accessed without VPN:
        - use the "*Hydra*" link under "*It Tools*" (or use RDP) at telework.si.edu.
        - SAO users can use login.cfa.harvard.edu to `ssh` to Hydra.
        - Access to the self serve password page is now working.

        - How to use Dropbox or Firefox Send (`ffsend`) to copy files to/from Hydra is documented on the Wiki.
    - The scrubbing policy has been modified as follows:
        - the scrubber will run on `/pool/sao` and `/pool/genomics` as usual, but
        - the scrubbed content will not be deleted for at least 21 days, and
        - we will accept requests to preserve what was scrubbed (beyond 21 days) as long as needed. To get your files restored, follow the usual instructions.
    - Users are asked to remain in contact via their SI email.
- **February 27, 2020** - `cpu_arch` resource and IDL 8.7.3
    - We have added a new resource, called `cpu_arch`, to allow users to direct jobs on nodes with CPUs of a specific (list of) architecture(s). If you run jobs/codes that can only run on (a) specific type(s) of processors, look at the new section CPU Architecture under the Available Queues page.
    - IDL version 8.7.3 has been installed on Hydra, and is accessible via the idl/8.7.3 module. The idl/8.7 module is now pointing to idl/8.7.3
- **January 14, 2020** - Increased total slot limit
    - The total number of slots (CPUs) a user can grab has been increased from 512 to 640.

Last updated 03 May 2024 SGK

Hydra Policies

Existing Smithsonian General Policies and SI Computer and Network Usage Policies

The Smithsonian Institution High Performance Computing resources (i.e., SI/HPC: the `Hydra` cluster and its associated resources) is an official SI asset and, as such, is subject to all of the regulations and policies outlined in SI's official directives, such as SD-931.

Since all SI/HPC users must have a Smithsonian network account before being granted an account, they have already agreed to the policies described in SD-931.

💡 New SI/HPC account holders should consider reacquainting themselves with SD-931. In addition, it is expected that SI/HPC users' computer security awareness training (CSAT) is up to date.

User accounts

Individuals who have been granted a Smithsonian network account are eligible for a `Hydra` account.

In addition, it is expected that each SI/HPC user will provide and keep up to date the following, which will be gathered through an online form:

- Name
- Unit/Department
- Supervisor name
- Supervisor approval via online interface
- 1-2 sentence description of the work they plan to conduct on Hydra.

SAO/CfA affiliates should the account request form hosted at the SAO/CfA ITS/HPC info page, managed by the SAO HPC analysist (currently: Sylvain Korzennik).

Introduction to Hydra workshops are held quarterly; contact SI-HPC@si.edu for more information.

Note:

- The sharing of credentials on SI/HPC assets (like `Hydra`) is strictly prohibited.
- *Borrowing* disk space from other users to bypass quota is also prohibited. While it is, of course, OK to share data with collaborators, users should not try to bypass quota limit by having their data stored by others.
    - Either policy violation may/will result in suspension/cancellation of the user's account. Users should contact us if they need more resources to complete some specific task.
- Users who have temporary appointments (students, postdocs, fellows, contractors, etc.) will need to renew their accounts annually (including SAO's postdocs/fellows).
- Users and their supervisors will be contacted upon account expiration (via the email in their `~/.forward` file). If there is no response for the user or their supervisor within 30 days, the data will be subject to deletion.
  ⚠️ If someone else should be contacted regarding a user's account/data, or the user's supervisor has changed or left, it is the user's responsibility to notify their SI/HPC contact person.

Hydra Use

As with all assets on the Smithsonian network, `Hydra` users should have no expectation of privacy concerning their use of `Hydra`. Users should adhere to "*Rule 5*" of SD-931 regarding appropriate computer and network use.

`Hydra`'s administrators reserve the right to suspend, cancel, or modify user accounts, quotas on the public disks, queue configuration, and other configuration settings, without warning and as needed to maintain the cluster integrity and optimal use as a *shared resource.*

Users should keep in mind that `Hydra` is a *shared resource.* As such, users should avoid conducting analyses on the login nodes or the head node, be mindful that others are using the cluster, and understand that others' work is no less important than yours.

💡 If you are wondering whether your behavior is in violation of "the spirit" of shared-use, consider whether dozens of people doing the same thing as you would adversely affect the functioning of the cluster.

✅ Users are responsible for monitoring the status and the progress of their jobs. Users who plan to conduct many similar operations should start with a small (set of) test job(s) before scaling up their use. Users who have jobs running on `Hydra` should be able to access the cluster while their jobs are running, so they can adjust their use if need be.

✅ Since the SI/HPC resources are a shared resource, users are expected to do a best effort to estimate their needs (CPU time, memory, disk space) and have a handle on how it scales with the size of their analysis.

⚠️ Users are subject to suspension or cancellation of their accounts if they systematically abuse a scarce or depleted resource (memory -aka RAM- disk space, especially high throughput disks like SSDs, etc.) or bypass the resource limits in place.

Disk Use on Hydra

Disk storage on `Hydra` is not to be used for archival storage. Users should have no expectation of the long-term viability of their files stored on `Hydra`. While the disk system on `Hydra` is highly reliable, the Smithsonian is not responsible for any data that are lost on `Hydra`.

Data on disks on Hydra are not backed up, except for the `/home` and `/data` disks, and old data on the public disks will be regularly removed (scrubbed) according to the following model:

- Files and directories on `/scratch/public` (aka `/scratch/{biology|genomics|nasm|sao})` will be scrubbed after 180 days.
- Files and directories on `/pool/public` (aka `/pool/{biology|genomics|nasm|sao})` will be scrubbed after 180 days.
- The backups of `/home` and `/data` are mainly for disaster recovery, since we have snapshots enabled on these disks.

Remember:

- All public disks have quotas and are scrubbed.
- Analyses on large files and data-sets should be conducted using the `/pool` or `/scratch` disks.
- The `/home` and `/data` disks should not be used for large files, their low quota will prevent users from storing large files.
- All data associated with an expired account is subject to deletion 30 days after the account expires, unless agreed otherwise.
- It is the responsibility of the user leaving SI to either save her/his data or pass on the responsibility of her/his data to her/his supervisor.

Oversubscribed and Inefficient Jobs

We monitor the cluster usage for the following conditions:

- Oversubscribed jobs: scaled CPU usage > 133%;
- Very inefficient or “hosed” jobs: efficiency (CPU/age) < 10% and age > 36hr;
- Inefficient jobs: scaled CPU usage < 33%;
- Jobs that over-reserved memory: >2.5x actual RAM use.

Users with such jobs receive warning emails (sent to the email listed in their `~/.forward` file).

User responsibility to respond to warnings:

✅ For “hosed” and oversubscribed jobs, users should respond to warnings by emailing si-hpc-admin@si.edu within 24 hours of receiving the warnings, whenever possible, to help determine whether jobs should be killed. Oversubscribed jobs that are expected to run for over 24 hours after receiving the first warning should be killed and resubmitted with the correct parameters.

⚠ When Hydra’s usage is high, or when the over-subscription is excessive, jobs may get killed promptly by the system administrator team at their discretion. When the cluster load is very high, users with a lot of inefficient jobs will have some of their jobs automatically killed and will receive a warning. Currently the load threshold for this is 70% and users will have their inefficient jobs trimmed down to have on only 100 'unused' slots per user.

✅ For inefficient and memory over-reserved jobs, users should monitor their jobs, not ignore the warnings and contact the support staff if they do not know or understand why the warnings are occurring and/or how to fix the problem for future jobs.

Our overarching goal is to support all users: to get all jobs through the queue and help users learn how to best make use of a shared resource. Remember that oversubscribed jobs are likely to slow down someone else’s job(s) running on the same compute node(s), while inefficient or memory over-reserved jobs are clobbering the system and preventing the scheduler from starting the jobs waiting in the queue.

💡As a reminder, all users are expected to read messages sent to the email address listed in their ~/.forward file.

Communication

All users are expected to check **and** read their email regularly, including those from the SI HPCC-L listserv. Users will be added to the HPCC-L listserv when they are granted an account.

The SI/HPC admins use email to:

1. announce new features (e.g. on `Hydra`),
2. warn users when their jobs are improperly using resources,
3. warn users when their files will be scrubbed,
4. warn users whose accounts are expiring,
5. announce configuration changes, and
6. announce new policies or changes to existing policies.

Users should be sure that the file `~/.forward` (in their home directory on `Hydra`) contains a working email address that is regularly checked.

A `~/.forward` file is created for each new account, using the user *canonical* email. See here about creating a `~/.forward` file.

Whom to Contact

- SAO users: Sylvain Korzennik at hpc@cfa.harvard.edu
- Non-SAO users: Alex E. White, Matthew Kweskin, Vanessa Gonzalez, and Mike Trizna and Kenneth (Tripp) Macdonald at si-hpc@si.edu

Hydra Training

Introduction to Hydra

All new (non-SAO) Hydra users are required to watch the video available at this link. The purpose of this training is to:

1. Provide an overview of Hydra, the Smithsonian High Performance Computing Cluster
2. Give Hydra users the basic information to get started using Hydra
3. Inform users about the Hydra usage policies (posted here) and ensure that they agree to abide by those policies

For questions concerns about the training, please contact us at si-hpc@si.edu (non-SAO users) or hpc@cfa.harvard.edu (SAO users).

Other training materials are available in this GitHub repository.

We also offer quarterly synchronous workshops - these are announced via the HPCC-L email list.

Advanced Workshops

We also offer advanced Hydra workshops on a semi regular basis.

Past workshops are accessible at this GitHub repository, where you can find slides, hand-on material and recording:

- Job Arrays (given on 2023-06-21)
- Shell scripting and useful tools (e.g. sed, awk, grep, ..., given on 2023-05-10)
- Installing software and writing modules on Hydra (given on 2023-03-22)
- Conda on Hydra (given on 2023-02-15

Future advanced workshops will be announced via the HPCC-L email list.

⚠ Note that you must be on the SI staff network, SI-STAFF VPN, CfA's VPN or telework to access SI's Microsoft Stream content. ⚠

Last updated 22 Jan 2024 RBD/SGK.

Quick Start Guide

The quick start guide tells you how to:

- Log into Hydra;
- Change your password;
- Transfer files to and from the cluster;
- Use the QSub Generator Utility to help you write a job script;
- Submit a job on the cluster.

Last Updated 13 Nov 2019 MK/SGK.

Logging into Hydra

> ⚠ **Note:**
>
> The material on this page is part of the Quick Start Guide and is not exhaustive. For more details, please see the Reference Pages.

- Access to the Hydra cluster is through a remote terminal connection:
  - If you are using a Mac you will use the built-in `Terminal` App,
  - for Windows, you can use the ssh client integrated into new versions of the Windows command prompt, the program `PuTTY`, or another ssh client
  - for Linux use `ssh`.
- When you received your email from the Hydra admin team with your user account information it contained your `username` and a link to reset your initial `password`.
  - You will need your username to reset your initial password,
  - and enable VPN, log into telework.si.edu, or use a "trusted" computer to connect to the self-serve password change/reset page.
- There are two computers (known as login nodes) that you can log into for Hydra access:
  - `hydra-login01.si.edu`, and
  - `hydra-login02.si.edu`
  - You can use either one.

> ⚠ **Note:**
>
> To connect to Hydra you must either:
>
> - use a "*trusted*" computer: one connected to the Smithsonian network - SI or SAO/CfA, or
> - use SI's or SAO/CfA's VPN, or
> - *for SAO/CfA users only*, log on one of these hosts first:
>   - `login.cfa.harvard.edu` (for CF users) or
>   - `pogoN.cfa.harvard.edu` (for HEA users), where `N` is 1, 2, ..., 6

Requesting Access

- SAO users should follow the instruction posted on the CF: Services: High Performance Computing web page,
- Non-SAO users should fill out this online form for a new Hydra account and this online form for a VPN account.

Logging in From a Computer Running MacOS

1. Open the Terminal application by going to /Applications/Utilities and finding Terminal.
   a. You can get to the Utilities folder by going to the Go menu in the Finder and choosing Utilities
2. At the command prompt (that ends with `%` by default on newer versions of macOS) type in this command, replacing `username` with your hydra username (all lower case) from you welcome email:

   ```
   % ssh username@hydra-login01.si.edu
   ```

   ⚠ The first time you login you will see this message, type `yes` to continue:

   ```
   The authenticity of host 'hydra-login01.si.edu' can't be established.
   RSA key fingerprint is ...
   Are you sure you want to continue connecting (yes/no)? yes
   ```

   You will get a password prompt where you should enter the password you selected when using self-serve password reset page (check here how to change or reset your password).

   ⚠ Note: no text, stars or bullets will appear when you type in your password.

   ```
   Warning: Permanently added 'hydra-login01.si.edu' (RSA) to the list of known hosts.
   Password:
   [username@hydra-login01 ~]$
   ```

You are now logged into Hydra!

Logging in From a Computer Running Windows

Recent releases of Windows 10 (version 1803 and newer) come with a ssh client that is available through the Command Prompt or PowerShell.

1. Open Command Prompt or PowerShell which can be found by searching the Start menu.
2. At the command prompt type in this command, replacing `username` with your hydra username (all lower case) from you welcome email:

```
C:\Users\WindowsUsername> ssh username@hydra-login01.si.edu
```

⚠ The first time you login you will see this message, type `yes` to continue:

```
The authenticity of host 'hydra-login01.si.edu' can't be established.
RSA key fingerprint is ...
Are you sure you want to continue connecting (yes/no)? yes
```

You will get a password prompt where you should enter the password you selected when using self-serve password reset page (check here how to change or reset your password).

⚠ Note: no text, stars or bullets will appear when you type in your password.

```
Warning: Permanently added 'hydra-login01.si.edu' (RSA) to the list of known hosts.
Password:
[username@hydra-login01 ~]$
```

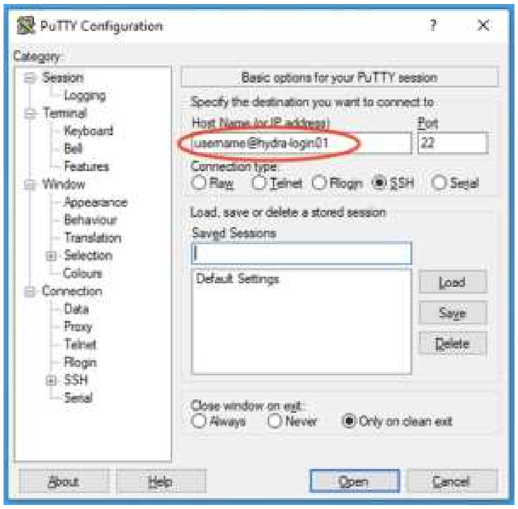
The change in the command prompt from the Windows prompt `C:\Users\...>` to `[useranme@hydra-login01 ~]$` shows that you are now connected to Hydra.

If your release of Windows 10 does not have ssh (you get an error message like: `'ssh' is not recognized as an internal or external program...`), we recommend using the program PuTTY to connect to Hydra.

The program to download is putty.exe.

1. The downloaded putty.exe can be run without running a Windows installer program.

2. Start PuTTY and in the Configuration screen that opens enter `username@hydra-login01.si.edu` in the `Host Name` text box (replacing `username` with your hydra username (all lower case) from you welcome email), then press `Open`



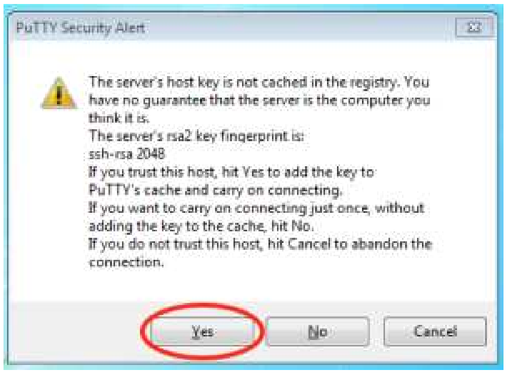
Host Name (or IP address): username@hydra-login01

Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Close window on exit: ○ Always ○ Never ● Only on clean exit

⚠️ The first time you connect you will get a warning about the Server's host key, choose the "Yes" option



The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

1. A terminal window will open.

   At the `Password:` prompt use the password you selected when using self-serve password reset page (check here how to change or reset your password).

   ⚠️ Note: no text, stars or bullets will appear when you type in your password.

   ```
   username@hydra-login01's password:
   [username@hydra-login01 ~]$
   ```

   You are now logged into Hydra!

Logging in From a Computer Running Linux

- From a trusted computer (i.e. most CF- or HEA-managed machines, or after enabling SI's or SAO/CfA's VPN) use `ssh` to connect to one of the two login nodes:
    - `ssh hydra-login01.si.edu`

  or

    - `ssh hydra-login02.si.edu`

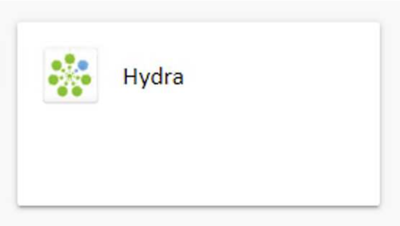
💡 Linux (and Mac) users can enable password-less login into Hydra using the Public key authentication.

- Follow the instructions here or Google "passwordless ssh mac" or "passwordless ssh mac" and follow the instructions.

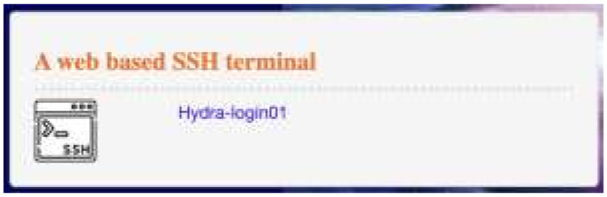
Logging in via telework.si.edu

The telework.si.edu is available from inside the Smithsonian network as well as remotely. There is a web-based terminal program available on telework.si.edu to access Hydra.

After logging in, expand the "IT Tools" section choose "Hydra".



Choose one of the "A web based SSH terminal" links to start the web terminal connection to one of the login nodes.



At the `login:` prompt, enter your Hydra username and at the `password:` prompt, enter your Hydra password.

Last updated 20 Nov 2021 MK/SGK.

Changing your Hydra Password

You can change your password either:

1. Logging in into hydra and using CLI (command line interface), *before it expires;*
2. Using the Self-Serve Web Page (*up to 14-days after it expired*).

Note:

The material on this page is part of the Quick Start Guide and is not exhaustive. For more details, please see the Reference Pages.

Passwords must be changed when your account is created and every 180 days thereafter.

Note

- You will receive an email notification to change your password before it expires.
- Please change it before it expires. See below how to change or reset it *yourself*, see below.
- *Do not send us an email requesting to have it reset, use the Self-Serve web page.*

Changing your Password using CLI (command line interface)

1. Log into `hydra-login01.si.edu` or `hydra-login02.si.edu` (using `ssh` or an ssh client).
2. At the command prompt use the `passwd` command, as follows:

```
[username@login-30-1 ~]$ passwd
Changing password for user username.
(current) LDAP password:
New password:
Retype new password:
```

Since we use LDAP, this is it!

Note the following restrictions:

Passwords must conform to SI password policies and meet the following requirements:

a. at least 12 characters in length, and include at least:
   i. one digit,
   ii. one upper case,
   iii. one lower case, and
   iv. one special character.
b. moreover, new passwords cannot be too similar to old passwords.

Resetting or Changing your Password via the Self-Serve Web Page

1. If you need your password to be reset go to the Self-Serve Password Page either from direct link (VPN or trusted computers) or via the Hydra option on the telework.si.edu site and choosing "Change Password on Hydra":

   a. you will receive a link via email (to your canonical email, the one ending in `.edu` in most cases).
   b. follow that link to enter your new password.
2. That same page also allows you to change your password, by entering your current one and the new one.
   a. Note that there is a 14-day grace period to allow you to change you password *after it expired* before your account gets locked.

Last updated 13 May 2024 SGK.

Transferring Files to/from Hydra

> ⚠ **Note:**
>
> The material on this page is part of the Quick Start Guide and is not exhaustive. For more details, please see the Reference Pages.

Introduction

This page explains how to copy files for your analysis to and from Hydra.

- File transfers can be done either
  - through command line, using `scp`, `rsync` or `sftp, or`
  - using a GUI application like FileZilla, WinSCP etc. That is, any tool that implements the `scp` or `sftp` protocol.

⚠ While the graphical interface of these programs is convenient, we recommend the command line tools `scp`, `rsync` or `sftp` for greater control and reliability, *especially for large transfers.*

- Transfers to Hydra can only be initiated from "*trusted*" computers (i.e., you may need to enable VPN);
- File transfers using the telework.si.edu require the use of a cloud service or use of an intermediate trusted computer (see instructions on using ffsend or rclone);
- Transfers *from* Hydra to your local machine are not limited to trusted destinations.

⚠ Large files should always be copied directly to the `/pool`, `/data` or `/scratch` directories, not to your home directory (`/home`) because of space limitations.

Consult the page about disk space and usage for more about data storage on Hydra.

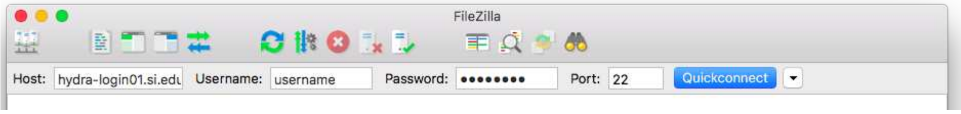
Additional Details

The following quick start instructions show

- how to configure `FileZilla` for file transfers,
- how to use `WinSCP`, and
- a short introduction to `scp`, `rsync` or `sftp`.

Using FileZilla

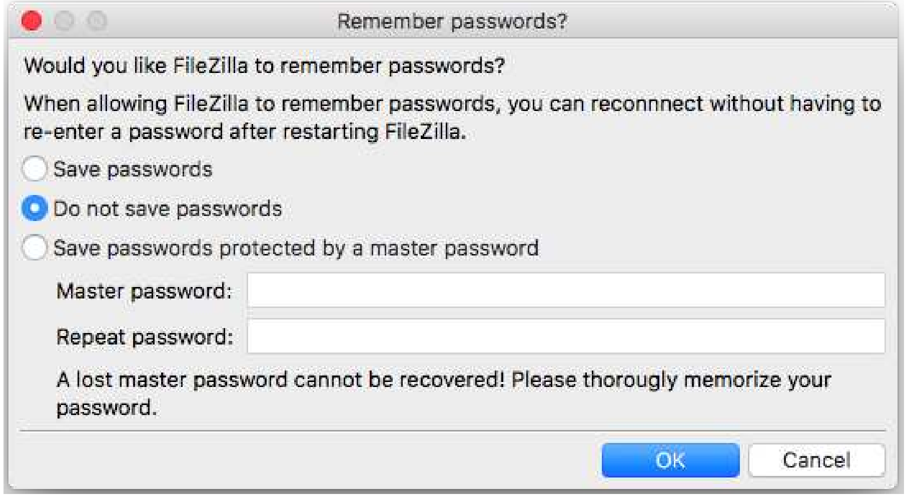
1. "FileZilla Client" is available for Mac, Windows and Linux systems.
   Mac screenshots are shown here, but use on other systems is similar.
2. Open the FileZilla application.
3. In the Quickconnect toolbar at the top of the window enter:
   a. Host: `hydra-login01.si.edu` or `hydra-login02.si.edu`
   b. Username: your Hydra username
   c. Password: your Hydra password
   d. Port: 22



Host: hydra-login01.si.edu Username: username Password: •••••••• Port: 22

4. Press the "Quickconnect" button to start the connection.

   ⚠ If you get a warning about Saving passwords, choose "Do not save Passwords" and then the OK button.



Remember passwords?

Would you like FileZilla to remember passwords?

When allowing FileZilla to remember passwords, you can reconnnect without having to re-enter a password after restarting FileZilla.

○ Save passwords

● Do not save passwords

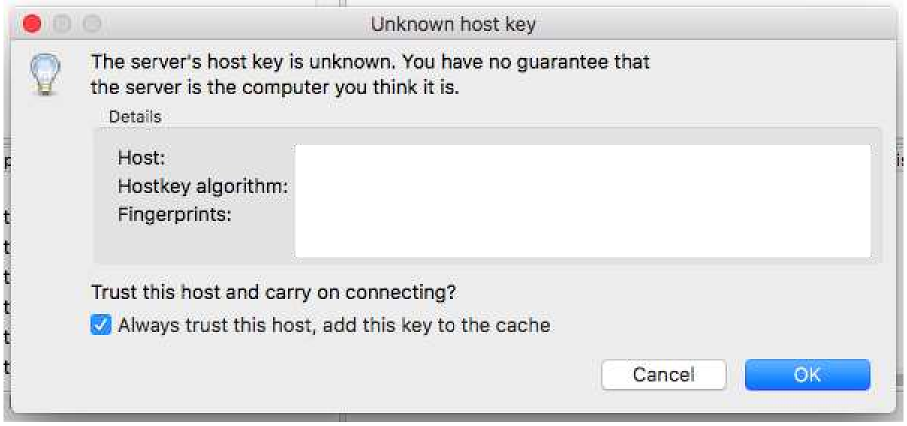
○ Save passwords protected by a master password

Master password:

Repeat password:

A lost master password cannot be recovered! Please thorougly memorize your password.

⚠ If you get a warning about Unknown host key, click the "Always trust..." checkbox and then the OK button.



Unknown host key

The server's host key is unknown. You have no guarantee that the server is the computer you think it is.

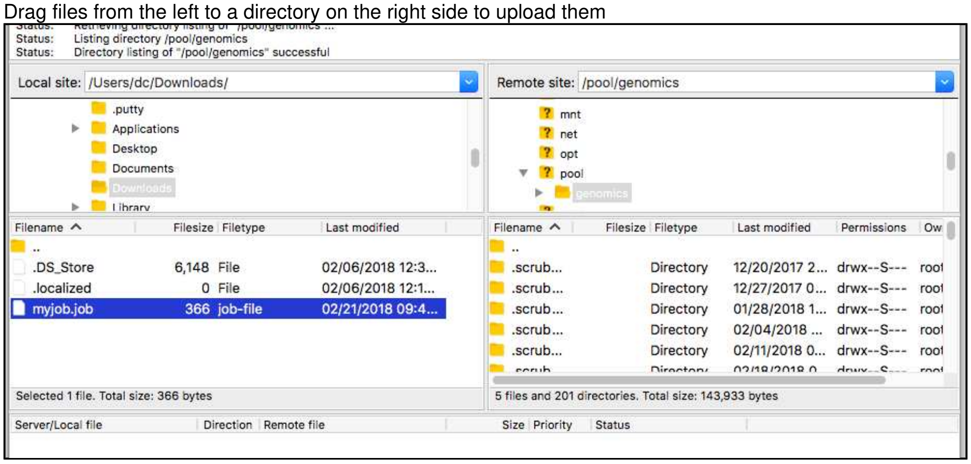
Details

Host:
Hostkey algorithm:
Fingerprints:

Trust this host and carry on connecting?

☑ Always trust this host, add this key to the cache

5. The files listed on the left side of the window are on your local computer, those on the right are on Hydra.
    a. Use the file tree on the left to navigate to the directory with your files to upload
    b. For the destination on Hydra, enter the path of your destination in the "Remote site:" text box
    c. Drag files from the left to a directory on the right side to upload them



```
Status:   Retrieving directory listing of "/pool/genomics"...
Status:   Listing directory /pool/genomics
Status:   Directory listing of "/pool/genomics" successful
```

Local site: /Users/dc/Downloads/

Filename	Filesize	Filetype	Last modified
..			
.DS_Store	6,148	File	02/06/2018 12:3...
.localized	0	File	02/06/2018 12:1...
myjob.job	366	job-file	02/21/2018 09:4...

Selected 1 file. Total size: 366 bytes

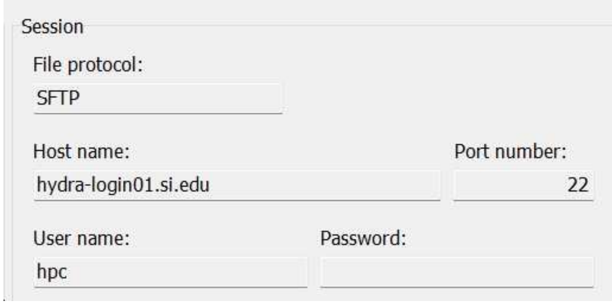
Remote site: /pool/genomics

Filename	Filesize	Filetype	Last modified	Permissions	Ow
..					
.scrub...		Directory	12/20/2017 2...	drwx--S---	root
.scrub...		Directory	12/27/2017 0...	drwx--S---	root
.scrub...		Directory	01/28/2018 1...	drwx--S---	root
.scrub...		Directory	02/04/2018 ...	drwx--S---	root
.scrub...		Directory	02/11/2018 0...	drwx--S---	root
.scrub...		Directory	02/18/2018 0...	drwx--S---	root

5 files and 201 directories. Total size: 143,933 bytes

Using WinSCP

- WinSCP is available for Windows and is a GUI that performs `scp/sftp` and can be downloaded from here.
- Open the WinSCP application and use:
    1. Protocol: SFTP
    2. Host name: `hydra-login01.si.edu` or `hydra-login02.si.edu`
    3. Port: 22
    4. Username: your Hydra username
    5. Password: your Hydra password (or leave blank)



Session

File protocol: SFTP

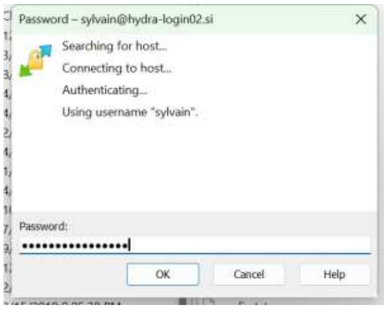
Host name: hydra-login01.si.edu

Port number: 22

User name: hpc

Password:

- Click the Login button, and enter your password:

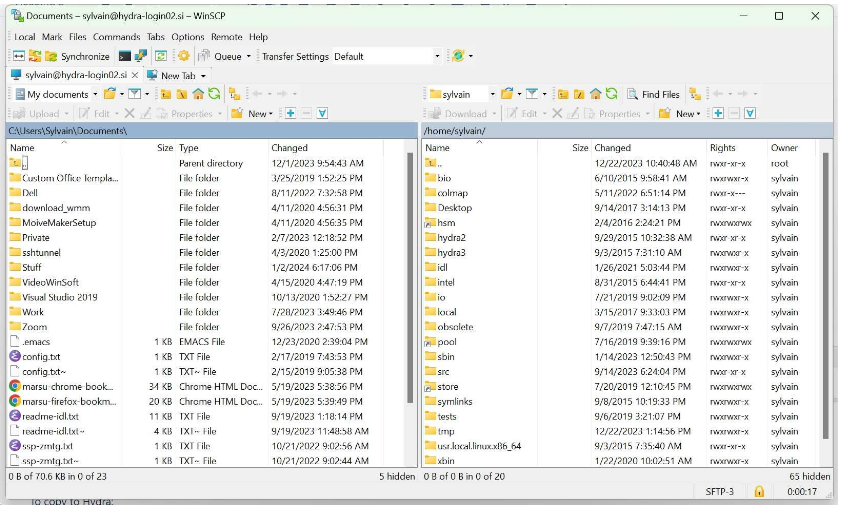


Password – sylvain@hydra-login02.si

Searching for host...
Connecting to host...
Authenticating...
Using username "sylvain".

Password: ••••••••••••••••

- WinSCP can be configured, via its settings, to
  - show two windows: local and remote files ("commander mode"),



Documents – sylvain@hydra-login02.si – WinSCP

C:\Users\Sylvain\Documents\

Name	Size	Type	Changed
..		Parent directory	12/1/2023 9:54:43 AM
Custom Office Templa...		File folder	3/25/2019 1:52:25 PM
Dell		File folder	8/11/2022 7:32:58 PM
download_wmm		File folder	4/11/2020 4:56:31 PM
MoiveMakerSetup		File folder	4/11/2020 4:56:35 PM
Private		File folder	2/7/2023 12:18:52 PM
sshtunnel		File folder	4/3/2020 1:25:00 PM
Stuff		File folder	1/2/2024 6:17:06 PM
VideoWinSoft		File folder	4/15/2020 4:47:19 PM
Visual Studio 2019		File folder	10/13/2020 1:52:27 PM
Work		File folder	7/28/2023 3:49:46 PM
Zoom		File folder	9/26/2023 2:47:53 PM
.emacs	1 KB	EMACS File	12/23/2020 2:39:04 PM
config.txt	1 KB	TXT File	2/17/2019 7:43:53 PM
config.txt~	1 KB	TXT~ File	2/15/2019 9:05:38 PM
marsu-chrome-book...	34 KB	Chrome HTML Doc...	5/19/2023 5:38:56 PM
marsu-firefox-bookm...	20 KB	Chrome HTML Doc...	5/19/2023 5:39:49 PM
readme-idl.txt	11 KB	TXT File	9/19/2023 1:18:14 PM
readme-idl.txt~	4 KB	TXT~ File	9/19/2023 11:48:58 AM
ssp-zmtg.txt	1 KB	TXT File	10/21/2022 9:02:56 AM
ssp-zmtg.txt~	1 KB	TXT~ File	10/21/2022 9:02:44 AM

0 B of 70.6 KB in 0 of 23 5 hidden

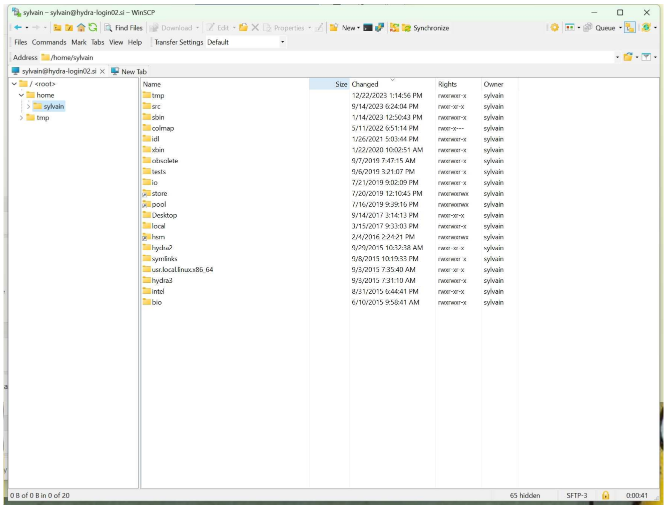
/home/sylvain/

Name	Size	Changed	Rights	Owner
..		12/22/2023 10:40:48 AM	rwxr-xr-x	root
bio		6/10/2015 9:58:41 AM	rwxrwxr-x	sylvain
colmap		5/11/2022 6:51:14 PM	rwxr-x---	sylvain
Desktop		9/14/2017 3:14:13 PM	rwxr-xr-x	sylvain
hsm		2/4/2016 2:24:21 PM	rwxrwxrwx	sylvain
hydra2		9/29/2015 10:32:38 AM	rwxr-xr-x	sylvain
hydra3		9/3/2015 7:31:10 AM	rwxrwxr-x	sylvain
idl		1/26/2021 5:03:44 PM	rwxrwxr-x	sylvain
intel		8/31/2015 6:44:41 PM	rwxr-xr-x	sylvain
io		7/21/2019 9:02:09 PM	rwxrwxr-x	sylvain
local		3/15/2017 9:33:03 PM	rwxrwxr-x	sylvain
obsolete		9/7/2019 7:47:15 AM	rwxrwxr-x	sylvain
pool		7/16/2019 9:39:16 PM	rwxrwxrwx	sylvain
sbin		1/14/2023 12:50:43 PM	rwxrwxr-x	sylvain
src		9/14/2023 6:24:04 PM	rwxr-xr-x	sylvain
store		7/20/2019 12:10:45 PM	rwxrwxrwx	sylvain
symlinks		9/8/2015 10:19:33 PM	rwxrwxr-x	sylvain
tests		9/6/2019 3:21:07 PM	rwxrwxr-x	sylvain
tmp		12/22/2023 1:14:56 PM	rwxrwxr-x	sylvain
usr.local.linux.x86_64		9/3/2015 7:35:40 AM	rwxr-xr-x	sylvain
xbin		1/22/2020 10:02:51 AM	rwxrwxr-x	sylvain

0 B of 0 B in 0 of 20 65 hidden

SFTP-3 0:00:17

  - or only the remote files ("explorer" mode).



Transfer Settings Default

Address /home/sylvain

Name	Size	Changed	Rights	Owner
tmp		12/22/2023 1:14:56 PM	rwxrwxr-x	sylvain
src		9/14/2023 6:24:04 PM	rwxr-xr-x	sylvain
sbin		1/14/2023 12:50:43 PM	rwxrwxr-x	sylvain
colmap		5/11/2022 6:51:14 PM	rwxr-x---	sylvain
idl		1/26/2021 5:03:44 PM	rwxrwxr-x	sylvain
xbin		1/22/2020 10:02:51 AM	rwxrwxr-x	sylvain
obsolete		9/7/2019 7:47:15 AM	rwxrwxr-x	sylvain
tests		9/6/2019 3:21:07 PM	rwxrwxr-x	sylvain
io		7/21/2019 9:02:09 PM	rwxrwxr-x	sylvain
store		7/20/2019 12:10:45 PM	rwxrwxrwx	sylvain
pool		7/16/2019 9:39:16 PM	rwxrwxrwx	sylvain
Desktop		9/14/2017 3:14:13 PM	rwxr-xr-x	sylvain
local		3/15/2017 9:33:03 PM	rwxrwxr-x	sylvain
hsm		2/4/2016 2:24:21 PM	rwxrwxrwx	sylvain
hydra2		9/29/2015 10:32:38 AM	rwxr-xr-x	sylvain
symlinks		9/8/2015 10:19:33 PM	rwxrwxr-x	sylvain
usr.local.linux.x86_64		9/3/2015 7:35:40 AM	rwxr-xr-x	sylvain
hydra3		9/3/2015 7:31:10 AM	rwxrwxr-x	sylvain
intel		8/31/2015 6:44:41 PM	rwxr-xr-x	sylvain
bio		6/10/2015 9:58:41 AM	rwxrwxr-x	sylvain

0 B of 0 B in 0 of 20 65 hidden SFTP-3 0:00:41

  - WinSCP layout can be re-arranged, so your may look somewhat differently.
- Navigate via the GUI to the location of the source files and to the location of the destination, or type the full path in the "Address" and then
  - drag and drop the files you want to copy, selecting a slew of them, or only one at a time.
- In the "explorer" mode, use Windows' File Explorer to navigate your local disk.

Using SCP

The command `scp` is available on any Linux, Windows (as of Windows 10) or Mac machine.

- To access it from a Mac, start Terminal to get a Unix prompt.
- On Windows open a Command Prompt or PowerShell session from the Start menu.

Use the command `cd` to go to where the files you want to copy are.

You can copy a local file with:

```
scp myfile remuser@hydra-login01.si.edu:/path/to/dest/
```

where:

- `myfile` is the file name you want to copy on your (local) machine,
- `remuser` is your Hydra username,
- `/path/to/dest` is the directory specification where you want the file copied (it must exists), like `/pool/genomics/username/big/stuff`

To copy to Hydra:

You can copy multiple files with:

```
scp myfile1 myfile2 remuser@hydra-login01.si.edu:/path/to/dest/
scp myfile* remuser@hydra-login01.si.edu:/path/to/dest/
```

The command scp always copy the file(s), the option '`-p`' (like in `scp -p`) will preserve the date information of the file(s) copied.

And to copy from Hydra,:

To copy from Hydra

```
scp remuser@hydra-login01.si.edu:/path/to/location/myfile .
```

or, if you want to use a wildcard in the file name specification, use quotes like in:

You can use wildcards (remote location), as follows:

```
scp 'remuser@hydra-login01.si.edu:/path/to/location/myfile*' .
```

To learn more about `scp`, read the manual page (`man scp` on Linux or Mac systems). The flag `-p` will preserve the file(s) timestamps, like in

```
scp -p myfile* remuser@hydra-login01.si.edu:/path/to/dest/
```

Using RSYNC

The command `rsync` is available on any Linux or Mac machine, it is not available on Windows systems. To access it from a Mac, start a terminal to get a Unix prompt.

Rsync synchronizes files between two hosts (machines), so it will not copy files that exists already and are up-to-date: it will copy a file if the one on the source location is older than the one on the destination.

Use the command `cd` to go to where the files you want top copy are. To copy to Hydra:

Examples of copying to Hydra:

```
rsync myfile remuser@hydra-login01.si.edu:/path/to/dest/
rsync myfile1 myfile2 remuser@hydra-login01.si.edu:/path/to/dest/
rsync myfile* remuser@hydra-login01.si.edu:/path/to/dest/
```

And to copy from Hydra:

From Hydra to your local machine:

```
rsync remuser@hydra-login01.si.edu:/path/to/location/myfile .
rsync 'remuser@hydra-login01.si.edu:/path/to/location/myfile*' .
```

The difference with `scp`, is that `rsync will only copy what is new, so use:`

so you can do:

```
rsync * remuser@hydra-login01.si.edu:/path/to/dest/
```

to synchronize the content of the current working directory.

Use the option '`-n`' (like in `rsync -n`) to check what `rsync` will do (it will list what will be copied, but not do it, aka 'dry run').

Three more useful options are:

- `-a` – archive mode (equals `-rlptgoD`). It is a quick way of saying you want recursion and want to preserve almost everything (with -H being a notable omission).
- `-z` – compress file data during the transfer (speed up)
- `-v` – verbose mode

that can be combined as `rsync -azv.` To learn more about `rsync`, read the manual page (`man rsync`).

Using SFTP or LFTP

You can also use the command `sftp` or `lftp` to copy files, it is also available on any/most Linux or Mac machine and recent releases of Windows 10.

To access it from a Mac, start a terminal to get a Unix prompt, on Windows use the Start menu to open a Command Prompt or PowerShell session.

Use the command `cd` to go to where the files you want top copy are.

For example:

```
sftp remuser@hydra-login01.si.edu
sftp> cd /path/to/dest/
sftp> put myfile
sftp> get myresults
sftp> exit
```

The main `sftp` commands are cd, lcd, put and get:

- cd - change the directory on the remote host,
- `lcd` – change the directory on the local host,
- `put` - copy a file from the local host to the remote one,
- `get` - copy a file from the remote host to teh local one.

To learn more about `sftp` or `lftp`, read the manual page (`man sftp, man lftp` on Linux or Mac systems).

Last updated 05 Jan 2024 MK/PF/SGK.

The QSub Generator

⚠ **Note:**

The material on this page is part of the Quick Start Guide and is not exhaustive. For more details, please see the Reference Pages.

Introduction

When submitting jobs to the cluster, you need to tell the job scheduler details about your run (run time, memory needed, CPUs needed etc.) AND the actual commands that you will be running for your analysis.

💡 You can use the QSub Generator for both. 💡

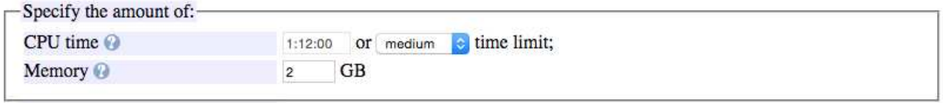
How to Use QSubGen

To use the QSub Generator:

1. Open the QSub Generation Utility web page.
   ⚠ This utility works best on Chrome or Firefox. Safari is not recommended.

   👍 The question mark icons by each entry in the form provide useful information on each item including formatting: ?
   👍 The example below shows how to setup a RAxML job that will use a single CPU to run bootstraps and a best tree analysis.

2. *Time and memory*

   Specify the amount of:

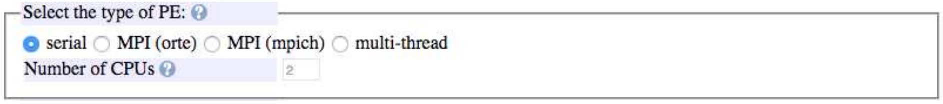
   CPU time ? 1:12:00 or medium time limit;

   Memory ? 2 GB

   Time: Enter the maximum time allowed for your job in "CPU time" by entered an exact time in the text box or using the drop-down menu.
   ⚠ After this time is reached, your job will be terminated.
   👍 The drop-down menu will update the time in the text box to show the length of the pre-defined times.
   👍 CPU time is the amount of time that the processor is using, not the "wall clock" time which is the elapsed time since the job started.
   👍 For jobs using more than one CPU, the amount of CPU time granted to the job (before it gets terminated) is the product of number of CPUs and the time allocated, while the allowed wall clock time is not multiplied by the number of CPUs.

   Memory: is the maximum memory that your run will use.
   👍 Some programs will have memory use estimators. In this RAxML example you can use a memory estimator here.
   ⚠ The value in the memory box is per-CPU.

3. *Parallel Environment*

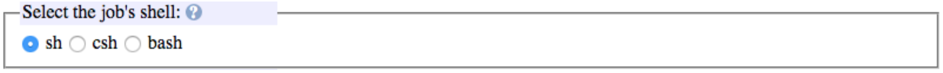
   Select the type of PE: ?

   ● serial ○ MPI (orte) ○ MPI (mpich) ○ multi-thread

   Number of CPUs ? 2

   In this section you define how many CPUs your job will use. The software documentation will indicate if it is limited to one CPU (serial), multiple CPUs spread across servers in the cluster (MPI), or multiple CPUs residing on one server (multi-thread).
   In this example we are only using one CPU (serial).
   👍 Refer to the software documentation for information about parallel use supported. Software documentation through the module system (described below) may also give more information.

4. *Shell*

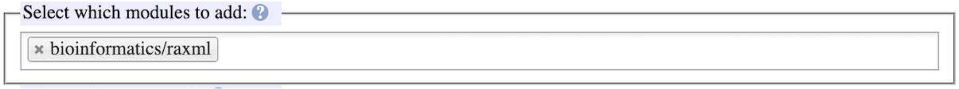
   

   Select the job's shell: ?

   ● sh ○ csh ○ bash

   For novice bioinformatics users, we recommend you keep this with sh.
   A shell is the program on the computer that interprets your commands. Consult Wikipedia article for more info.

5. *Modules*

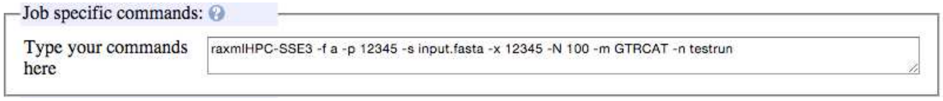
   

   Select which modules to add: ?

   × bioinformatics/raxml

   Hydra uses the modules system to load programs that use the command line. There is a module for each program on the server which includes: where the program is located, other programs it depends on, some help information about starting the program on the cluster. The module may refer to a specific version, or in this case "bioinformatics/raxml" will always refer to the newest version of RAxML installed on the cluster. A web-based list of available modules is available.
   👍 You can start typing a program name to see a list of modules that match.

6. *Commands*

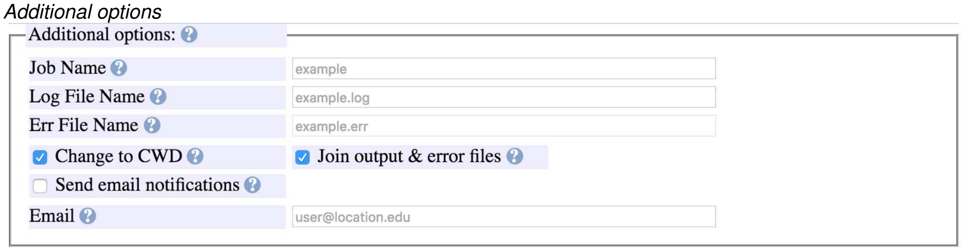


Job specific commands:

Type your commands here: raxmlHPC-SSE3 -f a -p 12345 -s input.fasta -x 12345 -N 100 -m GTRCAT -n testrun

In this section you put your commands that will be run on the cluster. You start with the name of executable, `raxmlHPC-SSE3` in this case, and include all command line options and references to data files.
👍 You can find the name of executables by logging into hydra via the terminal and typing `module help bio/raxml/latest` (replacing with the module file you will be using).
👍 For MPI jobs, start with `mpirun np $NSLOTS` followed by the executable name and command line options.

7. *Additional options*

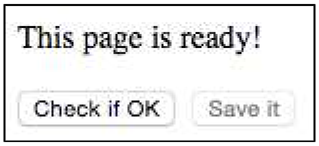


Additional options:

Job Name: example
Log File Name: example.log
Err File Name: example.err
☑ Change to CWD ☑ Join output & error files
☐ Send email notifications
Email: user@location.edu

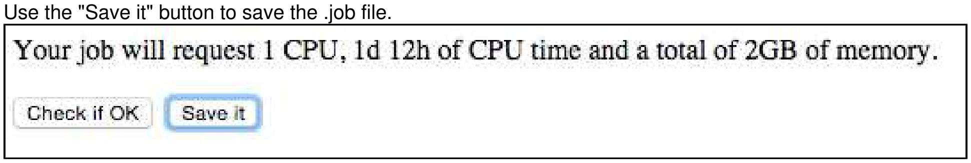
In this section you give some more information that will be used to run your job and log the output.
Job Name: Name the job will be called in the cluster job list (no spaces allowed)
Log File Name: File where stdout will be sent. Will be filled in automatically with the Job Name.
Error File Name: File where stderr will be sent if output and error files are *not* split. Will be filled in automatically with the Job Name.
Change to CWD: Always check. Will put log files in the directory from which you start your job (your current working directory).
Join stderr & stdout: Recommend to check. Will put all program output into one file (named in "Log File Name").
Send email notifications: Emails will be send when your job starts and ends.
Email: Where notifications are sent. This can be any email account.

8. *Check file and download*
Press the "Check if OK" button to confirm that script (in the gray box) was generated correctly.



This page is ready!

Use the "Save it" button to save the .job file.



Your job will request 1 CPU, 1d 12h of CPU time and a total of 2GB of memory.

👍 Above the Save it button is the time and total RAM being requested for your job.

9. Upload to Hydra
Upload this .job file that is generated to Hydra in the /scratch, /pool or /data folder to be used for your job.

Last update 13 May 2024 SGK/MPK

Submitting a Job

⚠ **Note:**

The material on this page is part of the Quick Start Guide and is not exhaustive. For more details, please see the Reference Pages.

To submit your job you will need to have created your job file and have your job file and data files copied to Hydra.

Follows the following steps:

1. Transfer job file and data files to Hydra. All data files should go into your folder in `/pool` , `/scratch` or `/data`.
   👍 It is recommended for beginner users to create a new directory for each analysis. Create a directory using `Filezilla` or `mkdir` from the command line.

2. Log into Hydra via a terminal and navigate to the directory with your data files.

   ⚠ Throughout this tutorial replace `username` your Hydra username, the name you log into Hydra with, typically the same as your SI email account name (all lower case).

```
[username@hydra-login01 ~]$ cd /pool/genomics/username/raxmltest/
[username@hydra-login01 raxmltest]$
```

3. Use `ls` to confirm your job and data files are present and use `cat` (or another text file viewer like `more` or `less`) to inspect your job file.

```
[username@hydra-login01 raxmltest]$ ls
input.fasta  raxmltest.job
[username@hydra-login-1 raxmltest]$ cat raxmltest.job
# /bin/sh
# ----------------Parameters---------------------- #
#$ -S /bin/sh
#$ -q mThC.q
#$ -l mres=2G,h_data=2G,h_vmem=2G
#$ -cwd
#$ -j y
#$ -N raxmltest
#$ -o raxmltest.log
#$ -m abe
#$ -M user@si.edu
#
# ----------------Modules------------------------- #
module load bioinformatics/raxml
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
#
raxmlHPC-SSE3 -f a -p 12345 -s input.fasta -x 12345 -N 100 -m GTRCAT -n testrun
#
echo = `date` job $JOB_NAME done
```

4. Submit your job to the cluster with the `qsub` command.

```
[username@hydra-login01 raxmltest]$ qsub raxmltest.job
Your job 825184 ("raxmltest") has been submitted
```

5. Check your job status with the `qstat` command and `ls`

   ⚠ Errors in the job file will often cause the job to fail quickly, so check `qstat` during the first few minutes of a job run to confirm that it was submitted successfully. If nothing is displayed then you have no current jobs.

```
[username@hydra-login01 raxmltest]$ qstat
job-ID  prior   name       user         state submit/start at     queue                          slots
ja-task-ID
-----------------------------------------------------------------------------------------------------------
--------
 825184 0.55500 raxmltest  username     r     09/22/2015 10:18:01 mThC.q@compute-81-01.cm.clust       1
[username@hydra-login01 raxmltest]$ ls
input.fasta  RAxML_bootstrap.testrun  RAxML_info.testrun  raxmltest.job  raxmltest.log
```

👍 You can now logout of Hydra with the `exit` command and log back in later to check your job progress

6. When your job is complete the job will no longer appear in qstat and you will receive an email if you configured your job file to send email notifications. Inspect the output files to confirm that job was successful.

```
[username@hydra-login01 raxmltest]$ qstat
[username@hydra-login01 raxmltest]$ tail raxmltest.log
Best-scoring ML tree written to: /pool/genomics/username/raxmltest/RAxML_bestTree.testrun
Best-scoring ML tree with support values written to: /pool/genomics/username/raxmltest
/RAxML_bipartitions.testrun
Best-scoring ML tree with support values as branch labels written to: /pool/genomics/username/raxmltest
/RAxML_bipartitionsBranchLabels.testrun
Overall execution time for full ML analysis: 149.352777 secs or 0.041487 hours or 0.001729 days
= Tue Sep 22 10:20:30 EDT 2015 job raxmltest done
```

7. Transfer the output files to your computer or continue with further analysis of the dataset on Hydra.
👍 It is recommended to compress your output before transferring to your local computer

```
[username@hydra-login01 raxmltest]$ cd ..
[username@hydra-login01 username]$ tar -czvf raxmltest.tar.gz raxmltest
raxmltest/
raxmltest/raxmltest.job
raxmltest/input.fasta
raxmltest/raxmltest.log
raxmltest/RAxML_info.testrun
raxmltest/RAxML_bootstrap.testrun
raxmltest/RAxML_bestTree.testrun
raxmltest/RAxML_bipartitions.testrun
raxmltest/RAxML_bipartitionsBranchLabels.testrun
```

⚠️ Storage on Hydra is not for long-term data storage. Please transfer and remove your data files and output when your analyses are complete.

Last updated 13 Oct 2021 SGK/MPK

Reference Pages

The references pages are organized as follows:

- Introduction
- Hardware
    - Login and Head Nodes
    - Compute Nodes
    - Network: Ethernet and InfiniBand
    - Storage Systems (Disks)
- Submitting Jobs
    - Conceptual Examples
    - Serial Jobs
    - Parallel Jobs
    - Job Arrays
    - Available Queues
    - Resource Limits
    - Job Monitoring
    - Help Choosing a Queue
    - Help Writing a Job Script
    - Where to Find Examples
    - Running Jupyter on Hydra
- Cluster Monitoring
- Disk Space and Usage
    - Disk Space Configuration
    - How to Copy Files to/from Hydra
    - How to Check Disk & Quota Usage
    - How to Recover Old or Deleted Files using Snapshots
    - Scrubber and How to Request Scrubbed Files to be Restored
    - How to Use Local SSD Space
    - How to Use NAS Storage and the I/O Queue
    - How to Use "bigmp" - Access to Large Temporary Disk Space
- Software
    - The "module" Command
    - Compilers & Libraries & MPI or Multi-threaded Programs
    - Packages/Tools
        - IDL, FL, Java, Julia, Matlab, Python, R
        - Conda: Anaconda & Miniconda
            - Installing Miniconda on Hydra
            - Using Conda
    - Genomics Software
    - Bioinformatics Software and Modules
- Additional Tools
- How to Use GPUs
- How to Use Containers

Let us know if you find typos or mistakes.

Last updated 13 May 2024 SGK

Introduction

1. Preamble

- The cluster, known as `Hydra`, is made of
  1. one front-end node,
  2. two login nodes,
  3. a queue manager/job scheduler, and
  4. a set of compute nodes.

- To access the cluster you must log on one of the two login nodes, using `ssh`:
  - `ssh hydra-login01.si.edu`
    or
  - `ssh hydra-login02.si.edu`

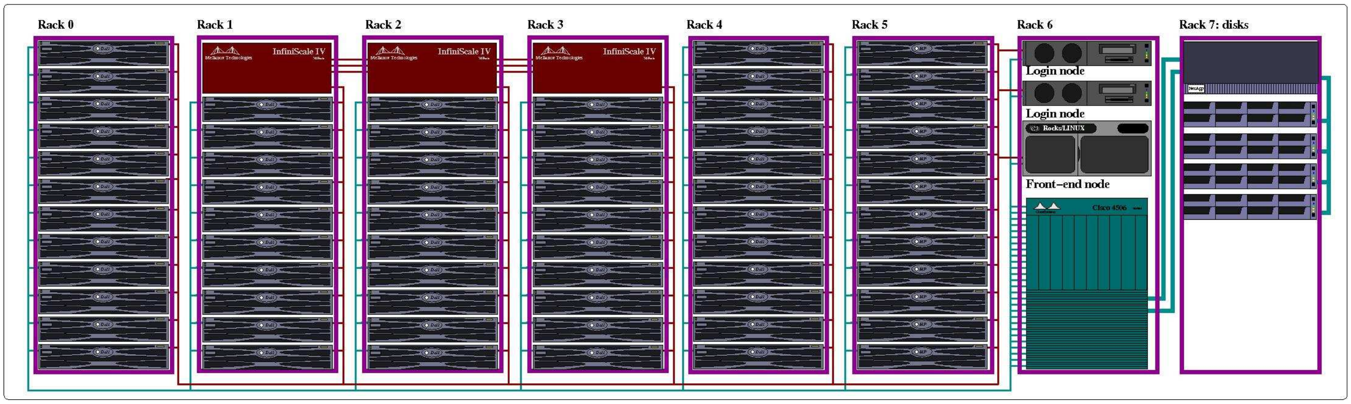
- From either login node you submit and monitor your job(s) via the queue manager/scheduler.
- The job scheduler on Hydra is the Univa Grid Engine, simply GE or UGE (Univa was recently acquired by Altair).

- The Grid Engine runs on the front-end node (`hydra-7.si.edu`), hence the front-end node **should not be used** as a login node.
  - There is no reason for users to ever have to log on `hydra-7`.

- All the nodes (login, front-end and compute nodes) are interconnected
  - via Ethernet (at 10Gbps, aka 10GbE), and
  - via InfiniBand (at 40-100Gbps, aka IB).

- The disks are mounted off different types of dedicated devices:
  1. a two nodes NetApp filer for `/home`, `/data`, and `/pool` (via 10GbE),
  2. a two NSDs GPFS for `/scratch` (via IB),
  3. a low cost NAS for `/store` (via 10GbE), a near-line storage only available on some nodes,
  4. some partitions on the DAMS Qumolo storage system are accessible via `/qnas` on some nodes.

The following figure is a schematic representation of the cluster:



Note that it does not represent the actual physical layout.

2. Access to the Cluster

- To access the `Hydra` cluster you will need
  1. an account;
  2. a secure connection to log in either login nodes; and
  3. a `ssh` client (i.e., a program compatible with the secure shell protocol, aka `ssh`).

Requesting Accounts

- *SAO users* should request an account following the instructions posted on the CF's web page.
  Accounts on Hydra are separate from `CF`, `HEA`, `SI`'s Active Directory, or `VPN` accounts.

- *Non-SAO users* should request an account by submitting a request through the SI Service Portal.
  Accounts on Hydra are separate from `SI`'s Active Directory, or `VPN` accounts.

Secure Connection

- You can connect to the login nodes using `ssh` *only from a trusted machine*.
- Trusted machines are computers connected via a hardwired network link to SI's or SAO's network.
- Any other computer, like your laptop, a *"self-managed"* computer, a computer at another institution, or a computer connected via WiFi, must be *authenticated* using VPN.
- Information on VPN authentication is available elsewhere:
  - here for SI (via OCIO), and
  - here for SAO (via CF/HEA).
- SAO users can also connect to the cluster using `ssh` directly from *trusted* machine (like an ITS-managed machine).

SSH Clients

1. Linux users, use `ssh [<usenname>]@<host>`, where
   - `<username>` is your username on hydra if it is different from the computer you are `ssh`'ing from
   - <host> is either of the login nodes name, i.e., `hydra-login01.si.edu` or `hydra-login02.si.edu`

2. MacOS users can use `ssh [<usenname>]@<host>` as explained above from a `Terminal`.

   - Open the Terminal app by going to `/Applications/Utilities` and finding `Terminal`.
   - You can get to the `Utilities` folder by going to the `Go` menu in the `Finder` and choosing `Utilities`.

3. PC/Windows users need to install a `ssh` client. Public domain options are:
   - Window's `ssh` from the PowerShell (for recent versions of Windows)
   - PuTTY
   - Cygwin and use `ssh [<usenname>]@<host>` (note that Cygwin includes a X11 server.)

See also the Comparison of SSH clients Wikipedia page.

3. Using the Cluster

- To run a job on the cluster you will need to:
  1. install the required software, unless it is already installed;
  2. copy the data your job needs to the cluster; and
  3. write at least one (minimal) script to run your job.

Indeed, the login nodes are for interactive use *only* like editing, compiling, testing, checking results, etc.... and, of course, submitting jobs.

The login nodes are not compute nodes (neither is the head node), and therefore they should not be used for actual computations, except short debugging interactive sessions or short ancillary computations.

The compute nodes are the machines (aka hosts, nodes, servers) on which you run your computations, by submitting a job, or a set of jobs, to the queue system (the Grid Engine or GE).

This is accomplished via the `qsub` command, from a login node, and using a job script. You can also request an interactive session on a compute node using `qrsh.`

> ⚠ **Please**
>
> Do not run on the login or front-end nodes and do not run jobs *out-of-band*, this means:
>
> - do not log on a compute node to manually start a computation, always use `qsub;`
> - do not run scripts/programs that spawn additional tasks, or use multi-threads, unless you have requested the corresponding resources;
> - if your script runs something in the background (it shouldn't), use the command `wait` so your job terminates only when all the associated processes have finished;
> - to run multi-threaded jobs, read and follow the relevant instructions;
> - to run parallel jobs (MPI), read and follow the relevant instructions;
>   You don't start MPI jobs on the cluster the way you do on a stand alone computer (or laptop.)

Remember that

- you will need to write a script, even if trivial, to submit a job (there is a tool to help you do that);
- you should optimize your codes with the appropriate compilation flags for production runs;
- you will most likely need to specify multiple options when submitting your jobs, via the command `qsub;`
- things do not always scale up: as you submit a lot of jobs (in the hundreds), that will run concurrently (at the same time), ask yourself:
  1. is there name space conflict? all the jobs should not write to the same file, they should not have the same name;
  2. what will the resulting I/O load be? do all the jobs read the same file(s), do they write a lot of (useless?) stuff?;
  3. how much disk space will I use? Will my job fill up my allotted disk space? Is the I/O load high compared to the CPU load?

4. how much CPU time and memory does my job need? Jobs run in queues, these queues have limits.

- *Check-pointing*: computers do crash, networks go down and jobs get killed when they exceed limits, so
  - whenever possible, and especially for long jobs, you should save intermediate results so you can resume a computation from where it stopped.
    - this is known as check-pointing.
  - If you use some third party tool, verify if it uses check-pointing and how to enable it.
  - If you run your own code, you should include check-pointing for long computations.

The cluster is a shared resource: when a resource gets used, it is unavailable to others, hence:

- clobbering disk space prevents other from using it:
  1. trim down your disk usage;
  2. cleanup your disk usage after your computation;
  3. the disks on the cluster are not for long term storage;
  4. move what you don't use on the cluster back to your "home" machine.
- Running un-optimized code wastes CPU cycles and effectively delays/prevents other users from running their analyses.
- Reserving more memory than you need will *effectively* delay/prevent other users from using it.
- Fair use: we have implemented resource limits, do not bypass these limits.
  - If needed, feel free to contact us to review how these limits impact you.

4. Software Overview

The cluster runs a Linux distribution that is specific to clusters. We use BrightCluster (v10.0) to deploy Rocky 8.9 (Green Obsidian).

- As for any Unix system, you must properly configure your account to access the system resources.
  Your `~/.bash_profile`, or `~/.profile`, and or `~/.cshrc` files need to be adjusted accordingly.
- The configuration on the cluster is different from the one on the `CF`- or `HEA`-managed machines (for SAO users).
  We have implemented the command `module` to simplify the configuration of your Unix environment (or `conda)`.
- You can look in the directory `~hpc/` for examples of configuration files (with `ls -la ~hpc`).

Available Compilers

1. GNU compilers (`gcc, g++, gfortran, g90`)
2. Intel compilers and the Cluster Studio (debugger, profiler, etc: `ifort, icc`, ...)
3. NVIDIA compilers (PGI was acquired by NVIDIA, `nvfortran`, `nvcc`, ...)

Available Libraries

1. MPI, for GNU, PGI/NVIDIA and Intel compilers, w/ IB support;
2. the libraries that come with the compilers;
3. GSL, BLAS, LAPACK, etc.

Available Software Packages

1. We have 128 run-time licenses for IDL, (FL is available too).
2. Tools like MATLAB, JAVA, PYTHON, R, Julia, etc. are available; and
3. the Bioinformatics and Genomics support group has installed a slew of packages.

Refer to the Software pages for further details. Other software packages have been installed by users, or can be installed upon request.

5. Support

The cluster is located in Herndon, VA and is managed by ORCS/OCIO (Office of Research Computing Services/Office of the Chief Information Officer).

The cluster is supported by the following individuals:

- DJ Ding (DingDJ@si.edu), the the system administrator (at OCIO, Herndon, VA).
  - As the sys-admin, he is responsible to keep the cluster operational and secure.

- Alex White (WhiteAE@si.edu) - Research Data Scientist, OCIO Data Science Lab (Washington, D.C.);
- Matthew Kweskin (KweskinM@si.edu) - NMNH/L.A.B., IT specialist (Washington, D.C.).
- Mike Trizna (TriznaM@si.edu) - Data Scientist, OCIO Data Science Lab, Washington, D.C.).
- Vanessa Gonzalez (GonzalezV@si.edu) - NMNH/GGI, biologist (Washington, D.C.).
- Sylvain Korzennik (hpc@cfa.harvard.edu), Astronomer at SAO (Cambridge, MA);
  - he is the primary support person for SAO/CfA users.

Support is also provided by other OCIO staff members (networking, etc...).

Simple Rules

- *For sys-admin issues:* (something is not working any more, etc):
    - *All users* should contact DJ (and Rebecca & Sylvain) at SI-HPC-Admin@si.edu
- *For application support:* (how do I do this?, why did that fail?, etc)*:*
    - *SAO users* should contact Sylvain, *not* the `CF` or `HEA` support sys-admin groups, at hpc@cfa.harvard.edu,
    - *Non-SAO users* should contact Alex, Mike, Vanessa & Matt at SI-HPC@si.edu
- *Password problems:* go to the self-serve password page.
- Please use these email addresses to let the SI/HPC support team address your issues as soon as possible,
    - rather than emailing individuals directly.

Mailing List

A mailing list, called `HPPC-L` on SI's listserv (i.e., at `si-listserv.si.edu`, hence `HPCC-L@si-listserv.si.edu`) is available to contact all the users of the cluster.

This mailing list is used by the support people to notify the users of important changes, etc.

- You can also use this mailing list to communicate with other cluster users, share ideas, ask for help with cluster use problems, offer advice and solutions to problems, etc.
- The mailing list is read by the cluster sysadmin and support staff as well as by other cluster users.

To email to that list you must log to the `listserv` and post your message. Note that:

- replies to these messages are by default broadcast to the entire list; and
- you will need to set up a password on this `listserv` the first time you use it (look in the upper right, under "Options").

Last updated 13 May 2024 SGK.

Hardware Description

1. Introduction

As of May 2024, `Hydra-7` consists of:

- one head node and two login nodes;
- ~70 compute nodes that adds up to ~6,000 CPUs & 8 GPUs, ~45 TB memory;
- a set of 10Gbps network switches, all the nodes are on 10GbE;
- an InfiniBand (IB) director switch (144 ports, expandable to 256, at 100Gbps)
- a 3 tiers storage (disks) configuration:
    - a two nodes NetApp controller (FAS8300) with 8 shelves (total ~900TB):
        - a dedicated device that provides disk space to the cluster, i.e. to all the nodes using NFS (10GbE).
    - a GPFS system with two dedicated NSDs (total ~2PB):
        - a high performance general parallel file system (GPFS, aka IBM Spectrum Scale), using the InfiniBand for I/O.
    - Two NAS systems for near-line storage (total ~1.7PB):
        - a slower, more cost-effective storage available only on some nodes.
- all the nodes (head, login and compute nodes) are connected to the IB switch (except two).

Last updated 13 May 2024 SGK

Login and Head Nodes

The Head Node: hydra-7.si.edu

- manages the cluster;
- runs the job scheduler (the Grid Engine, aka UGE); and
- starts jobs.

It should never be accessed by users, except if directed by support staff for special operations.

The Login Nodes: hydra-login0[12].si.edu

- These are the computers available to the users to access the cluster:
    - they are currently 48 cores 128GB Dell R730 servers.
    - do not run your computations on the login nodes.

You can use either node, depending on the node load.

Last updated 13 May 2024 SGK

Compute Nodes

The compute nodes are named `compute-NN-MM`, where NN and MM are two numbers.

- These are the nodes (aka servers, hosts) on which jobs are being run, by submitting jobs to the scheduler, via `qsub`.
- A couple of nodes are dedicated for
    - interactive use (`qrsh`), and
    - I/O queue (for access to `/store`).
- Do not `ssh` to the compute node to start any computation "*out of band*" (we'll find and terminate them).

As of May 2024, we have deployed some 70 compute nodes that adds up to ~6,000 CPUs, 8 GPUs, ~45 TB memory as follows;

#	Cores/node	Mem/node	Model	Name	Note
1	72	512GB	R750XA	`compute-50-??`	4x L40s GPUs
16	40	384GB	R640	`compute-64-??`	
2	32	512GB	R640	`compute-64-??`	
29	64	512GB	R6515	`compute-65-??`	
5	128	756GB	R7525	`compute-75-??`	
2	128	1,024GB	R7525	`compute-75-??`	
2	192	1,536GB	R7625	`compute-76-??`	
12	128	1,024GV	R7625	`compute-76-??`	
2	20	128GB	R790	`compute-79-??`	2x GV100GL GPUs
1	112	896GB	R840	`compute-84-??`	
1	64	512GB	R930	`compute-93-??`	
3	72	760GB	R930	`compute-93-??`	
1	96	2,048GB	R930	`compute-93-??`	

Last updated 13 May 2024 SGK

Network: Ethernet and InfiniBand

All the nodes (i.e., the compute nodes, the login nodes, and the head node) are interconnected using not only the regular 10GbE network (Ethernet), but also via a high-speed, low latency, communication fabric, known as the InfiniBand (IB):

- The IB switch is capable of a 100Gbps transfer rate, although the older nodes have IB card capable of 40Gbps only.
- The GPFS storage use the Infiniband fabric for for its I/O.
- To use the IB for message passing (MPI) you must
    - build the executable the right way, and
    - specify that you want to use the IB in your job script.
    - We have modules to do precisely that.
- A MPI program will not use *by default* the IB for message passing - you need to build it *right* to make it use the IB.

Last updated 08 Oct 2021 SGK

Storage Systems (Disks)

The disk space available on the cluster is mounted off two dedicated devices (NetApp and GPFS); a third one (NAS) is not accessible from all the compute nodes.

The NAS is only accessible from the login, interactive and I/O nodes (hence it is a near-line storage system).

The available public disk space is divided in several area (aka partitions):

- a small partition for basic configuration files and small storage, the `/home` partition,
- a set of medium size partitions, the `/data` partitions,
- a set of large partitions, the `/pool` partitions,
- a set of very large partitions, the `/scratch` partitions,
- a set of medium size, low-cost, partitions, the `/store` partitions.

It should be used as follows:

Name	Typical Use	System	Size[*]
`/home`	For your basic configuration files, scripts and job files: - low quota limit but you can easily recover old stuff, - backup to AWS Glacier for disaster recovery (DR)	NetApp	20TB
`/data/public`	For important but small files like final results, config files, etc - medium quota limit, you can easily recover old stuff, - but when deleting files disk space is not released right away. - backup to AWS Glacier for DR	NetApp	105TB
`/pool/public`	For the bulk of your storage - high quota limit, and disk space is released right away when deleting files.	NetApp	190TB
`/scratch/public`	For the bulk of your large storage - faster storage, - high quota limit, and disk space is released right away when deleting files.	GPFS	400TB
`/store/public`	For near-line storage.	NAS	175TB

[*]: These sizes are only indicative, as we adjust them as needed.

Note

- We impose quotas (limit on how much can be stored on each partition by each user) and we monitor disk usage;
  - `/home` should not be used for storage of large files, use `/pool` or `/scratch` instead;
  - `/data` is best to store things like final results, code, etc.. (important but not too large);
- We implement an automatic scrubber: old stuff get deleted to make space,
  - files older than 180 days and empty directories on `/pool/public` or `/scratch/public` will scrubbed.
- None of the disks on the cluster are for long term storage:
  - please copy your results back to your "home" computer and
  - delete what you don't need any longer.
- Once you reach your quota you won't be able to write anything on that partition until you delete stuff.
- Some compute nodes also have local SSDs (solid state disks, see How to Use Local SSD Space).

ℹ A complete description is available at the Disk Space Configuration page.

Last updated 13 May 2024 SGK

Submitting Jobs

1. Introduction

Most computations on Hydra are run in batch mode using a job scheduler (aka workload manager).

Hydra uses the Univa Grid Engine (GE or UGE) as the job scheduler:

- Jobs are submitted from either login node to the job scheduler using the command `qsub` and a job file;
- submitted jobs may wait in the queue:
    - until the requested resource(s) is/are available, or
    - if a user has reached a resource usage limit, until that limit has cleared.
- The scheduler will eventually run each job, starting it on one or several compute nodes:
    - the job will run in batch, not interactive, mode;
    - it is the scheduler that selects on which compute(s) node to run a job on, and
    - if the job exceeds a limit, like it uses too much memory, or consumes too much CPU time, the scheduler will kill the job.

To run a computation (a job) on Hydra users must write a list of instructions, that specifies the step(s) needed to perform the said computation and if needed pass instructions (aka directives) to the job scheduler as to which resources are required to complete the said computation (like the amount of memory, CPU time, number of CPUs, etc.).

These steps are typically written in a file, aka the job script, while the directives are either passed as options to the `qsub` command or included in the job script, as embedded directives.

A job is thus submitted with the command `qsub`, with the required options (or embedded directives) followed by the name of the file containing the job script.

The different types of jobs are:

- serial jobs: computations that use only one CPU;
- parallel jobs: computation that use more than one CPU (either all on the same node, using multi-threading, or distributed across nodes, using message passing)
- job arrays: a set of similar computations, aka tasks, that use a single unique job script file and a number that identifies each task to be performed.

💡 A few compute nodes are set aside for interactive use, consult the section on using the interactive queue.

ℹ The Available Queues page describes in detail the available queues.

Every job running on the cluster is started in a queue.

- The GE will select a queue based on the resources requested and the usage in each queue.
- If you don't specify the right queues or the right resource(s), your job will either
    - not get queued,
    - wait forever and never run, or
    - start and get killed when it exceeds one of the limit of the queue it was started in.

The set of available queues is a matrix of queues:

- Four sets of queues: a high-CPU and a high-memory set of queues, complemented by a very-high-memory restricted queue and *special* queues.
- The high-CPU and a high-memory sets of queues have different time limits: short, medium, long and unlimited.

Type	Description
high-CPU	for serial or parallel jobs that do not need a lot of memory,
high-memory	for serial or multi-threaded parallel jobs that require a lot of memory,
very-high-memory	reserved for jobs that need a very large amount of memory,
other	for interactive use or projects that need special resources (GPUs, I/O, etc).

Notes

1. A job will run in a queue. Each queue has some form of limit:
   - in most cases, a job won't be allowed to run forever, nor grab as much memory as it may want to.
   - How to specify resources and what queues to use is explained at the Available Queues page.

2. There is some overhead in starting a job, so it is bad practice to submit a large number of very small jobs.
   While you may find it convenient to submit 10,000 five-minute-long jobs, the system will end up taking as much time starting the jobs as the jobs will take to run.
   As a precaution to prevent clobbering the system there is a limit on how many jobs a single user can submit to any of the queues (see explanations in the sections about hardware limits and resource limits).

3. The cluster is a shared resource:
   - there are limits on how much of the cluster resources (total amount of CPUs, memory, etc) a single user can grab at any time (concurrent use).

4. In most cases your job script also needs to load a module or a set of modules.

5. Do not use the login nodes to run any substantive computation:
   - the login nodes are monitored and processes running on one of the the login node that consume too much resources will have at first their priority reduced, and eventually terminated.

Last Updated 20 Nov 2021 SGK

Conceptual Examples

1. A Trivial Example

Let us assume that

- you have a program that you have successfully compiled/installed on the cluster and that you want to run, we will call it `crunch`;
- the executable `crunch` and all the files needed to run it are in one directory, and/or the files produced will go there.

The simplest way to submit this computation would be to write a two line job script, in a file located in a directory like `/pool/public/sao/hpc/demo` and name it `crunch.job`:

A trivial crunch.job

```
cd /pool/public/sao/hpc/demo
./crunch
```

You start the computation by submitting the job script file with:

```
% cd /pool/public/sao/hpc/demo
% qsub crunch.job
Your job NNNNNNN ("crunch.job") has been submitted
```

The qsub command, if successful, will produce the "*Your job ... has been submitted*" message where *NNNNNNN* is a unique number, the job ID assigned to that job.

By default, the output and error files associated with that job will be located in your home directory and named `crunch.job.oNNNNNNN` and `crunch.job.eNNNNNNN`

2. A Better Example

A better approach is to specify more parameters associated to your job and save more information about the job as follow:

A better crunch.job

```
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
./crunch
echo = `date` job $JOB_NAME done
```

and start the computation with:

```
% cd /pool/public/sao/hpc/demo
% qsub -N crunch -cwd -j y -o crunch.log crunch.job
Your job NNNNNNN ("crunch") has been submitted
```

With this approach:

- you specify the name of the job (`-N crunch`);
- you join error and output in a single file (`-j y`);
- you give a name to the output file (`-o crunch.log`); and
- you tell `qsub` to start the job in the current working directory, and to write the output file there (`-cwd`).

You also, this way, keep track, by saving it in the log file, of

- the job name and job id,
- in which queue the job ran, and on which compute node (host), and,
- when it started and when it ended.

3. Example with Embedded Directives

You can put it all in one file as follows:

A better crunch.job with embedded directives

```
#
#$ -N crunch -cwd -j y -o crunch.log
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
./crunch
echo = `date` job $JOB_NAME done
```

and you submit your job with simply:

```
% cd /pool/public/sao/hpc/demo
% qsub crunch.job
Your job NNNNNNN ("crunch") has been submitted
```

The command `qsub` will look for lines starting with `#$` (that are otherwise comments in the script) and parse these embedded directives as if they were options passed to the command itself.

4. Example with Embedded Directives and Arguments

Finally, the job file is a script: it has to be written according to a specific syntax (C-shell, or Bourne shell) and can take arguments.

Let's now assume that crunch takes two arguments:

- you can either write a different job file for each case you want to run, or
- you can write your job script to use arguments, like this:

A better crunch.job with embedded directives and arguments

```
# /bin/csh
#
# this script takes two arguments and is written using the C-shell syntax
#
#$ -N crunch -cwd -j y -o crunch.log
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
set OPTIONS = (-from $1 -to $2)
echo starting crunch $OPTIONS
./crunch $OPTIONS
echo = `date` job $JOB_NAME done
```

and you can start several jobs as follow:

```
% cd /pool/public/sao/hpc/demo
% qsub -N crunch-20-50 -o crunch-20-50.log crunch.job 20 50
Your job NNNNNNN ("crunch-20-50") has been submitted
% qsub -N crunch-100-150 -o crunch-100-150.log crunch.job 100 150
Your job NNNNNNN ("crunch-100-150") has been submitted
[etc...]
```

In this example the job name and the log file name are redefined to be specific to each job by specifying them on the `qsub` command line.

5. Notes

- Your job script can be an elaborate script, or can start an elaborate and/or convoluted script - how to write Un*x scripts is beyond the scope of this documentation.

Shell Selection

⚠ The GE is setup to use by default the C-shell (`csh`) syntax for job scripts.

- If you prefer using the Bourne shell syntax (`sh` or `bash`), you need to tell `qsub` to use that shell by passing the option `-S /bin/sh`, or adding it as an embedded directive as follows:

A better crunch.job with embedded directives and arguments, using Bourne shell syntax

```
# /bin/sh
#
# this script takes two arguments and is written using the Bourne-shell syntax
#
#$ -S /bin/sh
#$ -N crunch -cwd -j y -o crunch.log
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
OPTIONS="from $1 to $2"
echo starting crunch $OPTIONS
./crunch $OPTIONS
echo = `date` job $JOB_NAME done
```

- Unlike executable script, a `#!/bin/sh` on the first line of a job script is ignored (hence in the the examples I use `# /bin/csh` or `# /bin/sh` as reminders, not as specifiers);
- Unless you are fully versed in the idiosyncrasies of Linux and how `/bin/bash` differs from `/bin/sh` at startup, it is highly recommended to use `/bin/sh` and not `/bin/bash`.
  There are no syntax differences, the only differences are which initialization files are read at startup.

Passing Options to Qsub

- The options passed to the `qsub` command are setup as follow:
  1. as specified in the system wide file `$SGE_ROOT/$SGE_CELL/common/sge_request` (lines not starting w/ #);
  2. as specified in a `.sge_request` file, located in the current working directory (if there is one);
  3. as specified in a `~/sge_request` file, (located in your home directory, if there is one);
  4. as per the embedded options in the submitted job script (lines starting with `#$`);
  5. options passed to `qsub`.

For example, you can write a `~/.sge_request` file with the line

```
-cwd -j y
```

and all your jobs will include `-cwd -j y` as options. At each step you can override a set option.

Email Notifications

You can request that the GE notify you by email when a job

- is started (**b**eginning of the job)
- **e**nds, or
- is **a**borted.

This is accomplished by adding the `-m abe` option to `qsub`.

You can specify the list of users to which the GE will send mail, with the `-M <email_address>` option to `qsub`, by default mail is sent to the job owner, and will be redirected to the email in your `~/.forward` file (see Introduction).

These options can be set as an embedded directive in the job script file.

Environment Variables

When a job is started, the GE defines a slew of environment variables.

💡 The following is a subset of these variables that your job scripts may want to use:

Name	Description	Example of Value
`JOB_NAME`	The job name	`test`

`JOB_ID`	A unique job identifier	`8736123`
`HOSTNAME`	The name of the (master) node the job is running on	`compute-43-11`
`QUEUE`	The name of the cluster queue in which the job is running	`sTHC.q`
`NSLOTS`	The number of queue slots allocated to the job	`1`
`TMPDIR`	The absolute path to the job's temporary working directory	`/tmp/8736126.1.sThC.q`

ℹ You can find a complete list at the bottom of the `qsub` manual - or man page - `man qsub`.

Catching Time Limits

- All the queues, except a few, have time limits: a job will get killed if it exceeds either a CPU limit or an elapsed time (aka real time, wall clock) limit. What those limits are is explained elsewhere (Available Queues and resource limits.)
- All time-limited queues have a soft limit and a hard limit. When the soft limit is reached, a signal is sent by the GE to the script. That signal can be caught to execute something before reaching the hard limit (job termination).
- Jobs using the Bourne-shell syntax (`sh`) can catch the signal, jobs using the C-shell (`csh`) can't (a shortcoming of `Linux`' implementation of `csh`).
- So, especially for `csh` jobs, it is recommended to end the job script with a line like '`echo job done`' that will indicate that the job (script) completed.
- For `sh` jobs, the following example illustrates how to catch signals at the script level:

```
#
#$ -S /bin/sh
#
warn()
{
 echo @ `date` warning, received $1 signal.
}
#
trap "warn xcpu" SIGXCPU
trap "warn usr1" SIGUSR1
trap "warn kill" SIGKILL
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo sleeping $1
sleep $1
echo = `date` job $JOB_NAME done
```

- You can check
  - the status of a job with with the `qstat` command ,
  - the exit status of a job, and the resources it used (CPU, elapsed time, memory, I/Os), once it completed, with the `qacct` command (see the Jobs Monitoring page).

Miscellaneous

- As you queue more than one job beware of "*name space*":
  - jobs will run concurrently (on different compute nodes, or not), so they should not write to the same file(s) and you can keep track of them better if they do not have the same job names.

- Also, especially for `Emacs` users, the last line of a job file must be properly terminated with a newline character (in Unix the return key insert a NL, not a CR)
  The `csh` does not execute a line not properly terminated, and thus the last line of your script may not be executed if it is not followed by a blank line.
- Include the following in your `~/.emacs` file to make sure the last line ends with a NL:

  Add the following lines in your `~/.emacs` file

  ```
  ;; always end a file with a newline character
  (setq require-final-newline t)
  ```

- A job is likely to need resources (CPU time, memory, etc.) and to run in a specific queue. What queues to use and how to request resources is explained in the Available Queues page.
  How to monitor your job(s) and the cluster is explained elsewhere (Job Monitoring and Cluster Monitoring).

- Long jobs should, whenever possible, use *check-pointing*: save intermediate results so one can resume a computation from where it stopped.

Last updated 13 May 2024 SGK

Serial Jobs

A serial job is a job that uses only one CPU.

- It is either started by using a dedicated job script file (i.e. a different one for each job you need to run), or a more sophisticated job script that takes one or several arguments.
- Refer to the Conceptual Examples page to learn how to write jobs files and submit jobs.
- You can look under `~hpc/examples/serial` on Hydra for examples, using different compilers, of a trivial hello world program and job.

Last updated 13 May 2024 SGK

Parallel Jobs

1. Introduction

- A parallel job is a job that uses more than one CPU.
- Since the cluster is a shared resource, it is the GE that allocates CPUs, known as slots in GE jargon, to jobs, hence a parallel job must request a set of slots when it is submitted.
- A parallel job must request a parallel environment and a number of slots using the `-pe <pe-name> N` specification to `qsub`, where
    - `<pe-name>` is either `mpich`, `orte,ompi` or `mthread` (see below);
    - the value of `N` is the number of requested slots and can be specified a `N-M`, where `N` is the minimum and `M` the maximum number of slots requested;
    - that option can be an embedded directive.
    - The job script accesses the number of allocated slots via an environment variable (`NSLOTS`) and,
    - for MPI jobs, gets the list of computed nodes via a so-called machines file.
- There are several types of parallel jobs:

    a. MPI or distributed jobs: the CPUs can be distributed over multiple compute nodes.

PROs	There is conceptually no limit on how many CPUs can be used, the cumulative amount of CPUs and memory a job can use can get quite large. The GE can find (a lot of) unused CPUs on a busy machine by finding them on different nodes
CONs	Each CPU is assumed to be on a separate compute node and thus each process must communicate with the other CPUs to exchange information (aka message passing). Programming can get more complicated and the inter-process communication can become a bottleneck.

    b. Multi-threaded jobs: all the CPUs *must be* on the same compute node.

PROs	All CPUs can share a common memory space, inter-process communication can be very efficient (being local) and programming can be simpler;
CONs	Can only use as many CPUs as there are on the largest compute node, and can get them only if they are not in use by someone else.

    c. Hybrid jobs: the CPUs are distributed, but with the same number of CPUs on each compute node.

PROs	The CPUs on the same node can share a common memory space, while not all CPUs are on the same compute node, hence the total number of CPUs is not limited to the number of CPUs on the largest compute node;
CONs	Coding must mix inter-process communication (like MPI) with shared memory and multi-threading (like OpenMP). This can be tricky, but some problems can greatly benefit from this model.

- How do I know which type of parallel job to submit to?
  The author of the software will in most cases specify if the application can be parallelized and how
    - Some analyses are parallelized by submitting a slew of independent serial jobs,
        - in which case using a job array may be the best approach;
    - some analyses use explicit message passing (`MVAPICH` or `OpenMPI`); while
    - some analyses use a programming model that can use multiple threads (or `OpenMP`); while
    - some mix both: message passing and mutli-threading, hence hybrid jobs.

Last Updated 16 May 2024 SGK

MPI Jobs

1. Introduction: MPI, or Distributed Parallel Jobs with Explicit Message Passing

- An MPI job runs code that uses an explicit message passing programming scheme known as MPI.
- There are two distinct implementations of the MPI protocol:
    1. `OpenMPI (version 3, 4 and 5)`
    2. `MVAPICH (version 2)`
- `Most OpenMPI` implementations use `ORTE`;
- NVIDIA's implementation is slightly different;
- `MVAPICH` uses `MPICH and` supports the InfiniBand as transport fabric (faster message passing)

> ⚠ **Note: OpenMPI is not OpenMP**
>
> - `OpenMPI` is the `ORTE` implementation of MPI;
> - `OpenMP` is an API for multi-platform shared-memory parallel programming.

Modules

- To use MPI, you need to load a module specific to the compiler (GCC, Intel or NVIDIA) and the implementation (vendor's, OpenMPI or MVAPICH)

  ⚠ The MPI modules have been reorganized and renamed when Hydra was upgraded to Rocky 8.9 - aka Hydra-7.

- The following grid of modules, corresponding to a combination of compiler & implementation,
    - it is only a subset of what is available on Hydra:

Module	Note	Module	Note	Module	Note
		intel/24/mpi	VFV	nvidia/24/mpi	VFV
		intel/23/mpi	VFV	nvidia/23/mpi	VFV
		...		...	
gcc/13.2/openmpi	BFS	intel/24/openmpi	BFS	nvidia/24/openmpi	BFS
gcc/13.2/openmpi5	BFS	intel/24/openmpi5	BFS	nvidia/24/openmpi5	BFS
gcc/13.2/openmpi4	BFS	intel/24/openmpi4	BFS	nvidia/24/openmpi4	BFS
gcc/13.2/openmpi4.1.6-13.2.0	BFS	intel/24/openmpi4.1.6-24.0	BFS	nvidia/24/openmpi4.1.6-24.3	BFS
...		...		...	
gcc/12.2/openmpi	BFS	intel/23/openmpi	BFS	nvidia/23/openmpu	BFS
...					
gcc/13.2/mvapich	BFS	intel/24/mvapich	BFS	nvidia/24/mvapich	BFS
gcc/12.2/mvapich	BFS	intel/24/mvapich	BFS	nvidia/23/mvapich	BFS
...		...		...	

Key to notes:

- BFS: built from source;
- VFV: version from vendor;

Versions

- There are more version specific modules available, check with

```
% ( module -t avail ) |& egrep ^gcc/ | grep mpi
```

- for a complete list for gcc.
    - Replace gcc by intel or nvidia to get the respective lists, or consult the list of available modules.
- FYI, the module gcc/13.2/openmpi4.1.6-13.2.0 corresponds to OpenMPI v4.1.6 built from source with GCC v13.2.0
- Use

  ```
  % module whatis <module-name>
  ```

  or

  ```
  % module help <module-name>
  ```

  where `<module-name>` is one of the listed module, to get more specific information.

2. ORTE/OpenMPI Simple Example

The following example shows how to write an ORTE/OpenMPI job script:

Example of a ORTE/OpenMPI job script, using Bourne shell syntax

```
# /bin/sh
#
#$ -S /bin/sh
#$ -cwd -j y -N hello -o hello.log
#$ -pe orte 72
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo + NSLOTS = $NSLOTS distributed over:
cat $PE_HOSTFILE
#
# load gcc's compiler and openmpi
module load gcc/4.9/openmpi
#
# run the program hello
mpirun -np $NSLOTS ./hello
#
echo = `date` job $JOB_NAME done
```

This example will

- show the content of the machine file (i.e. the distribution of compute nodes)
- load the OpenMPI module for gcc version 4.9.2, and
- run the program `hello`,
- requesting 72 slots (CPUs).

It assumes that the program `hello` was built using `gcc v4.9.2`.

3. MPICH/MVAPICH Simple Example

The following example shows how to write a MPICH/MVAPICH job script:

Example of a MVAPICH job script, using C-shell syntax

```
# /bin/csh
#
#$ -cwd -j y - N hello -o hello.log
#$ -pe mpich 72
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo using $NSLOTS slots on:
sort $TMPDIR/machines | uniq -c
#
# load NVIDIA mvapich
module load nvidia/24/mvapich
#
# run the program hello
mpirun -np $NSLOTS -machinefile $TMPDIR/machines ./hello
#
echo = `date` job $JOB_NAME done
```

This example will

- show the content of the machine file (i.e. the distribution of compute nodes),
    - using `sort` & `uniq` to produce a compact list, in a "`hostname no_of_CPUs`" format.
- load the `MVAPICH` module for the NVIDIA compiler, and
- run the program `hello`,
- requesting 72 slots (CPUs).

It assumes that the program `hello` was build using the NVIDIA compiler and the `MVAPICH` library/module to enable the IB as the transport fabric.

4. Additional Notes

- The command `mpirun` is ***aliased*** by the module file to use the full path of the correct version for each case.
- Do not use a full path specification to a version of `mpirun`, using a wrong version of `mpirun` will result in unpredictable results.
  You can check which version corresponds to a module with either
  `% module show <module-name>`
  or, if you use the C-shell,
  `% alias mpirun`
  of, is your use the Bourne shell
  `% declare -f mpirun`
- The error message:

  ```
  [proxy:0:0@compute-N-M.local] HYDU_create_process

  (./utils/launch/launch.c:75): execvp error on file <code> (No such file or directory)
  ```

  means the `mpirun` could not find the executable `<code>`.
- The NVIDIA implementation needs a different machine file, so use the ompi parallel environment (-pe ompi), check the examples (see below).
- One can query the technical implementation details of MPI for each module,

  since each MPI-enabling module implements a slightly different version of MPI.

    - you can query the precise details of each implementation as follows:

Command	to
`% module show <module-file>`	Show how the module changes your Un*x environment. All the modules set the same env variables: `MPILIB` `MPIINC` `MPIBIN` plus either `OPENMPI`, `MPICH`, or `MVAPICH`, and set the alias `mpirun` to use the corresponding full path.
`% module help <module-file>`	Show details on the module, and how to retrieve the details of the specific build.
	Depending on the MPI implementation (`ORTE` or `MPICH`)

`% module load <module-file>`
`% ompi_info [-all]`
	Show precise details of an `ORTE` implementation
	(as shown by `module help <module-file>`)

or	

`% module load <module-file>`	Show precise details of an `MPICH/MVAPICH` implementation
`% mpirun -info`	(as shown by `module help <module-file>`)

5. Where to Find Examples

- Examples on Hydra can be found under `/home/hpc/examples/mpi`
- Look at the README file, it explains where you can find what kind of examples, namely
  - intel/ examples using the vendor supplied MPI, i.e. Intel;
  - nvidia/ examples using the vendor supplied MPI, i.e. NVIDIA;
  - mvapich/gcc/
  - mvapich/intel/
  - mvapich/nividia/ - examples using mvapich2.x build from source, for either of the 3 compilers;
  - openmpi3/gcc/
  - openmpi3/intel/
  - openmpi3/nvidia/ - examples using openmpi3.x build from source, for either of the 3 compilers;
  - openmpi4/gcc/
  - openmpi4/intel/
  - openmpi4/nvidia/ - examples using openmpi4.x build from source, for either of the 3 compilers;
  - openmpi5/gcc/
  - openmpi5/intel/
  - openmpi5/nvidia/ - examples using openmpi5.x build from source, for either of the 3 compilers;
  - share/ source code.

Last Updated 13 May 2024 SGK.

Multi-Threaded Jobs

Multi-threaded, or OpenMP, Parallel Jobs

A multi-threaded job is a job that will make use of more than one CPU but needs all the CPUs to be on the same compute node.

1. Multi-threaded jobs

The following example shows how to write a multi-threaded job script:

Example of a Multi-threaded job script, using Bourne shell syntax

```
# /bin/sh
#
#$ -S /bin/sh
#$ -cwd -j y -N demo -o demo.log
#$ -pe mthread 32
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo + NSLOTS = $NSLOTS
#
# load the demo (fictional) module
module load tools/demo
#
# convert the generic parameter file, gen-params, to a specific file
# where the number of thread are inserted where the string MTHREADS is found
sed "s/MTHREADS/$NSLOTS/" gen-params > all-params
#
# run the demo program specifying all-params as the parameter file
demo -p all-params
#
echo = `date` job $JOB_NAME done
```

This example will run the tool `demo` using 32 CPUs (slots).

The script

- loads the `tools/demo` module,
- parses the file `gen-params` and replaces every occurrence of the string `MTHREAD` by the allocated number of slots (via `$NSLOTS`),
    - using the stream editor `sed` (man sed),
- saves the result to a file called `all-params,`
- runs the tool `demo` and with the parameter file `all-params`.

2. OpenMP jobs

The following example shows how to write an OpenMP job script:

Example of a OpenMP job script, using C-shell syntax

```
# /bin/csh
#
#$ -cwd -j y -N hellomp -o hellomp.log
#$ -pe mthread 32
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo + NSLOTS = $NSLOTS
#
# load the nvida module
module load nvidia
#
# set the variable OMP_NUM_THREADS to the content of NSLOTS
# this tell OpenMP applications how many threads/slots/CPUs to use
setenv OMP_NUM_THREADS $NSLOTS
#
# run the hellomp OpenMP program, build w/ NVIDIA
./hellomp
#
echo = `date` job $JOB_NAME done
```

This example will run the program `hellomp`, that was compiled with the NVIDIA compiler, using 32 threads (CPUs/slots). The script

- loads the nvidia module,
- sets `OMP_NUM_THREADS` to the content of `NSLOTS` to specify the number of threads
- runs the program `hellomp`.

Examples

- You can find examples of OpenMP for all 3 compilers on Hydra under /home/hpc/examples/openmpm check the README file:
    - gcc/   - GNU compilers;
    - intel/ - Intel compilers;
    - nvidia/ - NVIDIA compilers;
    - share/ - source code.

Last Updated 13 May 2024 SGK.

Hybrid Jobs

Introduction

- A hybrid job is a job that will use `N` slots/CPUs/threads distributed as `M` CPUs/threads on `K` compute nodes, where `N = K x M`.
- The software must be written as to make use of this configuration, usually by combining message passing (MPI) and multi-threading (OpenMP).

1. How to Run a Hybrid Job

To run a job as a hybrid job you need to:

1. request one of the hybrid PEs (parallel environments)
2. source a configuration file that is created at run-time (`$TMPDIR/set-hybrid-config`)
3. run a code written for a hybrid PE.

2. Examples

Examples of hybrid jobs are under `/home/hpc/examples/hybrid`.

⚠ The key line(s) in the job file examples are ⚠:

csh syntax	sh syntax

`if (-e $TMPDIR/set-hybrid-config) then`
`  source $TMPDIR/set-hybrid-config`
`endif`
	`if [ -e $TMPDIR/set-hybrid-config ]`
	`then`
	`  source $TMPDIR/set-hybrid-config`
	`fi`

- The if statement (optional) allows the job file to be run in non-hybrid mode (if applicable).
- You will see in the output/log file the following:

-pe h8 32

```
--------------- hybrid_start 1.0/1 ---------------
hybrid_start: remember to 'source $TMPDIR/set-hybrid-config' to properly setup your env
--------------------------------------------------
....
PE=h8 NSLOTS=4 OMP_NUM_THREADS=8
```

- The last line shown above will be produced only of you source the configuration file.
- Use `$NSLOTS` and `$OMP_NUM_TREADS` whenever needed.

3. Available PEs

You can use one of the following hybrid PEs:

Name	No of CPUs/node	Example	Means
`h2`	2	`-pe  h2 64`	request 64 slots distributed as 2 CPUs/node on 32 different nodes
`h4`	4	`-pe  h4 64`	request 64 slots as 4 CPUs/node on 16 different nodes
`h8`	8	`-pe  h8 64`	request 64 slots as 8 CPUs/node on 8 different nodes
`h12`	12	`-pe h12 48`	request 48 slots as 12 CPUs/node on 4 different nodes
`h16`	16	`-pe h16 64`	request 64 slots as 16 CPUs/node on 4 different nodes

etc... up to h64 as follows: `h2, h4, h8, h12, h16, h24, h32, h48` and `h64`

Note

⚠ You specify `M` and `N`, where `N = K x M`:

- `-pe h8 34` stands for `M=8 N=32  K=4`
- It will run on `K` different nodes, each node can use `M` threads/CPUs,
- The job will be rejected if `N` is not a multiple of `M`,
- The hybrid PEs are available in all the hi-CPUs queues (`?ThC.q`) .

More Details & Explanations

Simple job, using C-shell syntax, to run an `OpenMPI/OpenMP` hybrid code, called `hybrid`.

hybrid.job

```
# /bin/csh
#
#$ -q mThC.q -pe h8 64
#$ -N hybrid -o hybrid.log -cwd -j y
#
echo $JOB_NAME started `date` on $HOSTNAME in $QUEUE jobID=$JOB_ID
#
if (-e $TMPDIR/set-hybrid-config) then
  source $TMPDIR/set-hybrid-config
endif
#
module load gcc/4.9/openmpi
mpirun -np $NSLOTS -hostfile $HOSTFILE ./hybrid
#
echo `date` $JOB_NAME done.
```

This job will produce as output:

hybrid.log

```
--------------- hybrid_start 1.0/1 ---------------
hybrid_start: remember to 'source $TMPDIR/set-hybrid-config' to properly setup your env
--------------------------------------------------
Warning: no access to tty (Bad file descriptor).
Thus no job control in this shell.
hybrid started Thu Aug 18 13:32:05 EDT 2016 on compute-1-4.local in mThC.q jobID=6374206
PE=h8 NSLOTS=8 OMP_NUM_THREADS=8
hello from iRank=  1, iSize=  8, hostname=compute-1-4.local
hello from iRank=  1, iSize=  8, hostname=compute-1-4.local
hello from iRank=  1, iSize=  8, hostname=compute-1-4.local
hello from iRank=  1, iSize=  8, hostname=compute-1-4.local
hello from iRank=  1, iSize=  8, hostname=compute-1-4.local
hello from iRank=  1, iSize=  8, hostname=compute-1-4.local
hello from iRank=  1, iSize=  8, hostname=compute-1-4.local
hello from iRank=  1, iSize=  8, hostname=compute-1-4.local
hello from iRank=  5, iSize=  8, hostname=compute-3-11.local
hello from iRank=  5, iSize=  8, hostname=compute-3-11.local
hello from iRank=  5, iSize=  8, hostname=compute-3-11.local
hello from iRank=  5, iSize=  8, hostname=compute-3-11.local
hello from iRank=  5, iSize=  8, hostname=compute-3-11.local
hello from iRank=  5, iSize=  8, hostname=compute-3-11.local
hello from iRank=  5, iSize=  8, hostname=compute-3-11.local
hello from iRank=  5, iSize=  8, hostname=compute-3-11.local
hello from iRank=  2, iSize=  8, hostname=compute-1-5.local
hello from iRank=  2, iSize=  8, hostname=compute-1-5.local
hello from iRank=  2, iSize=  8, hostname=compute-1-5.local
hello from iRank=  2, iSize=  8, hostname=compute-1-5.local
hello from iRank=  2, iSize=  8, hostname=compute-1-5.local
hello from iRank=  2, iSize=  8, hostname=compute-1-5.local
hello from iRank=  2, iSize=  8, hostname=compute-1-5.local
hello from iRank=  2, iSize=  8, hostname=compute-1-5.local
hello from iRank=  3, iSize=  8, hostname=compute-1-6.local
hello from iRank=  3, iSize=  8, hostname=compute-1-6.local
hello from iRank=  3, iSize=  8, hostname=compute-1-6.local
hello from iRank=  3, iSize=  8, hostname=compute-1-6.local
hello from iRank=  3, iSize=  8, hostname=compute-1-6.local
hello from iRank=  3, iSize=  8, hostname=compute-1-6.local
hello from iRank=  3, iSize=  8, hostname=compute-1-6.local
hello from iRank=  3, iSize=  8, hostname=compute-1-6.local
hello from iRank=  4, iSize=  8, hostname=compute-2-15.local
hello from iRank=  4, iSize=  8, hostname=compute-2-15.local
hello from iRank=  4, iSize=  8, hostname=compute-2-15.local
hello from iRank=  4, iSize=  8, hostname=compute-2-15.local
hello from iRank=  4, iSize=  8, hostname=compute-2-15.local
hello from iRank=  4, iSize=  8, hostname=compute-2-15.local
hello from iRank=  4, iSize=  8, hostname=compute-2-15.local
hello from iRank=  4, iSize=  8, hostname=compute-2-15.local
hello from iRank=  6, iSize=  8, hostname=compute-3-3.local
hello from iRank=  6, iSize=  8, hostname=compute-3-3.local
hello from iRank=  6, iSize=  8, hostname=compute-3-3.local
hello from iRank=  6, iSize=  8, hostname=compute-3-3.local
hello from iRank=  6, iSize=  8, hostname=compute-3-3.local
hello from iRank=  6, iSize=  8, hostname=compute-3-3.local
hello from iRank=  6, iSize=  8, hostname=compute-3-3.local
hello from iRank=  6, iSize=  8, hostname=compute-3-3.local
hello from iRank=  7, iSize=  8, hostname=compute-3-4.local
hello from iRank=  7, iSize=  8, hostname=compute-3-4.local
hello from iRank=  7, iSize=  8, hostname=compute-3-4.local
hello from iRank=  7, iSize=  8, hostname=compute-3-4.local
hello from iRank=  7, iSize=  8, hostname=compute-3-4.local
hello from iRank=  7, iSize=  8, hostname=compute-3-4.local
hello from iRank=  7, iSize=  8, hostname=compute-3-4.local
hello from iRank=  7, iSize=  8, hostname=compute-3-4.local
hello from iRank=  0, iSize=  8, hostname=compute-1-3.local
hello from iRank=  0, iSize=  8, hostname=compute-1-3.local
hello from iRank=  0, iSize=  8, hostname=compute-1-3.local
hello from iRank=  0, iSize=  8, hostname=compute-1-3.local
hello from iRank=  0, iSize=  8, hostname=compute-1-3.local
hello from iRank=  0, iSize=  8, hostname=compute-1-3.local
hello from iRank=  0, iSize=  8, hostname=compute-1-3.local
hello from iRank=  0, iSize=  8, hostname=compute-1-3.local
Thu Aug 18 13:32:10 EDT 2016 hybrid done.
```

✓ The program `hybrid` corresponds to the following trivial F90 code:

hybrid.f90

```
program hello
  !
  include 'mpif.h'
  !
  integer iErr, iRank, iSize
  integer mpiComm, msgTag
  !
  character*40 hostname
  call HOSTNM(hostname)
  !
  mpiComm = MPI_COMM_WORLD
  msgTag  = 0
  !
  call MPI_INIT(iErr)
  call MPI_COMM_RANK(mpiComm, iRank, iErr)
  call MPI_COMM_SIZE(mpiComm, iSize, iErr)
  !
  !$omp parallel
  print 9000, 'hello from iRank=',iRank, &
       ', iSize=', iSize, &
       ', hostname=', trim(hostname)
  !$omp end parallel
  !
  call MPI_FINALIZE(iErr)
9000 format(a,i3,a,i3,a,a)
!
end program hello
```

The configuration file (`$TMPDIR/set-hybrid-config`) does the following:

- resets the values of `NSLOTS`, and sets `OMP_NUM_THREADS`, and show their values, and
- rewrites the machine file (`$MACHINEFILE`, for `MPI`), and the host file (`$HOSTFILE, for OpenMPI`),
- shows the resulting values of `PE, NSLOTS` and `OMP_NUM_THREADS`.

You can find more examples in `~hpc/examples/hybrid`, where this example is build and run for different compilers (`gnu, Intel, NVIDIA`), using either `MPI` or `OpenMPI` and using the `sh` or `csh` syntax.

Last Updated 13 May 2024 SGK.

Job Arrays

1. Introduction

A jobs array is specified by adding a task range to `qsub` via the `-t` flag:

```
% qsub -t 1-100 model.job
```

```
Your job-array NNNNNN.1-100:1 ("model.job") has been submitted
```

The scheduler will start 100 jobs, each starting the job script file `model.job`, and pass to each job a task identifier (a number between 1 and 100) via an environment variable.

The syntax for the `-t` flag is `-t n[-m[:s]]`, namely:

`-t 1-20`	run 20 tasks, with task IDs ranging from1 to 20
`-t 10-30`	run 21 tasks, with task IDs ranging from 10 to 30
`-t 50-140:10`	run 10 tasks, with task IDs ranging from 50 to 140 by step of 10 (50, 60, ..., 140)
`-t 20`	run *one* task, with task ID 20

Each instantiation of the job will have access to the following four environment variables:

`SGE_TASK_ID`	unique ID for the specific task
`SGE_TASK_FIRST`	ID of the first task, as specified with `qsub`
`SGE_TASK_LAST`	ID of the last task, as specified with `qsub`
`SGE_TASK_STEPSIZE`	task ID step size, as specified with `qsub`

You can also limit the number of concurrent tasks with the `-tc` flag, for example:

```
% qsub -t 1-1000 -tc 100 model.job
```

will request to run 1,000 jobs, but no more than 100 running at the same time.

2. Example of a Job Script

The follow example shows how to submit a job array, using embedded directives:

Example of job script to submit a job array, using C-shell syntax

```
# /bin/csh
#
#$ -N model-1k -cwd -j y -o model.$TASK_ID.log
#$ -t 1-1000 -tc 100
#
echo + `date` $JOB_NAME started on $HOSTNAME in $QUEUE with jobID=$JOB_ID and taskID=$SGE_TASK_ID
#
set ID = model.$SGE_TASK_ID
./model -id $ID
#
echo = `date` $JOB_NAME for taskID=$SGE_TASK_ID done.
```

This example

- requests to run 1,000 models, using a task ID ranging from 1 to 1000, but limited to 100 running at the same time;
- assumes that the model computation is run with the command `./model -id <ID>`, where `<ID>` is the string `model.N` and `N` a number between 1 and 1000;
- also shows how to use the pseudo variable `TASK_ID` (not `SGE_TASK_ID`, yes, I agree, it is confusing) to give the log file of each task a different name: the output of task 123 will be `model.123.log` in the current working directory.

3. How to Convert a Task ID to a More Useful Set of Parameters

In most cases, when starting a computation you will need to convert a simple task identifier to a slew of parameters.

You can put that conversion into your code, but you may not want to do it, or can't do it because you are using some tool or package you can't modify.

Here are a few suggestions (Un*x tricks, using C-shell syntax) on how to do it:

1. You can use a separate input file for each job (task):
   If your code reads from `stdin` (standard input) you can do something like this:

   ```
   @ i = $SGE_TASK_ID
   ./domodel < input.$i
   ```

   You just need to prepare as many `input.NNN` files as cases you want to run, from `input.1` to `input.500` for example.

   If you prefer to call them `input.001` to `input.500`, you can use `awk` to reformat `$i` as follows:

   ```
   @ i = $SGE_TASK_ID
   set I = `echo $i | awk '{printf "%3.3d", $1}'`
   ./domodel < input.$I
   ```

   where `"%3.3d"` is the trick (a C-like format specifier) to convert the integer `$i` into a 3 character string `$I` with leading zeros if needed.

   The Bourne shell (`sh` or `bash`) equivalent is:

   ```
   i=$SGE_TASK_ID
   I=`echo $i | awk '{printf "%3.3d", $1}'`
   ./domodel < input.$I
   ```

2. You can use a single text file that lists a slew of parameters and extract one line, using the command `awk` (`man awk`):

   ```
   @ i = $SGE_TASK_ID
   set P = (`awk "NR==$i" parameters-list.txt`)
   ./compute $P
   ```

   This example will extract one line from the file `parameters-list.txt,` namely the line whose line number is stored in the variable `$i` (the 1st line, the 2nd line, etc). [NR stands for record number]

   It sets the variable `$P` to the content of that one line.

   You just have to create such a file with as many lines as cases of `compute` you wish to run. Each line holds the parameters that will be passed to `compute`.

   The Bourne shell (`sh` or `bash`) equivalent is:

   ```
   i=$SGE_TASK_ID
   P=`awk "NR==$i" parameters-list.txt`
   ./compute $P
   ```

3. You can write a tool (a small program or script, that I call here `mytool`) that does the conversion. You just run it to get the parameters:

   ```
   @ i = $SGE_TASK_ID
   set P = (`./mytool $i`)
   ```

or for Bourne shell (`sh` or `bash`) aficionados:

```
i=$SGE_TASK_ID
P=`./mytool $i`
```

4. You can use the shell syntax to use or manipulate the variable $SGE_TASK_ID to execute the right (set of) command(s) from the given task integer

4. How to Consolidate Small Jobs into Fewer Larger Jobs

- There is some overhead each time the GE starts a job, or a task.
- So if you need to compute let's say 5,000 similar tasks, each taking only 3 minutes,
    - it may be convenient to submit a 5,000 task job array, but
    - it will be inefficient: the system will spend 25 to 50% of its time starting and keeping track of a slew of small jobs.
- The following script illustrates a simple trick to consolidate such computations when using a job array:

Example of job array consolidation wrapper script, domodel.job, using C-shell syntax

```
# /bin/csh
#
# simple wrapper to consolidate using the step size
#
#$ -N model-1k20 -cwd -j y -o model-$TASK_ID-by-20.log
#$ -t 1-1000:20
#
echo + `date` $JOB_NAME started on $HOSTNAME in $QUEUE with jobID=$JOB_ID
#
@ iFr = $SGE_TASK_ID
@ iTo = $iFr + $SGE_TASK_STEPSIZE - 1
if ($iTo > $SGE_TASK_LAST) @ iTo = $SGE_TASK_LAST
#
echo running model.csh for taskIDs $iFr to $iTo
@ i = $iFr
while ($i <= $iTo)
  ./model.csh $i >& model-$i.log
  @ i++
end
#
echo = `date` $JOB_NAME for taskIDs $iFr to $iTo done.
```

This wrapper, that I call `domodel.job`, will run 20 models in a row, using the step size of the job array, via the `csh` script `model.csh`.

So instead of running 1,000 three-minute-long jobs, it will run 50 one-hour-long jobs.

You can, of course, adjust the step size accordingly, including setting it to 1 for cases when individual models lead to long computations.

The script `model.csh` is simply:

C-shell script model.csh called by domodel.job

```
#!/bin/csh
#
set TID = $1
echo + `date` model.csh started for taskID=$TID
#
./model -id $TID
#
echo = `date` model.csh for taskID=$TID done.
```

but it can be as complex as you may need/want it to be. (BTW, there is nothing C-shell or Bourne shell syntax specific in this example)

The file `model.csh` must be authorized to be executed with the command:

```
% chmod +x model.csh
```

You can use a bash script, or any other valid Linux command in place of the line `./model.csh`.

5. Rules for Submitting Job Arrays that use Parallel Environments (like MPI)

⚠️ While one can submit a job array that uses a parallel environment (`-pe` and `-t,` or parallel job arrays), one ***must*** use a the following approach to avoid a race condition specific to SGE.

- By default, SGE makes a local copy of each job script on the compute nodes it runs on.
- Parallel job arrays should avoid this to prevent a race condition, where for a small fraction of the tasks the scheduler starts the script *before* it is copied, hence some tasks fails to start.

- The output of `qstat -j 9616234` will show something like this:
  `error reason  11:         03/24/2016 11:09:11 [10464:63260]: unable to find job file "/opt/gridengine/default/spool/compute-2-2/job_scripts/9416234"`
  and the SGE reporting file will list:
  `job never ran -> schedule it again`
- The output of `qstat -f -explain E | grep QERROR` will show something like this:
  `queue mThC.q marked QERROR as result of job 9616234's failure at host compute-1-2.local`
  leaving a queue entry in Error state.

How to Write a Parallel Job Array

1. Do not use embedded directive (😞).
2. Write a script (`sh`, `csh`, `perl`, `python`, etc) with the required steps, as for a job script.
3. Make that script executable (`chmod +x`), you can use the `#!` mechanism to specify the interpreter (aka shebang).
4. Write a file with the `qsub` command and all the options that you would otherwise put as embedded directives.
5. Pass the `-b y` option to `qsub` and specify the full path of the script to execute.
6. Source that file to submit the parallel job array.
7. ⚠️ Do not modify the executable script file while the job array is running.

Example

The following job script with embedded directives must be broken into two files:

<table>
<tr><th>one job script with embedded directives</th><th>is replaced by two files, a qsub_XXX.sou and a XXX.sh</th></tr>
<tr><td>

demo.job

```
#--------
#$ -q mThC.q
#$ -pe orte 20
#$ -l mres=4G,h_data=4G,h_vmem=4G
#$ -cwd -y j -N demo -o demo.$TASK_ID.log
#$ -t 1-100
#-------
module load some/thing
#-------
echo + `date` job $JOB_NAME started in $QUEUE
with jobID=$JOB_ID on $HOSTNAME
echo + taskID=$SGE_TASK_ID
echo + NSLOTS=$NSLOTS distributed over:
cat $PE_HOSTFILE
#
mpirun -np $NSLOTS crunch -i input.$SGE_TASK_ID -o output.$SGE_TASK_ID
#
echo = `date` job $JOB_NAME done
```

</td><td>

qsub_demo.sou

```
qsub \
 -q mThC.q \
 -pe orte 20 \
 -l mres=4G,h_data=4G,h_vmem=4G \
 -cwd -y j -N demo -o 'demo.$TASK_ID.log' \
 -t 1-100 \
 -b y $PWD/demo.sh
```

⚠️ no spaces after the '\'

</td></tr>
</table>

demo.sh

```
#!/bin/sh
# any embedded directives here will be ignored
#-------
source /etc/profile.d/modules.sh
module load some/thing
#-------
echo + `date` job $JOB_NAME started in $QUEUE with
jobID=$JOB_ID on $HOSTNAME
echo + taskID=$SGE_TASK_ID
echo + NSLOTS=$NSLOTS distributed over:
cat $PE_HOSTFILE
#
mpirun -np $NSLOTS crunch -i input.$SGE_TASK_ID -o
output.$SGE_TASK_ID
#
echo = `date` job $JOB_NAME done
```

💡 This can be any type of executable script, but if you use:

- `#!/bin/sh` or `#!/bin/bash,` you'll need `source /etc/profile.d/modules.sh`" to access modules,
- `#!/bin/sh -l` or `#!/bin/bash -l`, module will be defined, but the script will read `~/.profile`,
- `#!/bin/csh` or `#!/bin/tcsh`, module will be defined,
- `#!/bin/csh -f` or `#!/bin/tcsh -f`, you'll need `source /etc/profile.d/modules.csh` to access modules.
  😉 Don't you love Linux? (It's all in the man pages, tho).

Before submitting the job array, make sure the script is executable:

```
chmod +x demo.sh
```

To submit the job array, simply source the `qsub_XXX.sou` file:

```
source qsub_demo.sou
```

You can edit the `qsub_demo.sou` to submit more tasks, but do not modify the executable script file while the job array is running.

Last updated 26 Jun 2018 SGK.

Available Queues

1. Introduction

Every job running on the cluster is started in a queue.

- The GE will select a queue based on the resources requested and the current usage in each queue.
- If you don't specify the right queue or the right resource(s), your job will either
    - not get queued,
    - wait forever and never run, or
    - start and get killed when it exceeds one the limits of the queue it ran in.

All jobs run in batch mode, unless you use the interactive queue, and the default GE queue (`all.q`) is not available.

2. Matrix of Queues

The set of available queues is a matrix of queues:

- Five sets of queues:
    1. a high-CPU set,
    2. a high-memory set, complemented by
        - a very-high-memory restricted queue,
    3. a GPU set,
    4. an interactive queue, and
    5. a special I/O queue.

- The high-cpu, high-memory and GPU sets of queues have different time limits: short, medium, long and unlimited.
    - the high-cpu queues are for serial or parallel jobs that do not need a lot of memory (less than 8GB per CPU),
    - the high-memory queues are for serial or multi-threaded parallel jobs that require a lot of memory (more then 6GB, but limited to 450GB),
    - the very-high-memory queue is reserved for jobs that need a very large amount of memory (over 450GB),
- interactive queues, to run interactively on a compute node (w/out or w/ a GPU), although it also has limits, and
- a special queue reserved for projects that need special resources (I/Os).

The list of queues and their characteristics (time versus memory limits) are:

Memory limit per CPU	**Time limit (soft CPU/elapsed time)**				**Available parallel environments**	**Type of jobs**
	short	**medium**	**long**	**unlimited**		
resident / virtual	**T<7h /14h**	**T<6d /12d**	**T<30d /60d**			
8GB / 64GB	**sThC.q**	**mThC.q**	**lThC.q**	**uThC.q**	`mpich`, `orte`, `mthread`, `hybrid`	serial or parallel, that needs less than 8GB of memory per CPU
450GB / 900GB	**sThM.q**	**mThM.q**	**lThM.q**	**uThM.q**	`mthread`	serial or multi-threaded, 8GB < memory needed < 450GB
2TB / 2TB				**uTxlM.rq**	`mthread`	serial or multi-threaded, memory needed > 450GB, restricted
64GB / 128G	**`sTgpu.q`**	**`mTgpu.q`**	**`lTgpu.q`**		`mthread`	queue to access GPUs
		T<12h /24h	**T<12h /72h**			
8GB / 64GB		**`qrsh.iq`**			`mthread`	interactive queue, use `qrsh` or `qlogin`; 12h of CPU 24h of wallclock.

64GB / 65GB		**qgpu.iq**			mthread	interactive queue to access GPUs, restricted; 12h of CPU 24h of wallclock.
8GB / 64 GB			**lTIO.sq**		mthread	I/O queue, to access `/store`; 12h of CPU, 72h of wallclock.

Notes

- the listed time limit is the soft CPU limit (`s_cpu`), and elapsed time (s_rt)
    - the soft elapsed time limit (aka real time, `s_rt`) is twice the soft CPU limit for all the queues.
    - the hard time limits are 15m longer than the soft ones (i.e., `h_cpu`, and `h_rt`).
      Namely, in the short-time limit queues
        - a serial job is warned if it has exceeded 7 hours of consumed CPU, and killed after it has consumed 7 hours and 15 minutes of CPU, or
        - warned if it has spent 14 hours in the queue and killed after spending 14 hours and 15 minutes in the queue.
        - For parallel jobs the consumed CPU time is scaled by the number of allocated slots,
        - the elapsed time is not.
- memory limits are per CPU,
    - so a parallel job, in a high-cpu queue can use up to `NSLOTS` x 6 GB, where `NSLOTS` is the number of allocated slots (CPUs)
    - parallel jobs in the other queues are limited to multi-threaded jobs (no multi-node jobs)
    - the limit on the virtual memory (vmem) is set to be higher that the resident memory (rss)
- memory usage is also limited by the available memory on a given node.
- If you believe that you need access to a restricted or test queue, contact us.

3. How to Specify a Queue

- By default jobs are most likely to run in the short high-cpu queue (`sThC.q`).
- To select a different queue you need to either
    - specify the name of the queue (via `-q <name>`), or
    - pass a requested time limit (via `-l s_cpu=<value>` or `-l s_rt=<value>`).
- To use a high-memory queue, you need to
    - specify the memory requirement (with `-l mres=X,h_data=X,h_vmem=X`)
    - confirm that you need a high-memory queue (`-l himem`)
    - select the time limit either by
        - specifying the queue name, or (via `-q <name>`), or
        - pass a requested time limit (via `-l s_cpu=<value>` or `-l s_rt=<value>`).
- To use a GPU queue, you need to specify -l gpu
- To use the unlimited queues, i.e., uThC.q or uThM.q, you need to confirm that you request a low priority queue (`-l lopri`)

💡 Why do I need to add `-l himem`,`-l gpu` or `-l lopri`?

- This prevents the GridEngine from submitting a job to these queues that did not request the associated resources, just because one of these queues are less used.
  It prevent the scheduler from "*wasting*" valuable resources.

Examples

qsub flags	Meaning of the request
`-l s_cpu=48:00:00`	48 hour of consumed CPU (per slot)
`-l s_rt=200:00:00`	200 hour of elapsed time
`-q mThC.q`	use the `mThC.q` queue
`-l mres=120G,h_data=12G,h_vmem=12G -pe mthread 10`	12GB of memory (per CPU), for a 10 CPUs parallel job will reserve 120GB
`-q mThM.q -l mres=12G,h_data=12G,h_vmem=12G,himem`	to run in the medium-time high-memory queue. This is a correct, i.e., complete, specification (memory use specs and `himem`)
`-q uThC.q -l lopri`	to run in the unlimited high-cpu queue, note the `-l lopri`
`-q uTxlM.rq -l himem`	unlimited-time, extra-large-memory, restricted to a subset of users

All jobs that use more than 2 GB of memory (per CPU) should include a memory reservation and requirement with `-l mres=X,h_data=X,h_vmem=X`.

- If you do not, your job(s) may not be able to grab the memory they need at run-time and crash, or crash the node.
- memory reservation is for serial and mthread jobs, not MPI.
    - MPI jobs can specify `h_data=X` and `h_vmem=X` resources.
- X is a number followed by a unit, like 100M or 10G, if you specify h_vmem=5 (no unit) your job can only use 5 bytes and will die right away.

4. Interactive Queue

You can start an interactive session on a compute node using the command `qrsh` or `qlogin` (not `qsub`, nor `qsh`)

- Some compute nodes are set aside for interactive use,
    - the corresponding queue is named `qrsh.iq`
- To start an interactive session, use `qrsh` or `qlogin`
    - `qrsh` will start an interactive session on one of the interactive nodes,
        - it takes both options and arguments (like `qsub)`
    - `qlogin` is similar to `qrsh`, although
        - it will propagate the `$DISPLAY` variable, so you can use X-enabled applications (like to plot to the screen) if you've enabled X-forwarding, and
        - it does not take any argument, but will take options.
- Unless you need X-forwarding, use `qrsh`

Limits on Interactive Queues

- Like any other queue, the interactive queue has its own limits:

CPU	12h per slot (CPU/core)
Elapsed Time	24h per session
Memory	8GB/64GB per slot (CPU/core)

- Like for `qsub`, you can request more than one slot (CPU/thread) with the `-pe mthread N` option,
    - where `N` is a number between 2 and 16, as in:

```
qrsh -pe mthread 4
```

    - requesting more slots allows you also to use more memory (4 slots means up to 4 x 8G = 32G).
- Each user is limited to one concurrent interactive session, and up to 16 slots (CPUs/cores).
- The overall limits of 512 slots/user include all the queues (so if you have 512 slots use in batch mode, you won't be able to get an interactive session).

The NSLOTS Variable

- As of Feb 3, 2020, `$NSLOTS` is properly propagated by `qrsh`, but not by `qlogin.`
    - There is no mechanism to propagate `$NSLOTS` with `qlogin` and enable X-forwarding.

⚠ Remember, Hydra is a shared resource, do not "*waste*" interactive slots by keeping your `qlogin` or `qrsh` session idle.

5. Memory Reservation

- We have implemented a memory reservation mechanism, (via `-l mres=XT`)
    - This allows the job scheduler to guarantee that the requested amount of memory is available for your job on the compute node(s), by keeping track of the reserved memory and not scheduling jobs that reserve more memory than available.
    - Hence reserving more than you will use prevents others (including your own other jobs) from accessing the available memory and indirectly the available CPUs (like if you use one, or even just a few, CPUs but grab most of the memory of a given compute node).

- We have at least 2GB/CPUs, but more often 4GB/CPUs on the compute node, still
  ⚠ it is recommended to reserve memory if your job will use more than 2GB/CPU, and
  set `h_data=X` and `h_vmem=X` to the value used in `mres=XT`.

- ⚠ Remember:
    - The memory specification is
        - per JOB in `mres=XT` (total)
        - per CPU in `h_data=X` and `h_vmem=X`, it should be `XT` divided by the number of requested slots.
    - Memory is a scarce and expensive resource, compared to CPU, as we have fewer nodes with a lot of memory.
    - Try to *guesstimate* the best you can your memory usage, and
    - monitor the memory use of your job(s) (see Job Monitoring).

- ⚠ Note:
    - Do not hesitate to re-queue a job if it uses a lot more or a lot less than your initial guess, esp. if you plan to queue a slew of jobs;
    - often, memory usage scales with the problem in predictable ways, or the documentation might indicate memory requirements, consider running some test cases to help you *guesstimate*, and
      trim down your memory reservation whenever possible: a job that requests oodles of memory may wait for a long time in the queue for that resource to free up.
    - Consider breaking down a long task into separate jobs if different steps need different type of resources.

Resources Format Specifications

- The format for
  - memory specification is a positive (decimal) number followed by a unit (aka a multiplier), like `13.4G` for 13.4 GB;
  - CPU or RT time specification is `h:m:s`, like `100:00:00` for 100 hours (or "`100::`", while "`100`" means 100 seconds).

  see `man sge_types`.

6. Host Groups

The GridEngine supports the concept of a host group, i.e., a list of hosts (compute nodes).

- You can request that a job run only on computers in a given a host group, and we use these host groups to restrict queues to specific list of hosts.
- To request computers of a given host group, use something like this "`-q mThC.q@@ib-hosts`" (*yes* there is a double "`@`"),
- In fact the queue specification to `qsub` can be a RE, so `-q '?ThC.q@@ib-hosts'`, means any high-cpu queue but only hosts on IB.

- You can get the list of all the host groups with (show host group list)
```
% qconf -shgrpl
```
- and get the list of hosts for a specific host group with
```
% qconf -shgrp <host-group-name>
```

- Here is the list of host groups:

Name	Description
`@all-hosts`	all the hosts
`@hicpu-hosts`	high CPU hosts
`@himem-hosts`	high memory hosts (521GB/host)
`@xlmem-hosts`	extra large memory hosts (>=1TB/host)
`@io-hosts`	hosts in the IO queue
`@gpu-hosts`	hosts with GPUs
`@ssd-hosts`	hosts with local SSD
`@24c-hosts`	hosts with 24 CPUs
`@NNc-hosts`	hosts with NN CPUs (NN=value up to 192)
`@avx-hosts`	hosts with AVX-capable CPUs
`@avx2-hosts`	hosts with AVX2-capable CPUs

7. CPU Architecture

- The cluster is composed of compute nodes with different CPU architectures.
- You can tell the scheduler to run your job on specific CPU architecture(s) using the `cpu_arch` resource.

Composition

- The cluster's CPU architecture composition is as follows:

Compute Nodes	CPU Arch.	CPU Model Name
`compute-64-xx`	`skylake`	Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz
`compute-65-xx`	zen	AMD EPYC 7713P 64-Core @1.94GHz
`compute-75-xx`	zen	AMD EPYC 7H12 64-Core @ 2.53GHz
`compute-76-xx`	zen	AMD EPYC 9654 64-Core
	zen	AMD EPYC 9534 64-Core
`compute-79-xx`	`skylake`	Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz

`compute-84-xx`	`skylake`	Intel(R) Xeon(R) Platinum 8280 CPU @ 2.70GHz
`compute-93-xx`	`haswell`	Intel(R) Xeon(R) CPU E7-8867 v3 @ 2.50GHz
	`broadwell`	Intel(R) Xeon(R) CPU E7-8860 v4 @ 2.20GHz

Usage

Grid Engine/Qsub

- You can use the `cpu_arch` resource to request a specific architecture or a set of architectures or avoid specific architecture(a), like in

  `qsub -l cpu_arch=haswell`

  such job will only run on nodes with a `haswell` CPU architecture.

- Alternatively, you can use logic constructs as follows:

  `qsub -l cpu_arch=\!broadwell` - any CPU, except `broadwell`.

  `qsub -l cpu_arch='haswell|skylake'` - either `haswell` OR `skylake` CPUs.

Tools

- You can retrieve the node's CPU architecture with the command `get-cpu_arch`, accessible after loading the module `tools/local`.
- You can query the `cpu_arch` list within a queue with, for example,

  `qstat -F cpu_arch -q sThC.q`

- You can set the environment variable `cpu_arch` by loading the module `tools/cpu_arch`.

Compilers

- Each compiler allows you to specify specific target processors (aka CPU architectures).
- The syntax is different for each compiler, read carefully the compiler's user guide.

Examples

- Run only on two types of architectures:

Restrict to some type of arch

```
#
#$ -cwd -j y -N demo1 -o demo1.log
#$ -l cpu_arch='haswell|skylake'
#
./crunch
#
```

- Run a different executable depending on the node's CPU architecture, and tell the scheduler to avoid AMD processors:

Run different code for different arch

```
#
#$ -cwd -j y -N demo2 -o demo2.log
#$ -l cpu_arch='!broadwell'
#
module load tools/cpu_arch
bin/$cpu_arch/crunch
#
```

8. Queue Selection Validation and/or Verification

- You can submit a job script and verify if the GE can run it, i.e.,
  💡 can the GE find the adequate queue and allocate the requested resources?

- The qsub flag "`-w v`" will run a verification, while "`-w p`" will poke whether the job can run, but ***the job will not be submitted***:

  ```
      % qsub -w v my_test.job
  ```

  or

  ```
      % qsub -w p my_test.job
  ```

  The difference being that "`-w v`" checks against an *empty* cluster, while "`-w p`" validates against the cluster *as is* status.

- ⚠️ By default all jobs are submitted with "-w e", producing an error for invalid requests.
  Overriding it with "`-w w`" or "-w n" can result in jobs that are queued but will never run
  as they request more resources than will ever be available.

- 💡 You can also use the `-verify` flag, to print detailed information about the would-be job as though `qstat -j` was used, including the effects of command-line parameters and the external environment, ***instead*** of submitting job:

  ```
    % qsub -verify my_test.job
  ```

9. Hardware limits

The following table shows the current hardware limits:

Queue name	Number of		Number of CPU per node	Available Memory	
	nodes	slots			Comment
`?ThC.q`	60	5000	40 to 128	>4GB/CPU	high-CPU queues
`?ThM.q`	50	4552	32 to 192	>512GB per node	high-memory queues
`uTxlM.rq`	3	480	96 to 192	>1TB per node	extra large memory queue, restricted
`?Tgpu.q`	3	8 GPUs	-	-	GPU queues, need `-l gpu`
`lTIO.sq`	2	8	-	-	I/O queue to access `/store`
`qrsh.iq`	2	40	-	256GB per node	use `qrsh` or `qlogin` (not `qsh` or `qsub`)
`qgpu.iq`	3	8 GPUs	-	-	GPU interactive queue, use `qrsh -l gpu`

The values in this table change as we modify the hardware configuration, you can verify them with either

```
   % qstat -g c
```

or

```
  % module load tools/local

  % qstat+ -gc
```

and

```
   % qhost
```

or

```
   % qhost+
```

Notes

- We also impose software limits, namely how much of the cluster a single user can grab (see resource limits in Submitting Jobs.)
- If your pending requests will exceed these limits, your queued jobs will wait;
- If you request inconsistent or unavailable resources, you will get the following error message:

  ```
  Unable to run job: error: no suitable queues.
  ```

  You can use "`-w v`" or "`-verify`" to track down why the GE can't find a suitable queue, as described elsewhere on this page.

Last updated 13 May 2024 SGK

Resource Limits

⚠️ While each queue has a set of limits (CPU, memory), the cluster also has some global limits.

1. What are the Resource Limits

There are limits on

1. how many jobs can be queued simultaneously:
    - there can't be more that 25,000 jobs queued at any time,
    - a single user can't queue more than 2,500 jobs, and
    - a job array can't request more that 10,000 tasks.
2. how many jobs can run simultaneously, in particular there is:
    - a limit on how many slots a single user can use (name=u_slots value=640)
    - a limit on how many slots a user can grab in each queue, with fewer slots allowed in queues with longer time limits.
3. how much memory can be simultaneously reserved, in particular
    - a limit on how much memory can be reserved by a single user in each queue.
4. and for some queues how many concurrent jobs a user can have
    - users are limited to one concurrent interactive job, two I/O jobs, and 3 in the uTxlM.q queue.

- The more resources a job uses (more CPU time, more memory), the fewer similar jobs a single user can run concurrently,
    - in other words you can run a lot of small jobs at the same time but fewer very big/long jobs.
- For example, users can't grab more than 71 slots (or CPUs) and 2.6TB of reserved memory concurrently for jobs running in the long-time high-memory queue (`lThM.q`).

⚠️ The actual limits are subject to change depending on the cluster usage and the hardware configuration

2. How to Check the Resource Limits

- To check the global limits:

    ```
    % qconf -sconf global | grep max
    ```

    and the explanation of these parameters can be found in

    ```
    % man 5 sge_conf
    ```

- To check the queue specific resource limits, use

    ```
    % qconf -srqs
    ```

- As of May 2024, this command returns:

`qconf -srqs` returns the following: (click on "Expand source" to view content)

```
{
   name         max_slots_per_user
   description  Limit slots/user for all queues
   enabled      TRUE
   limit        users {*} to slots=840
}
{
   name         max_hC_slots_per_user
   description  Limit slots/user in hiCPU queues
   enabled      TRUE
   limit        users {*} queues {sThC.q} to slots=840
   limit        users {*} queues {mThC.q} to slots=840
   limit        users {*} queues {lThC.q} to slots=417
   limit        users {*} queues {uThC.q} to slots=139
}
{
   name         max_hM_slots_per_user
   description  Limit slots/user for hiMem queues
   enabled      TRUE
   limit        users {*} queues {sThM.q} to slots=840
   limit        users {*} queues {mThM.q} to slots=569
   limit        users {*} queues {lThM.q} to slots=379
```

```
   limit        users {*} queues {uThM.q} to slots=71
}
{
   name         max_xlM_slots_per_user
   description  Limit slots/user for xlMem restricted queue
   enabled      TRUE
   limit        users {*} queues {uTxlM.rq} to slots=480
}
{
   name         qrsh_u_slots
   description  Limit slots/user for interactive (qrsh) queues
   enabled      TRUE
   limit        users {*} queues {qrsh.iq} to slots=16
}
{
   name         total_gpu
   description  Limit GPUs for all users in GPU queues
   enabled      TRUE
   limit        users * queues {qgpu.iq} to num_gpu=4
   limit        users * queues {sTgpu.q,mTgpu.q,lTgpu.q,uTgpu.q} to num_gpu=4
}
{
   name         max_gpu_per_user
   description  Limit GPUs per user in GPU queues
   enabled      TRUE
   limit        users {*} queues {qgpu.iq} to num_gpu=1
   limit        users {*} queues {sTgpu.q} to num_gpu=4
   limit        users {*} queues {mTgpu.q} to num_gpu=3
   limit        users {*} queues {lTgpu.q} to num_gpu=2
   limit        users {*} queues {uTgpu.q} to num_gpu=1
}
{
   name         blast2GO
   description  Limit to set aside a slot for blast2GO
   enabled      TRUE
   limit        users * queues !lTb2g.q hosts {@b2g-hosts} to slots=110
   limit        users * queues lTb2g.q hosts {@b2g-hosts} to slots=1
   limit        users {*} queues lTb2g.q hosts {@b2g-hosts} to slots=1
}
{
   name         bigtmp_space_per_user
   description  Limit total bigtmp concurrent request per user
   enabled      TRUE
   limit        users {*} to big_tmp=25
}
{
   name         idlrt_license_per_user
   description  Limit total number of idl licenses per user
   enabled      TRUE
   limit        users {*} to idlrt_license=102
}
{
   name         io_slots_per_user
   description  Limit slots for io queue per user
   enabled      TRUE
   limit        users {*} queues {lTIO.sq} to slots=8
}
{
   name         max_concurrent_jobs_per_user
   description  Limit the number of concurrent jobs per user for some queues
   enabled      TRUE
   limit        users {*} queues {uTxlM.rq} to no_concurrent_jobs=3
   limit        users {*} queues {lTIO.sq} to no_concurrent_jobs=2
   limit        users {*} queues {qrsh.iq} to no_concurrent_jobs=1
   limit        users {*} queues {qgpu.iq} to no_concurrent_jobs=1
}
```

⚠️ Note that these values get adjusted as needed.

- The explanation of the resource quota set (rqs) can be found in

```
% man 5 sge_resource_quota
```

- To check how much of these resources (queues quota) are used overall, or by your job(s), use:

  ```
  % qquota
  ```

  or

  ```
  % qquota -u $USER
  ```

- You can also inquire about a specific resource (`qquota -l mem_res`), and use the local tools (`module load tools/local`) `qquota+` to
  - get a nicer printout of the reserved memory,
  - get the % of usage with respect to its limit
- like in

  ```
  % qquota+ +% -l slots -u hpc
  ```

  (more info via `qquota+ -help` or `man qquota+`.)

- To check the limits of a specific queue (CPU and memory), use

  ```
  % qconf -sq sThC.q
  ```

  and the explanation of these parameters can be found in

  ```
  % man 5 queue_conf
  ```

  under the `RESOURCE LIMITS` heading.

3. NOTES

- You can submit a job and tell the GE to let it start only after another job has completed, using `-hold_jid <jobid>` flag to `qsub`:

  ```
  % qsub -N FirstOne pre-process.job
  Your job 12345678 ("FirstOne") has been submitted
  % qsub -hold_jid 12345678 -N SecondOne post-process.job
  Your job 12345679 ("SecondOne") has been submitted
  ```

- You can be more sophisticated (or use `qchain` see below):

Script that submit 3 jobs that must run sequentially, using C-shell syntax

```
#!/bin/csh
#
set parameter = $1
set name = $2
#
set jid1 = `qsub -terse -N "pre-process-$name" pre-process.job $parameter`
echo $jid1 submitted '("'pre-process-$name'")'
set jid2 = `qsub -terse -hold_jid $jid1 -N "process-$name" process.job $parameter`
echo $jid2 submitted '("'pre-process-$name'")'
set jid3 = `qsub -terse -hold_jid $jid2 -N "post-process-$name" post-process.job $parameter`
echo $jid3 submitted '("'post-process-$name'")'
```

This example will submit 3 jobs: `pre-process.job`, `process.job` and `post-process.job` to be run sequentially,

- each takes one argument, the parameter,
- and is given a compounded name.
- The embedded directives in the three job scripts may request different resources, like
  - lots of memory for pre-processing,
  - lots of CPUs for processing, and
  - neither for post processing.

  This way a task is broken up to avoid grabbing more resources than needed at each step.

- You can use the `qchain` tool by loading the `tools/local` module, to submit jobs that must run sequentially.

```
module load tools/local
qchain *.job
```

will submit the job files that match "`*.job`" in the order given by "`echo *.job`".

By using quotes, as follows:

```
module load tools/local
qchain '-N start first.job 123' '-N crunch second.job 123' '-N post-process finish.job 123'
```

`qchain` allows you to pass arguments to both `qsub` and the job scripts.

- You can limit how many jobs you submit with the following trick:

How to limit the number of jobs submitted, using C-shell syntax

```
# define how many jobs to queue
@ NMAX = 250
#
loop:
  @ N = `qstat -u $USER | tail --lines=+3 | wc -l`
  if ($N >= $NMAX) then
    sleep 180
    goto loop
  endif
#
```

This example counts how many jobs you have in the queue (running and waiting) using `the command` `qstat` (and `tail` and `wc -l`) and pauses for 3 minutes (180 seconds) if that count is 250 or higher.

You would include these lines in a script that submits a slew of jobs, but should not queue more than a given number at any time (to count only the queued jobs, add `-s p` to `qsub`).

- Or you can use the tool `q-wait` (needs the module `tools/local`), that takes an argument and two options:
  `% q-wait blah`
  will pause until you have no job whose name has the string '`blah`' left queued or running.
- The options allow you to specify the number of jobs, and how often to check, i.e.:
  `% q-wait -N 125 -wait 3600 crunch`
  will pause until there are 250 or fewer jobs whose name has the string '`crunch`' left queued or running, checking once an hour.

- ⚠ Avoid using the `-V` flag to `qsub`
  - The `-V` flag passes all the active environment variables to the script.
  - While it may be convenient in some instances, it creates a dependency on the precise environment configuration when submitting the job,
    thus the same job script may fail when it is submitted at a later time (or from a different log in) from a different configuration.

Last updated 13 May 2024 SGK

Job Monitoring

1. Introduction

After submitting a job, or a set of jobs, with `qsub`, you can

- check on your job(s) with the command `qstat,`
- kill your job(s) with the command `qdel,`
- alter the requested resources of a queued job with `qalter` ,
- check on a finished job with `qacct` ,
- use Hydra-specific home-grown tools.

2. How to Check on Jobs

The `qstat` command returns the status of jobs in the queue (`man qstat`), here are a few usage examples:

`qstat -u $USER`	shows only your jobs.
`qstat -u '*'`	shows everybody's jobs.
`qstat -s r`	shows only running jobs.
`qstat -s p`	shows only pending jobs.
`qstat -r`	shows also requested resources and the full job name.
`qstat -s r -u $USER -g t`	shows master/slave info for parallel jobs
`qstat -s r -u $USER -g d`	shows task-ID for array jobs
`qstat -j 4615585`	produces a more detailed output, for a specific job ID.
`qstat -explain E -j 4615585`	produces the explanation for the error state of a specific job, specified by its job ID.

The job status abbreviations, returned by `qstat,` correspond to

`qw`	pending (waiting in queue)
`r`	running
`t`	in transfer (typically from `qw` to `r`)
`Eqw`	error and waiting in queue (for ever)
`d`	marked for deletion

- Jobs in the `Eqw` status will not run, and the reason for the error status can be found via `qstat -explain E -j <jobid>`
- Jobs in the `qw` status are waiting for the requested resources to become available, or because you have reached some resource limit
  - These jobs can be `qalter`'ed
  - If your job is lingering in the queue (remains in `qw` for a long time, i.e., hours to days) and you have not reached some resource limit, you are most likely requested a scarce resource, or more of a resource than is available. In this case, feel free to contact us.
- Jobs in the `t` status are in most cases about to get started.
- Jobs in the `d` status are about to be deleted and/or killed.

We provide a tool, `q+`, that uses `qstat` to provide an easier way to monitor jobs and the status of the queue.

3. How to Delete/Kill a Job or Jobs

These are instructions for how to stop a running job or delete a submitted job that has not started running yet.

The `qdel` command (`man qdel`) allows you to either:

- delete a job from the queue (for job in the "`qw`" status), or
- kill a running job (for jobs in the "`r`" status).

You delete/kill a specific job by its job id as follows:

```
% qdel 4615585
```

or you can kill ***all*** your jobs (queued and running) with

```
% qdel -u $USER
```

💡 Here is a trick to kill a slew of jobs, but not all of them:

```
qstat -u $USER | grep $USER | awk '{print "qdel", $1} > qdel.sou
[edit the file qdel.sou]
source qdel.sou
```

This example

- uses `grep` to filter the lines from `qstat` that have your username, and then
- uses `awk` to produce a list of lines like "`qdel <jobid>`", and
- saves the result to a file `(qdel.sou`);

you then

- edit that file to keep the lines you want, and
- use `source` to execute each line of that edited file as if you had typed them.

💡 You can use `grep` to better filter the output of `qstat`, like this:

```
qstat -u $USER | grep $USER | grep rax | awk '{print "qdel", $1}' > qdel.sou
[edit the file qdel.sou]
source qdel.sou
```

This example

- filters the output of `qstat` for lines with your user name and
- with the string "`rax`" to identify a sub set of jobs via their job name.

💡 You can add `"-s r"` or "`-s p`" to `qstat` to limit the list of job IDs to only your running or pending (queued) jobs.

4. Modifying a Queued Job

The `qalter` command allows you to alter (modify) the properties of a job, namely its requested resources or parameters.

You can alter

- most of the properties of a queued job;
- and a few of its properties once it has started.

For example:

you can	with
move a queued job to a different queue with	`% qalter -q mThC.q <jobid>`
change the name of the job's output file	`% qalter -o run-3.log <jobid>`
change whether you want email notifications	`% qalter -m abe <jobid>`
change the requested amount of CPU	`% qalter -l s_cpu=240:: <jobid>`

where `<jobid>` is the job ID (see `man qalter`.)

5. Checking on a Completed Job

The qacct command shows the GE accounting information and can be used to check on

- the resources used by a given job that has completed, and
- its exit status.

For example:

```
% qacct -j <jobid>
```

will list the accounting information for a finished job with the given job ID.

It will show the following useful information:

qname	name of the queue the job ran in
hostname	name of the (master) compute node the job ran on
taskid	the task ID (for job arrays)
qsub_time	when the job was queued
start_time	when the job started
end_time	when the job ended
granted_pe	what parallel environment was used
slots	how many slots were allocated
failed	did the job fail to complete (1 means job did fail, i.e., was killed by GE b/c memory or time limit exceeded)
exit_status	the job script exit status (0 means job script completed OK)
ru_wallclock	wall clock time elapsed, in seconds
ru_utime	consumed user time, as reported by the O/S (actual CPU time), in seconds
ru_stime	consumed system time, as reported by the O/S (non-CPU time, usually related to I/O, or other system wait), in seconds
cpu	CPU time computed, as measured by the GE (ru_utime+ru_stime may not add to the cpu time), in seconds
mem	total memory*time used in GB seconds: the mean memory usage is mem/cpu in GB
io	a measure of the I/Os operations executed by the job
maxvmem	the maximum amount of memory used by the job at any time during its execution

💡 Users that run jobs in the high memory queues can use this to check if they used close to the amount of memory they have reserved.

The qacct command can also be used to check on the past usage of a given user. For example:

```
% qacct -d <ndays> -o <username>
```

will return the usage statistics of the user specified in <username>, over the past <ndays> days (use man qacct for more details).

You can get details information for each job that ran, with the "-j" option, as follow:

```
% qacct -d <ndays> -o <username> -j > qacct.log
```

and save its output, if long, to a file (qacct.log). You can parse that file with the command egrep.

For example, to check all the jobs the user hpc ran over the past 3 days:

```
% qacct -d 3 -o hpc -j > qacct.log
```

You than parse the output with:

```
% egrep 'jobname|jobnumber|failed|exit_status|cpu|ru_w|===' qacct.log > qacct-filtered.log
```

⭐ The command `egrep` is used to print any line that has one of strings in the quoted list separated by the '|' ('|' that means "or" in this context, see `man egrep`).

6. Additional Tools

There is a "*better*" `qstat, namely qstat+` and a "*better*" `qacct, namely qacct+`

1. `qstat+` runs `qstat` but display things with more details and additional flexibility.
2. `qacct+` queries a database where the accounting info is ingested, it runs faster that `qacct` and returns more info and has additional flexibility.

See the additional tools page for instruction how to use these.

Last Updated 17 Nov 2021 SGK.

Help Choosing a Queue

1. What Queue Should I use?
2. Why Can't I Queue that Job?
3. Why Is my Job Queued but not Running?

1. What Queue Should I Use?

To choose a queue, you need to know

1. whether is it a serial (single CPU) or parallel (multiple CPUs) job,
2. if it is a parallel job, what kind,
3. how much memory this job will need,
4. how much CPU time it will require.

Indeed:

If your computation will use	your job script needs to	`qsub` option needed/recommended
more than one CPU (parallel jobs need)	request a `PE` and `N` slots	`-pe <pe-name> N or -pe <pe-name> N-M`
more than 2GB/CPU of memory	reserve the required memory	`-l mres=X,h_data=X,h_vmem=X`
more than 8GB/CPU of memory	use a high-memory queue, and reserve the required memory	`-l mres=X,h_data=X,h_vmem=X,himem`
up to T hours of CPU (per CPU)	specify the required amount	`-l s_cpu=T:0:0`
	or specify the queue	`-q mThC.q`
no idea how much CPU	use an unlimited, low priority queue	`-q uThC.q -l lopri`

- `X` can be something like 2GB
- `T` can be something like 240 (for 240 hours or 10 days)
- You may need to combine PE, memory and CPU resource requests.
- Remember, that the more resources your job requests, the fewer concurrent similar jobs can run at any tim
- Similar jobs will need similar resources, so when in doubt and before queuing a slew of similar jobs:
  - run one job and monitor its resource usage, then
  - queue the other jobs after trimming the requested resources (CPU and memory).

2. Why Can't I Queue that Job?

There can be different reasons why a job is rejected:

- inconsistency in your resources request, like asking for more CPU or memory that the limit of a given queue;
- unavailable resources, like asking for more CPUs or more memory on a single node than exists on any compute nodes;
- exceeding resource limits, like asking for more CPUs than are allowed per user in a given queue.

Use the `-w v` or the `-verify` flag to `qsub`, see queue selection validation and/or verification, to check a job script file.

3. Why Is my Job Queued but not Running?

There can be different reasons why a job remains in the queue:

- the requested resources are not available, like there is no compute node with the requested number of CPUs or amount of memory currently available;
- the user resource quota has been reached, like the allowed total amount of CPUs or memory used by a single user was reached.

Use the command `qconf -srqs` or `qquota`, see how to check under resource limits.

Last Updated 13 May 2024 SGK

Help Writing a Job Script

QSubGen: is a Job Script Generator for Hydra

There is a web page with an app to help you choose a queue and write a job script.

It writes out the embedded directives as you specify the resources your computation will need and can help selecting which modules to load

- Go to the QsubGen page to run the job script generator, that page is only accessible from a trusted computer, with VPN turned on, or from `telework.si.edu.`

Last updated 13 May 2024 SGK

Where to Find Examples

You can find examples of simple/trivial test cases with source code, `Makefile`, job script files and resulting log files on Hydra under `~hpc/examples.`

- The examples are organized as follows:

bigtmp/	example using big temp storage
c++11/	example using the C++11 extension
containers/	example using singularity
gpu/	GPU examples and (old) timings
gsl/	simple test that uses GSL
hybrid/	examples for using the hybrid PE
idl/	examples for running IDL/GDL/FL jobs
java/	example running JAVA
lapack/	example linking with LAPACK and Intel's MKL
matlab/	example using MATLAB (runtime)
memtest/	examples for large memory use and reservation
misc/	miscellaneous
mpi/	examples using MPI: with each compiler (gcc, Intel, NVIDIA) for various implementation (vendor, MVAPICH, OPENMPI)
openmp/	example using OpenMP
python/	list of different implementations of PYTHON available on Hydra
serial/	simple (hello world) serial job, for each compiler (gcc, Intel, NVIDIA)
ssd/	example using local SSD

- You can use the command `find` to get a list of all the sub-directories under `~hpc/examples,` i.e.:

```
% find ~hpc/examples -type d -print
```

Last updated 13 May 2024 SGK

Running Jupyter on Hydra

Jupyter notebooks deliver a "literate programming" interface with Python that combines MarkDown-based documentation with code cells.

Using Jupyter on Hydra is especially helpful for:

- Keeping a record of a your analysis in notebook format, that can be accessed later
- Working with images or visualizations that cannot be viewed through the command line interface.

From the log-in node, access the interactive queue.

```
qrsh -pe mthread [N]
```

...where N is the number of threads you would like to request.

If you have access to the GPU nodes, you can use `qrsh -l gpu`.

You will notice that your prompt will change from `[user@hydra-login01]` to `[user@computer-XX-XX]` where *XX-XX* is the name of the Hydra compute node you are now on.

Then load the conda module, and activate a specific module with jupyter, if necessary.

```
module load ~/modulefiles/miniconda
conda activate jupyter
```

If you have access to the GPU nodes, you will also need to run `module load nvidia/24/cuda` *to load the CUDA libraries. You can test that it worked with the command* `nvidia-smi`.

Now run the following command to launch a jupyter notebook. It will run it from the directory you are in, so make sure you are "above" the directory where you want your notebooks to live. The interactive queue puts you into your `/home` directory by default.

```
 jupyter lab --no-browser --ip=`hostname` --port=8888
```

This will launch a Jupyter lab server, and produce a bunch of output looking something like this:

```
[I 14:38:44.628 NotebookApp] Serving notebooks from local directory: /pool/genomics/triznam
[I 14:38:44.628 NotebookApp] The Jupyter Notebook is running at:
[I 14:38:44.628 NotebookApp] http://compute-08-31.local:8888/?
token=e54bd4f1469387555c114697278fe2ff10089cbf723c595b
[I 14:38:44.628 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip
confirmation).
[C 14:38:44.636 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/triznam/.local/share/jupyter/runtime/nbserver-25198-open.html
    Or copy and paste one of these URLs:
        http:compute-08-31.local:8888/?token=e54bd4f1469387555c114697278fe2ff10089cbf723c595b
```

Now on your local computer, open up a terminal, and run the following command:

```
ssh -N -L 8888:compute-XX-XX:8888 triznam@hydra-login01.si.edu
```

...where *XX-XX* is the compute node that the `qrsh` command placed you. The first 8888 is the port of your local computer. If you are already running a local Jupyter notebook, then simply change this to a free port. The second 8888 is the port of the compute node, so change this if you used a different port in the `jupyter notebook` command above.

If it worked, it will prompt you for your Hydra password. Enter it.

Now open a browser and go to http://localhost:8888. It should be a Jupyter-looking window, but will ask for a token or password before you can continue. Copy and paste the long token from the end of the URL that was printed out when you launched Jupyter on Hydra above. In that case it would be: *e54bd4f1469387555c114697278fe2ff10089cbf723c595b*

Setting up a Jupyter Server password

To avoid having to paste that long token every time, you can create a password for yourself.

```
jupyter server password
```

(and then enter a memorable password)

Now next time you launch Jupyter, you will be prompted to enter a password instead of that unique token.

Creating conda environment specific kernels

To create a Jupyter *kernel* that is pre-built with all of the conda packages from your conda environment, do the following.

```
conda activate env_name
python -m ipykernel install --user --name="env_name"
```

Then you should see that kernel as a dropdown in Jupyter notebook or as a notebook option in Jupyter Lab.

Having kernels is also useful for running notebooks as a job using PaperMill.

Last updated 13 May 2024 MK/SGK

Cluster Monitoring

1. Cluster Status

The command

```
% qstat -g c
```

returns the cluster status, in a tabular form, i.e.:

```
CLUSTER QUEUE                   CQLOAD   USED    RES  AVAIL  TOTAL aoACDS  cdsuE
--------------------------------------------------------------------------------
all.q                             -nan      0      0      0      0      0      0
lTIO.sq                           0.00      0      0      8      8      0      0
lTb2g.q                           0.27      0      0      2      2      0      0
lTgpu.q                           0.00      0      0     64     64      0      0
lThC.q                            0.38    634      0   4374   5008      0      0
lThM.q                            0.37    390      0   4162   4552      0      0
lThMuVM.tq                        0.28      0      0    384    384      0      0
mTgpu.q                           0.00      0      0     64     64      0      0
mThC.q                            0.38   1331      0   3677   5008      0      0
mThM.q                            0.37     14      0   4538   4552      0      0
qgpu.iq                           0.00      4      0     60     64      0      0
qrsh.iq                           0.00     11      0     29     40      0      0
sTgpu.q                           0.00      0      0     64     64      0      0
sThC.q                            0.38      5      0   5003   5008      0      0
sThM.q                            0.33      0      0   5032   5032      0      0
uThC.q                            0.38      8      0   5000   5008      0      0
uThM.q                            0.37     70      0   4482   4552      0      0
uTxlM.rq                          0.00      0      0    480    480      0      0
```

You can also use

```
% qstat+ -gc
```

(no space in -gc) to get:

```
---- queue ----  ----- #nodes ---- - ---------- #slots ----------- - ------------
 name       load  total avail  down - total used  resvd  down avail - %full  %eff

sThC.q   1880.5    63    63     0 -  5008    5      0     0  5003 -   0.1
mThC.q   1880.5    63    63     0 -  5008 1331      0     0  3677 -  26.6
lThC.q   1880.5    63    63     0 -  5008  634      0     0  4374 -  12.7
uThC.q   1880.5    63    63     0 -  5008    8      0     0  5000 -   0.2  95.1

sThM.q   1662.8    55    55     0 -  5032    0      0     0  5032 -   0.0
mThM.q   1662.5    52    52     0 -  4552   14      0     0  4538 -   0.3
lThM.q   1662.5    52    52     0 -  4552  390      0     0  4162 -   8.6
uThM.q   1662.5    52    52     0 -  4552   70      0     0  4482 -   1.5 350.7

uTxlM.rq    0.2     3     3     0 -   480    0      0     0   480 -   0.0   0.0

lTIO.sq     0.0     2     2     0 -     8    0      0     0     8 -   0.0   0.0

sTgpu.q     0.0     1     1     0 -    64    0      0     0    64 -   0.0
mTgpu.q     0.0     1     1     0 -    64    0      0     0    64 -   0.0
lTgpu.q     0.0     1     1     0 -    64    0      0     0    64 -   0.0   0.0

qgpu.iq     0.0     1     1     0 -    64    4      0     0    60 -   6.2
qrsh.iq     0.1     2     2     0 -    40   11      0     0    29 -  27.5
```

⚠ the actual numeric values will be slightly different when you run these commands, since they reflect the precise configuration and the load.

2. Compute Nodes Status

The command

```
% qhost
```

returns the list of hosts (compute nodes) and their respective properties.

⚠ Under UGE, qhost alone returns more columns (equiv to qhost -cb under SGE). The option -ncb returns the same columns as in SGE.

You can restrict the list by specifying the hosts, like

```
% qhost -h compute-64-02 compute-64-03
```

but you can't use `RE`s. So you use a filter, like `egrep`, to parse its output:

```
% qhost | egrep 'LOAD|e-[46]'
```

This will print any line with either the string '`LOAD`' or a line that matches the `RE` "`e-[46]`", and will thus match `compute-4`, `compute-6`, etc....

The utility `egrep` combined with `RE`s (regular expressions) can be a very powerful filter.

The command `qhost` takes the "`-q`" or the "`-j`" option to show the queues or the jobs associated with each host(s):

`qhost -q -h compute-64-02`	show which queues include the compute node 64-02
`qhost -j -h compute-64-02`	show which jobs are running on the compute node 64-02

There is also a `qhost+` command, see the Additional Tools page.

3. Query the Cluster Configuration

The command `qconf` is used to both set and query the queue configuration.

All the options of `qconf` that start with `-s` correspond to a query: i.e., show something.

The following options may be useful:

`-sc`	show complex attributes
`-sconfl`	show a list of all local configurations
`-sconf [host_list]`	show configurations
`-shgrpl`	show host group list
`-shgrp group`	show host group
`-srqsl`	show resource quota set list
`-srqs [rqs_list]`	show resource quota set(s)
`-spl`	show all parallel environments
`-sp pe-name`	show a parallel environment
`-sql`	show a list of all queues
`-sq [queue_list]`	show the given queue
`-ssconf`	show scheduler configuration
`-sul`	show a list of all userset lists
`-su listname_list`	show the given userset list

⚠️ Use the command

```
% qconf -srqs
```

to query the resource quota set, *i.e.* the limits on queues, or

```
% qconf -srqs u_slots
```

to query a specific quota.

💡 Use the command

```
% qconf -sq sThM.q
```

to show the configuration of the `sThM.q` queue. Some of the options to the command `qconf` take `RE`s, so for example the command:

```
% qconf -sq '?ThM.q' | egrep 'qname|s_cpu|s_rt'
```

returns the soft CPU and R/T limits for all the hi-mem queues, using egrep to filter the output of `qconf`, namely:

```
qname                 lThM.q
s_rt                  1440:00:00
s_cpu                 720:00:00
qname                 mThM.q
s_rt                  144:00:00
s_cpu                 72:00:00
qname                 sThM.q
s_rt                  14:00:00
s_cpu                 7:00:00
qname                 uThM.q
s_rt                  INFINITY
s_cpu                 INFINITY
```

4. Cluster Status Web Page

We also maintain a cluster status web page that can be accessed at two locations:

- from a trusted machine, here (at `.si.edu`), or
- from anywhere, here (at `.cfa.harvard.edu`).
- You can specify up to 3 arguments to the URL, especially useful if you bookmark it, to specify either:
    1. the sorting in the cluster snapshot graph with `sortby=`, like in `sortby=nCPU`
    2. the length of the plots vs time with `len=`, like in `len=7d`
    3. which user's job(s) to highlight with `user=`, like in `user=hpc`
- by adding "`?sortby=nCPU&user=hpc&len=15d`" to the URL; valid values for each parameter are those listed in the corresponding drop down menus.

These pages give you a good overview of the cluster current status and past usage, and include the disk space usage information.

Last Updated 13 May 2024 SGK/PBF

Disks Space and Usage

1. Introduction: What Disks to Use

The disk space available on the cluster is mounted off a set of dedicated devices:

1. A NetApp filer, via NFS,
2. A two NSD GPFS, via the Infiniband fabric,
3. A low cost NAS, via NFS on only a subset of nodes.

The available disk space is divided in several area (aka volumes, filesets or partitions):

- a small partition for basic configuration files and small storage, the `/home` partition,
- a set of medium size partitions, the `/data` partitions,
- a set of large partitions, the `/pool` partitions,
- a set of very large partitions for temporary storage, the `/scratch` partitions,
- a set of medium, size low-cost, partitions, the `/store` partitions

⚠️ The public `/pool` and `/scratch` partition are scrubbed: files older that 180 days are automatically removed ⚠️

- Consult the Scrubber and How to Request Scrubbed Files to be Restored page for more information.

SSD

A subset of nodes have local SSDs (solid state disks) that can be used for applications that require very high I/O rates, and will complete faster when using SSDs.

- Jobs that do not perform intensive I/O should not use the SSDs - this is a scarce shared resource.

These disks are local to the compute nodes, hence:

- you cannot see the SSDs from either login nodes,
- your job will be able to use the SSD ***only while the job is running***, hence your job need to be adjusted accordingly and request SSD space.
- If your job exceeds the amount of SSD space requested, your job won't be able to write any longer to the SSD,
- consult the How to Use Local SSD Space page for more information.

Remember

- We impose quotas:
  - limits on how much can be stored on each disk (partition/volume/fileset) by each user, and
  - we monitor disk usage;
- `/home` should not be used to keep large files, use `/pool,` `/scratch,` or, `/data` instead;
- `/pool` and `/scratch` are for active temporary storage (i.e., while a job is running).
- If your job(s) need a lot of disk space or your job(s) perform a lot of I/Os, use `/scratch` rather than `/pool`.
  - Public space on both partitions (`/pool` and `/scratch`) are scrubbed: old stuff is deleted to make sure there is space for active users.
- None of the disks on the cluster are for long term storage:
  - please copy your results back to your *home* computer and
  - delete what you don't need any longer.
- While the disk systems on `Hydra` are highly reliable, most of the disks on the cluster are ***not*** backed up, although:
  - some partitions have snapshots enabled: this allows you to 'undelete' files that were recently deleted (see How to Recover Old or Deleted Files using Snapshots)
  - `/home` and `/data` are backed up to AWS Glacier for disaster recovery (DR).
- Once you reach your quota you won't be able to write anything on that partition until you delete stuff.
- ⚠️ Do not keep a very large number of files in the same directory:
  - 💡best practice is to keep less then 5,000 - 50,000 files in the same directory.
- If you keep too many of them in the same directory:
  - you may not be able to write more files,
  - listing the content of such directory will be exceedingly slow.
- What to do instead?
  - Use subdirectories to better organize your files (and your work).

Last Updated 19 Jan 2024 SGK/PBF.

Disks Space Configuration

1. Configuration

As of 13 May 2024

		Quotas per user				
Name	**Disk**	**Space**	**No. of files**			**Purpose**
	size	**soft/hard**	**soft/hard**	**Snapshots?**	**Scrubbed?**	
see note (1)	(2)	(3)	(4)	(5)	(6)	**Public disks**
`/home`	`25T`	`480G/512G`	`9M/10M`	yes: 4 wk	no	For your basic configuration files, scripts and job files ▪ your limit is low but you can recover old stuff up to 4 weeks.
`/data/public`	`105T`	`4.3T/4.5T`	`9.5M/10M`	yes: 2 wk	no	For important and smaller files like final results, configurations, etc. ▪ your limit is medium, you can recover old stuff, so disk space is not released right away.
`/pool/public`	`190T`	`7.1T/7.5T`	`17M/18M`	no	yes	For storing large files ▪ your limit is high, and disk space is released right away.
`/scratch/public`	`400T`	`14T/15T`	`36M/38M`	no	yes	For storing very large file (more space than under `/pool/public`) ▪ your limit is highest, ▪ the underlying system is fastest.
						Project specific disks (/pool)
`/pool/kozakk`	`11T`	`10.5/11.0T`	`26/278M`	no	no	STRI/Krzysztof Kozak
`/pool/nmnh_ggi`	`21T`	`15.0/15.8T`	`37/40M`	no	no	NMHN/GGI
`/pool/sao_access`	`21T`	`15.0/15.8T`	`37/40M`	no	no	SAO/ACCESS
`/pool/sao_rtdc`	`11T`	`2.8/3.0T`	`2.5/3.0M`	no	no	SAO/RTDC
`/pool/sylvain`	`30T`	`29/30T`	`71/75M`	no	no	SAO/Sylvain Korzennik
						Project specific disks (/scratch)
`/scratch/bradys`	`25T`	`–`	`–`	no	no	NMNH/Seán Brady/BRADY_LAB
`/scratch/kistlerl`	`100T`	`98/100T`	`254/260M`	no	no	NMNH/Logan Kistler
`/scratch/meyerc`	`25T`	`24/25T`	`52/62M`	no	no	NMNH/Christopher Meyer
`/scratch/nmnh_corals`	`20T`	`18/20T`	`52/54M`		no	MNNH/Corals
`/scratch/nmnh_ggi`	`77T`	`75/77T`	`195/200M`	no	no	NMNH/GGI
`/scratch/nmnh_lab`	`25T`	`4/5T`	`10/12M`	no	no	NMNH/LAB
`/scratch/nmnh_mammals`	`25T`	`24/25T`	`66/68M`		no	NMNH/Mammals
`/scratch/nzp_ccg`	`35T`	`34/35T`	`89/92M`	no	no	NZP/Michael Campana/CCG

/scratch /sao_atmos	250T	245/250T	252/261M	no	no	SAO/ATMOS
/scratch /sao_cga	25T	7/8T	18/20M	no	no	SAO/CGA
/scratch /sao_tess	50T	36/40T	94/210M	no	no	SAO/TESS
/scratch /schultzt	25T	–	–	no	no	NMNH/Ted Schultz/SCHULTZ_LAB
/scratch /stri_ap	25T	4/5T	10/12M	no	no	STRI/W. Owen McMillan/STRI_AP
/scratch /sylvain	70T	68/70T	176/182M	no	no	SAO/Sylvain Korzennik
/scratch /usda_sel	25T	24/25T	52/62M	no	no	NMNH/Christopher Owen/USDA_SEL
/scratch/wrbu	40T	38/40T	99/100M	no	no	WRBU
						Near line (/store)
/store/admin	20T	–	n/a	yes: 8w	no	Sys Admin
/store/public	175T	5/5T	n/a	yes: 8w	no	Public, available upon request
/store/bradys	40T	–	n/a	yes: 8w	no	NMNH/Sean Brady/BRADY_LAB
/store/nmnh_ggi	90T	–	n/a	yes: 8w	no	NMNH/GGI
/store /sao_atmos	300TB	–	n/a	yes: 8w	no	SAO/ATMOS
/store/sylvain	100TB	–	n/a	yes: 8w	no	SAO/Sylvain Korzennik
/store/schultzt	40TB	–	n/a	yes: 8w	no	NMNH/Ted Schultz/SCHULTZ_LAB
/store/wrbu	40TB	–	n/a	yes: 8w	no	WRBU
						Extra
/data/admin /pool/admin /scratch/admin /store/admin	2T 10T 20T 20T			no	no	sys admin
/scratch/dbs	10T			no	no	data bases
/scratch/tmp	100T			no	no	scratch space for jobs that need a big temp, via `-l bigtem=XX`

Notes

1. The previous paths /pool/sao or /scratch/genomics still work, but they correspond to /pool/public/sao or /pool/public/genomics, respectively.
2. Sizes are indicative, since we adjust them as needed.
3. the notation 1.8T/2.0T means that the soft limit is 1.8TB and the hard limit is 2.0TB of disk space.
4. the notation 4M/5M means that the soft limit is 4 million `inodes` and the hard limit is 5 millions.
5. Snapshots allow to restore deleted files, up to that many weeks.
6. Public space under `/pool` and `/scratch` is scrubbed weekly - files older than 180 days are removed.

2. Storage Efficiency

- It is inefficient to store a slew of small files and if you do you may reach your `inodes` quota before your space quota (too many small files).
  - Some of the disk monitoring tools show the `inode` usage.
  - If your `%(inode)>%(space)` your disk usage is inefficient,
    consider archiving your files into `zip` or `tar-compressed` sets.

- While some of the tool(s) you use may force you to be inefficient while jobs are running, you should remember to
  - remove useless files when jobs have completed,
  - compress files that can benefit from compression (with `gzip`, `bzip2` or `compress`), and
  - archive a slew of files into a `zip` or a `tar-compressed set`, as follows:
    ```
    % zip archive.zip dir/
    ```
    or
    ```
    % tar -czf archive.tgz dir/
    ```
    both examples archive the content of the directory `dir/` into a `single zip` or a `tgz` file. You can then delete the content of `dir/` with
    ```
    % rm -rf dir/
    ```

- You can unpack each type of archive with
  ```
  % unzip archive.zip
  ```
  or
  ```
  % tar xf archive.tgz
  ```

3. Scrubbing of Public Disks

- We have implement a FIFO (first in first out) scrubbing model, where old files are deleted, aka scrubbed, to keep free space on the public disks.
  - There is an age limit, meaning that only files older than 180 days get deleted.
  - Older files get deleted before the newer ones (FIFO),
  - We run a scrubber on a weekly basis.
  - Consult the Scrubber and How to Request Scrubbed Files to be Restored page for additional information.
- In any case, we ask you to remove from `/pool` and `/scratch` files that you do not need for active jobs.

4. Dedicated Project Disk Space

- For projects that would benefit from dedicated disk space, such space can be secured with project's specific funds when we expand the disk farm (contact us).

Last Updated 13 May 2024 SGK/PBF

How to Copy Files to/from Hydra

Remember

⚠️ When copying to Hydra, especially large files, be sure to do it to the appropriate disk (and not `/home` or `/tmp`).

1. From a Computer running Linux

- You can copy files to/from `hydra` using `scp, sftp or rsync:`
  - to `Hydra` you can only copy from *trusted* hosts (computers on SI or SAO/CfA trusted network, or VPN'ed),
  - from `Hydra` to any host that allows external `ssh` connections (if you can `ssh` from Hydra to it, you can `scp, sftp and rsync` to it).

- For large transfers (over 70GB, sustained), we ask users to use `rsync`, and limit the bandwidth to 20 MB/s (70 GB/h), with the "`--bwlimit=`" option:
  - `rsync --bwlimit=20000 ...`
    If this pose a problem, contact us.

  - Baseline transfer rate from SAO to HDC (Herndon data center) is around 600 Mbps, single thread, or ~72 MB/s or ~252 GB/h
    The link saturates near 500 Mbps (50% of Gbps) or 62 MB/s or 220 GB/h

- Remember that `rm, mv` and `cp` can also create high I/O load, so consider to
  - limit your concurrent I/Os: do not start a slew of I/Os at the same time, and
  - serialize your I/Os as much as possible: run one *after* the other.

NOTE for SAO Users:

💡 Access from the "*outside*" to SAO/CfA hosts (computers) is limited to the *border control hosts* (`login.cfa.harvard.edu` and `pogoN.cfa.harvard.edu`), instructions for tunneling via these hosts is explained on

- the CF's SSH Remote Access page, or
- the HEAD Systems Group's SSH FAQ page.

2.From a Computer Running MacOS

A trusted or VPN'd computer running MacOS can use `scp, sftp or rsync`:

- Open the `Terminal` application by going to `/Applications/Utilities` and finding `Terminal`.
- At the prompt, use `scp, sftp or rsync, after cd'ing to the right place.`
- For large transfers limit the bandwidth and use `"rsync --bwlimit=4000".`

Alternatively you can use a GUI based `ssh/scp` compatible tool like FileZilla. Note, `Cyberduck` is not recommended because it uses a lot of CPU cycles on Hydra.

You will still most likely need to run VPN.

3. From a Computer Running Windows

💡 You can use `scp, sftp or rsync` if you install Cygwin - Note that Cygwin includes a X11 server.

Alternatively you can use a GUI based `ssh/scp` compatible tool like FileZilla or WinSCP. Note, `Cyberduck` is not recommended because it uses a lot of CPU cycles on Hydra.

You will still most likely need to run VPN.

Last Updated 08 Oct 2021 SGK/PBF.

Using Dropbox

- Files can be exchanged with Dropbox using the script Dropbox-Uploader, which can be loaded using the `tools/dropbox_uploader` module and running the `dropbox` or `dropbox_uploader.sh` script.
- Running this for script for the first time will give instructions on how to configure your Dropbox account and create a `~/.dropbox_uploader` configuration file with authentication information.
- Using this method will not sync your Dropbox, but will allow you to upload/download specific files.

Last Updated 08 Oct 2021 SGK/PBF.

Using Firefox Send

- Firefox Send ~~is~~ was a free online file-sending service (or a file exchange mechanism).
- Using this system along with the command ffsend available on Hydra (`module load tools/ffsend`), you can transfer files to/from Hydra *without needing VPN.*
- Firefox Send is a two-step process,
  1. you first upload a file (or a set of files packed in an archive) to the Firefox Send server which will generate a unique URL for the upload, and
  2. you download the file using that URL and recover the file's original name.

⚠️ While Firefox discontinued this service, you can still use it, using Tim Visée replacement, at https://send.vi.ee ⚠️

~~You can upload up to 1GB at a time, and if you sign up for a Firefox account, that limit increases to 2.5GB.~~

Optional: `ffsend` has options for setting a password and expiration. See `ffsend help` for more information.

Example 1: Sending *to* Hydra

a. Uploading files from your local machine (workstation/laptop) using the Send website:

1. Open the ~~Firefox~~ Send website (`send.vis.ee`) from any browser.
2. Choose a file to upload, and optionally:
   a. use `tar`, `zip` etc. to upload an archive of several files.
   b. modify the expiration of the file (number of downloads or number of days), the default is to allow only one download and it expires within one day.
   c. add a password that's needed to download the file.
3. Copy the URL generated for your upload.

⚠️ You need to save that unique URL to get that file later. Unlike Dropbox or Google Drive, Firefox Send will not show you what you uploaded.

b. Downloading on Hydra from Firefox Send using `ffsend`

on Hydra

```
$ module load tools/ffsend
$ ffsend download https://send.vis.ee/download/7800f8272ba5ef7b/#cNSwgMaNqmdsdwG6RxM71A
Download complete
```

Example 2: Sending *from* Hydra

a. Uploading from Hydra using `ffsend`

on Hydra

```
$ module load tools/ffsend
$ ffsend upload test.tar.gz
Upload
complete

https://send.vis.ee/download/0324d02485dc9a02/#cxER28yNyf2dcwzwfIla6g
```

b. Downloading to your local machine (workstation/laptop)

Open the URL created on Hydra in a web browser to download the file to your local machine.

Notes

- while `send.firefox.com` was discontinued, a workaround is to use `--host https://send.vis.ee` (see https://gitlab.com/timvisee/ffsend/-/issues/101.)
- as of version 0.2.65, `ffsend` will use `send.vis.ee` instead of the defunct `send.firefox.com`
- We have modified the module `bioinformatics/ffsend` to include two aliases:
  - `ffupload`, aliased to "`ffsend upload`", and
  - `ffdownload`, aliased to "`ffsend download`".
- We will not change the `ffupload` alias to revert to using `moz://a` since they discontinued the service.

Last Updated 11 Oct 2021 SGK/PBF.

Using rclone

- rclone is utility that can be configured to work with many online file storage systems including Dropbox, Microsoft OneDrive or Google Drive (see the full list here).
- You can use rclone on Hydra with `module load bioinformatics/rclone`
- When authorizing rclone access to your Dropbox, OneDrive or GoogleDrive you can use the rclone AppID created by the program developers or create your own.
    - These instructions are for using the already created rclone App ID.
    - The rclone documentation suggests creating your own app if you experience throttling using the shared App ID.
    - See the rclone documentation links for Dropbox and OneDrive for more information.

💡By default the rclone configuration file is `~/.config/rclone/rclone.conf`

Configuring rclone for use with Dropbox

Full documentation is available here: https://rclone.org/dropbox/

To authorize rclone's access to your Dropbox account, start the configuration on Hydra, then move to a workstation with a graphical interface and finally finish on Hydra.

💡If you need to deauthorize rclone's access, log in to your Dropbox account via a browser, go to Settings and then the Connected apps tab and remove rclone.

Part 1 (on Hydra)

We are naming this connection to dropbox "db". That name will be used in the rclone commands to identify the configuration to be used for the file transfer.

on Hydra

```
$ module load bioinformatics/rclone
$ rclone config
n) New remote
n/s/q> n
name> db
Type of storage to configure.
...
Storage> 9

OAuth Client Id
client_id> [leave blank]
OAuth Client Secret
client_secret> [leave blank]

Edit advanced config? (y/n)
y/n> n
Remote config
Use auto config?
* Say Y if not sure
* Say N if you are working on a remote or headless machine
y/n> n
result>
```

Part 2 (on your workstation)

Leave rclone on Hydra running with the `result>` prompt up.

Download and unzip the pre-compiled rclone version for your workstation OS: https://rclone.org/downloads/

On Mac or Linux systems:

Open a new terminal window and navigate to the directory with the unzipped copy of rclone and enter this command:

on your Mac or Linux system

```
$ ./rclone authorize "dropbox"
```

On Windows workstations:

Open a new Command Prompt and navigate to the rclone directory

on your Windows PC

```
C:\> rclone authorize "dropbox"
```

A browser window will open where you should log in to your Dropbox account and authorize rclone's access to your account.

In the Terminal window, you will get an access token that you should then enter into the `result>` prompt that rclone on Hydra (the text to copy is about 150 characters).

on Hydra

```
result> {"access_token":"...","token_type":"bearer","expiry":"..."}
y) Yes this is OK (default)
y/e/d> y
```

Configuring rclone for use with OneDrive

Like with the Dropbox configuratoin, this is done in two parts, first on Hydra with the `rclone config` command and then on your workstation by downloading and running a local copy of rclone and authorizing rclone's access to your cloud account.

💡 If you need to deauthorize rclone's access, open this page: https://portal.office.com/account Select “App permissions” and then revoke "rclone".

Part 1 (on Hydra)

Follow the steps shown above for the Dropbox setup, but choose the `Microsoft OneDrive` option, `(23)`. and choose a short name for the configuration like `od`.

When you get to the `result>` prompt, continue to the section on your workstation, keeping rclone running on Hydra.

Part 2 (on your workstation)

Follow the steps shown above for the Dropbox setup, but enter `rclone authorize "onedrive"` in your workstation's command prompt or terminal.

Paste the token result into Hydra (the text to copy is about 3500 chracters) :

on Hydra

```
result> {"access_token":"...","token_type":"bearer","expiry":"..."}
y) Yes this is OK (default)
y/e/d> y
Choose a number from below, or type in an existing value
 1 / OneDrive Personal or Business
   \ "onedrive"
...
Your choice> 1

Found 2 drives, please select the one you want to use:
0: OneDrive (business) id=...
1: OneDrive (business) id=...
Chose drive to use:> 0
Found drive 'root' of type 'business', URL: https://sinet-my.sharepoint.com/personal/USERNAME/Documents
Is that okay?
y/n> y
```

⚠️ The URL returned at the choice of drive should end with `Documents`. If it ends with `PreservationHoldLibrary` , enter `n` and restart the rclone configuration choosing the other option.

Transferring files with rclone

After you configured rclone with a cloud service you can use the rclone commands listed in the online documentation.

Example 1: Copying to or from Hydra with rclone copy

rclone copy

```
# A single file from Hydra to a directory "hydra_backup" on a remote system
rclone copy /path/to/hydra/file db:hydra_backup

# An entire directory from Hydra to 'dest' on remote system
rclone copy /path/to/hydra/dir/ db:hydra_backup/dest

# From the remote directory to your Hydra home directory
rclone db:hydra_backup/file ~
```

⚠ Replace `db` with the name of you used for your rclone configuration

Example 2: Syncing contents of a directory with rclone sync

rclone copy

```
# -i will prompt for each file to be copied or deleted
rclone -i sync /path/to/hydra/dir db:hydra_backup/dest
```

⚠ Replace `db` with the name of you used for your rclone configuration

How to Check Disk & Quota Usage

1. Introduction

The following tools can be used to monitor your disk usage.

- You can use the following Un*x commands:

`du`	show disk use
`df`	show disk free

  or
- you can use Hydra-specific home-grown tools, (these require that you load the `tools/local` or `tools/local+` modules)

`disk-usage`	run `df` and parse its output in a more user friendly format
`dus-report`	run `du` and parse its output in a more user friendly format

- You can also view the disk status at the cluster status web pages, either
  - here (at cfa.harvard.edu)
    or
  - here (at si.edu).

Each site shows the disk usage and a quota report, under the "Disk & Quota" tab, compiled 4x a day respectively, and has links to plots of disk usage vs time.

Last Updated 08 Oct 2021 SGK/PBF.

Disk Usage

1. Commands `du, df`

The output of `du` can be very long and confusing. It is best used with the option "`-hs`" to show the sum ("`-s`") and to print it in a human readable format ("`-h`").

⚠ If there is a lot of files/directory, `du` can take a while to complete.

💡 For example:

```
% du -sh dir/
136M    dir/
```

The output of `df` can be very long and confusing.

💡 You can use it to query a specific partition and get the output in a human readable format ("`-h`"), for example:

```
% df -h /pool/public
Filesystem                     Size  Used Avail Use% Mounted on
10.61.83.219:/vol_pool_public  190T   83T  108T  44% /pool/public

or

% df -h --output=source,fstype,size,used,avail,pcent,file /scratch/genomics
Filesystem     Type  Size  Used Avail Use% File
gpfs01         gpfs  400T   95T  306T  24% /scratch/genomics
```

2. Local Tools

a. Command `disk-usage`

The tool `disk-usage` runs `df` and presents its output in a more friendly format:

```
% disk-usage -d all+
Filesystem                                Size     Used    Avail Capacity  Mounted on
netapp-fas83:/vol_home                  18.00T   13.73T    4.27T  77%/11%  /home
netapp-fas83-n01:/vol_data_public      104.50T   32.41T   72.09T  32%/4%   /data/public
netapp-fas83-n02:/vol_pool_public      190.00T   82.79T  107.21T  44%/1%   /pool/public
gpfs01:public                          400.00T  241.83T  158.17T  61%/39%  /scratch/public
netapp-fas83-n01:/vol_pool_kozakk       11.00T    9.72T    1.28T  89%/1%   /pool/kozakk
netapp-fas83-n02:/vol_pool_nmnh_ggi     21.00T   13.80T    7.20T  66%/1%   /pool/nmnh_ggi
netapp-fas83-n01:/vol_pool_sao_access   19.95T    5.47T   14.48T  28%/2%   /pool/sao_access
netapp-fas83-n01:/vol_pool_sao_rtdc     10.45T  914.45G    9.56T   9%/1%   /pool/sao_rtdc
netapp-fas83-n01:/vol_pool_sylvain      30.00T   23.61T    6.39T  79%/6%   /pool/sylvain
gpfs01:nmnh_bradys                      25.00T   17.34T    7.66T  70%/21%  /scratch/bradys
gpfs01:nmnh_kistlerl                   120.00T   77.36T   42.64T  65%/5%   /scratch/kistlerl
gpfs01:nmnh_meyerc                      25.00T    7.79T   17.21T  32%/1%   /scratch/meyerc
gpfs01:nmnh_quattrinia                  50.00T   39.73T   10.27T  80%/35%  /scratch/nmnh_corals
gpfs01:nmnh_ggi                         77.00T   19.85T   57.15T  26%/4%   /scratch/nmnh_ggi
gpfs01:nmnh_lab                         25.00T    9.78T   15.22T  40%/2%   /scratch/nmnh_lab
gpfs01:nmnh_mammals                     25.00T   10.98T   14.02T  44%/15%  /scratch/nmnh_mammals
gpfs01:nmnh_mdbc                        50.00T    9.16T   40.84T  19%/1%   /scratch/nmnh_mdbc
gpfs01:nzp_ccg                          45.00T   34.56T   10.44T  77%/2%   /scratch/nzp_ccg
gpfs01:sao_atmos                       350.00T  279.68T   70.32T  80%/4%   /scratch/sao_atmos
gpfs01:sao_cga                          25.00T    8.97T   16.03T  36%/6%   /scratch/sao_cga
gpfs01:sao_tess                         50.00T   24.82T   25.18T  50%/83%  /scratch/sao_tess
gpfs01:scbi_gis                         80.00T   33.39T   46.61T  42%/35%  /scratch/scbi_gis
gpfs01:nmnh_schultzt                    25.00T   15.36T    9.64T  62%/26%  /scratch/schultzt
gpfs01:serc_cdelab                      15.00T    5.68T    9.32T  38%/4%   /scratch/serc_cdelab
gpfs01:stri_ap                          25.00T   18.23T    6.77T  73%/1%   /scratch/stri_ap
gpfs01:sao_sylvain                      70.00T   44.09T   25.91T  63%/48%  /scratch/sylvain
gpfs01:usda_sel                         25.00T    5.68T   19.32T  23%/11%  /scratch/usda_sel
gpfs01:wrbu                             40.00T   38.62T    1.38T  97%/6%   /scratch/wrbu
netapp-fas83-n02:/vol_data_admin         1.90T   29.83G    1.87T   2%/1%   /data/admin
netapp-fas83-n01:/vol_pool_admin        28.50T   17.88T   10.62T  63%/1%   /pool/admin
gpfs01:admin                            20.00T    3.20T   16.80T  17%/32%  /scratch/admin
gpfs01:bioinformatics_dbs               10.00T    4.04T    5.96T  41%/1%   /scratch/dbs
gpfs01:tmp                             100.00T   38.33T   61.67T  39%/9%   /scratch/tmp
gpfs01:ocio_dpo                         10.00T    2.66T    7.34T  27%/2%   /scratch/ocio_dpo
gpfs01:ocio_ids                          5.00T    0.00G    5.00T   0%/1%   /scratch/ocio_ids
nas1:/mnt/pool/admin                    20.00T    6.69T   13.31T  34%/1%   /store/admin
nas1:/mnt/pool/public                  175.00T   78.87T   96.13T  46%/1%   /store/public
nas1:/mnt/pool/nmnh_bradys              40.00T    6.51T   33.49T  17%/1%   /store/bradys
nas2:/mnt/pool/n1p3/nmnh_ggi            90.00T   36.28T   53.72T  41%/1%   /store/nmnh_ggi
nas1:/mnt/pool/nzp_ccg                  98.23T   51.02T   47.22T  52%/1%   /store/nzp_ccg
nas2:/mnt/pool/n1p2/ocio_dpo            50.00T   15.65T   34.35T  32%/1%   /store/ocio_dpo
nas2:/mnt/pool/n1p1/sao_atmos          500.00T  365.18T  134.82T  74%/1%   /store/sao_atmos
nas2:/mnt/pool/n1p2/nmnh_schultzt       40.00T   24.06T   15.94T  61%/1%   /store/schultzt
nas1:/mnt/pool/sao_sylvain              50.00T    8.41T   41.59T  17%/1%   /store/sylvain
nas1:/mnt/pool/wrbu                     40.00T    7.89T   32.11T  20%/1%   /store/wrbu
qnas:/hydra                             45.47T   28.83T   16.65T  64%/64%  /qnas/hydra
qnas:/nfs-mesa-nanozoomer              263.75T  214.77T   48.98T  82%/82%  /qnas/mesa
qnas:/sil                             3840.36T 2546.17T 1294.20T  67%/67%  /qnas/sil
```

Use

```
% disk-usage -help
```

to see how else to use it.

You can, for instance, get the disk quotas and the max size, for all the disks, including `/store`, with:

```
% disk-usage -d all+ -quotas
                                                                   quotas:  disk space   #inodes
Filesystem                                Size     Used    Avail Capacity    soft/hard   soft/hard Mounted on
netapp-fas83:/vol_home                  18.00T   13.73T    4.27T  77%/11%    480G/512G   9.0M/10M  /home
netapp-fas83-n01:/vol_data_public      104.50T   32.41T   72.09T  32%/4%     4.3T/4.5T   9.5M/10M  /data/public
netapp-fas83-n02:/vol_pool_public      190.00T   82.79T  107.21T  44%/1%     7.1T/7.5T    17M/18M  /pool/public
gpfs01:public                          400.00T  241.83T  158.17T  61%/39%     14T/15T     37M/38M  /scratch/public
netapp-fas83-n01:/vol_pool_kozakk       11.00T    9.72T    1.28T  89%/1%    10.5T/11.0T   26M/28M  /pool/kozakk
netapp-fas83-n02:/vol_pool_nmnh_ggi     21.00T   13.80T    7.20T  66%/1%    15.0T/15.8T   37M/39M  /pool/nmnh_ggi
netapp-fas83-n01:/vol_pool_sao_access   19.95T    5.47T   14.48T  28%/2%    15.3T/16.0T   37M/39M  /pool
/sao_access
netapp-fas83-n01:/vol_pool_sao_rtdc     10.45T  914.45G    9.56T   9%/1%     1.4T/1.5T   1.8M/2.0M /pool/sao_rtdc
netapp-fas83-n01:/vol_pool_sylvain      30.00T   23.61T    6.39T  79%/6%      29T/30.0T   71M/75M  /pool/sylvain
gpfs01:nmnh_bradys                      25.00T   17.34T    7.66T  70%/21%   25.0T/25.0T  100M/105M /scratch/bradys
gpfs01:nmnh_kistlerl                   120.00T   77.36T   42.64T  65%/5%      98T/100T   255M/260M /scratch
/kistlerl
gpfs01:nmnh_meyerc                      25.00T    7.79T   17.21T  32%/1%    24.0T/25.0T   61M/63M  /scratch/meyerc
gpfs01:nmnh_quattrinia                  50.00T   39.73T   10.27T  80%/35%   18.0T/20.0T   52M/55M  /scratch
/nmnh_corals
gpfs01:nmnh_ggi                         77.00T   19.85T   57.15T  26%/4%    75.0T/77.0T  195M/200M /scratch
/nmnh_ggi
gpfs01:nmnh_lab                         25.00T    9.78T   15.22T  40%/2%     4.0T/5.0T    10M/12M  /scratch
/nmnh_lab
gpfs01:nmnh_mammals                     25.00T   10.98T   14.02T  44%/15%   24.0T/25.0T   66M/68M  /scratch
/nmnh_mammals
gpfs01:nmnh_mdbc                        50.00T    9.16T   40.84T  19%/1%    45.0T/50.0T   45M/50M  /scratch
/nmnh_mdbc
gpfs01:nzp_ccg                          45.00T   34.56T   10.44T  77%/2%    34.0T/35.0T   89M/92M  /scratch
/nzp_ccg
gpfs01:sao_atmos                       350.00T  279.68T   70.32T  80%/4%     245T/250T   637M/650M /scratch
/sao_atmos
gpfs01:sao_cga                          25.00T    8.97T   16.03T  36%/6%     7.0T/8.0T    19M/21M  /scratch
/sao_cga
gpfs01:sao_tess                         50.00T   24.82T   25.18T  50%/83%   36.0T/40.0T   94M/210  /scratch
/sao_tess
gpfs01:scbi_gis                         80.00T   33.39T   46.61T  42%/35%     72T/80T     25M/28M  /scratch
/scbi_gis
gpfs01:nmnh_schultzt                    25.00T   15.36T    9.64T  62%/26%   25.0T/25.0T  100M/105M /scratch
/schultzt
gpfs01:serc_cdelab                      15.00T    5.68T    9.32T  38%/4%    14.0T/15.0T   58M:60M  /scratch
/serc_cdelab
gpfs01:stri_ap                          25.00T   18.23T    6.77T  73%/1%     4.0T/5.0T    10M/12M  /scratch
/stri_ap
gpfs01:sao_sylvain                      70.00T   44.09T   25.91T  63%/48%   68.0T/70.0T  176M/182  /scratch
/sylvain
gpfs01:usda_sel                         25.00T    5.68T   19.32T  23%/11%   24.0T/25.0T   61M/63M  /scratch
/usda_sel
gpfs01:wrbu                             40.00T   38.62T    1.38T  97%/6%    38.0T/40.0T  100M/101M /scratch/wrbu
netapp-fas83-n02:/vol_data_admin         1.90T   29.83G    1.87T   2%/1%     1.4T/1.5T   1.8M/2.0M /data/admin
netapp-fas83-n01:/vol_pool_admin        28.50T   17.88T   10.62T  63%/1%     6.3T/7.0T    14M/15M  /pool/admin
gpfs01:admin                            20.00T    3.20T   16.80T  17%/32%        -           -     /scratch/admin
gpfs01:bioinformatics_dbs               10.00T    4.04T    5.96T  41%/1%         -           -     /scratch/dbs
gpfs01:tmp                             100.00T   38.33T   61.67T  39%/9%    24.0T/25.0T   47M/52M  /scratch/tmp
gpfs01:ocio_dpo                         10.00T    2.66T    7.34T  27%/2%         -           -     /scratch
/ocio_dpo
gpfs01:ocio_ids                          5.00T    0.00G    5.00T   0%/1%         -           -     /scratch
/ocio_ids
qnas:/hydra                             45.47T   28.83T   16.65T  64%/64%        -           -     /qnas/hydra
qnas:/nfs-mesa-nanozoomer              263.75T  214.77T   48.98T  82%/82%        -           -     /qnas/mesa
qnas:/sil                             3840.36T 2546.17T 1294.19T  67%/67%        -           -     /qnas/sil
nas1:/mnt/pool/admin                    20.00T    6.69T   13.31T  34%/1%         -           -     /store/admin
nas1:/mnt/pool/public                  175.00T   78.87T   96.13T  46%/1%         -           -     /store/public
nas1:/mnt/pool/nmnh_bradys              40.00T    6.51T   33.49T  17%/1%         -           -     /store/bradys
nas2:/mnt/pool/n1p3/nmnh_ggi            90.00T   36.28T   53.72T  41%/1%         -           -     /store/nmnh_ggi
nas1:/mnt/pool/nzp_ccg                  98.23T   51.02T   47.22T  52%/1%         -           -     /store/nzp_ccg
nas2:/mnt/pool/n1p2/ocio_dpo            50.00T   15.65T   34.35T  32%/1%         -           -     /store/ocio_dpo
nas2:/mnt/pool/n1p1/sao_atmos          500.00T  365.18T  134.82T  74%/1%         -           -     /store
/sao_atmos
nas2:/mnt/pool/n1p2/nmnh_schultzt       40.00T   24.06T   15.94T  61%/1%         -           -     /store/schultzt
nas1:/mnt/pool/sao_sylvain              50.00T    8.41T   41.59T  17%/1%         -           -     /store/sylvain
nas1:/mnt/pool/wrbu                     40.00T    7.89T   32.11T  20%/1%         -           -     /store/wrbu
```

b. Command `dus-report`

You can compile the output of `du` into a more useful report with the `dus-report` tool. This tool will run `du` for you (can take a while) and parse its output to produce a more concise/useful report.

For example, to see the directories that hold the most stuff in `/pool/sao/hpc`:

```
% dus-report /pool/sao/hpc
 612.372 GB            /pool/sao/hpc
                       capac.   20.000 TB (75% full), avail.    5.088 TB
 447.026 GB  73.00 %   /pool/sao/hpc/rtdc
 308.076 GB  50.31 %   /pool/sao/hpc/rtdc/v4.4.0
 138.950 GB  22.69 %   /pool/sao/hpc/rtdc/vX
 137.051 GB  22.38 %   /pool/sao/hpc/rtdc/vX/M100-test-oob-2
 120.198 GB  19.63 %   /pool/sao/hpc/rtdc/v4.4.0/test2
 120.198 GB  19.63 %   /pool/sao/hpc/rtdc/v4.4.0/test2-2-9
  83.229 GB  13.59 %   /pool/sao/hpc/c7
  83.229 GB  13.59 %   /pool/sao/hpc/c7/hpc
  65.280 GB  10.66 %   /pool/sao/hpc/sw
  64.235 GB  10.49 %   /pool/sao/hpc/rtdc/v4.4.0/test1
  49.594 GB   8.10 %   /pool/sao/hpc/sw/intel-cluster-studio
  46.851 GB   7.65 %   /pool/sao/hpc/rtdc/vX/M100-test-oob-2/X54.ms
  46.851 GB   7.65 %   /pool/sao/hpc/rtdc/vX/M100-test-oob-2/X54.ms/SUBMSS
  43.047 GB   7.03 %   /pool/sao/hpc/rtdc/vX/M100-test-oob-2/X220.ms
  43.047 GB   7.03 %   /pool/sao/hpc/rtdc/vX/M100-test-oob-2/X220.ms/SUBMSS
  42.261 GB   6.90 %   /pool/sao/hpc/c7/hpc/sw
  36.409 GB   5.95 %   /pool/sao/hpc/c7/hpc/tests
  30.965 GB   5.06 %   /pool/sao/hpc/c7/hpc/sw/intel-cluster-studio
  23.576 GB   3.85 %   /pool/sao/hpc/rtdc/v4.4.0/test2/X54.ms
  23.576 GB   3.85 %   /pool/sao/hpc/rtdc/v4.4.0/test2-2-9/X54.ms
  23.576 GB   3.85 %   /pool/sao/hpc/rtdc/v4.4.0/test2/X54.ms/SUBMSS
  23.576 GB   3.85 %   /pool/sao/hpc/rtdc/v4.4.0/test2-2-9/X54.ms/SUBMSS
  22.931 GB   3.74 %   /pool/sao/hpc/rtdc/v4.4.0/test2/X220.ms
  22.931 GB   3.74 %   /pool/sao/hpc/rtdc/v4.4.0/test2-2-9/X220.ms
report in /tmp/dus.pool.sao.hpc.hpc
```

You can rerun `dus-report` with different options on the same intermediate file, like

```
% dus-report -n 999 -pc 1 /tmp/dus.pool.sao.hpc.hpc
```

to get a different report, to see the list down to 1%. Use

```
% dus-report -help
```

to see how else you can use it.

Last Updated 15 May 2024 SGK/PBF

Quota Usage

Quota

The Linux command `quota` is working with the NetApp (`/home`, `/data` & `/pool`), but not on the GPFS (`/scratch`) or the NAS (`/store`).

For example:

% quota -s

```
Disk quotas for user hpc (uid 7235):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
10.61.10.1:/vol_home
                  2203M  51200M    100G           46433   1800k   2000k
10.61.10.1:/vol_sao
                  1499G   1946G   2048G           1420k   4000k   5000k
10.61.10.1:/vol_scratch/genomics
                 48501M   2048G   4096G            1263   9000k  10000k
10.61.200.5:/vol/a2v1/genomics01
                   108M  14336G  15360G             613  10000k  12000k
10.61.10.1:/vol_home/hydra-2/dingdj
                  2203M  51200M    100G           46433   1800k   2000k
```

reports your quotas. The `-s` stands for `--human-readable`, hence the 'k' and 'G'. While

% quota -q

will print only information on filesystems where your usage is over the quota. (`man quota`)

💡The command `quota+` (need to load `tools/local`) return disk quota for all the disks (see the quota+ section in Additional Tool).

Other Tools

Hydra-specific tools, (i.e., requires that you load the `tools/local` module), to help manage quotas are:

- `quota+` - show quota values
- `parse-disk-quota-reports` - parse quota reports

Note: we compile a quota report 4x/day and provide tools to parse the quota report.

- The daily quota report is written around 3:00, 9:00, 15:00, and 21:00
  - in a file called `quota_report_YYDDMM_HH.txt, located in /data/sao/hpc/quota-reports/unified/`.
- The string `YYDDMM_HH` corresponds to the date & hour of the report: "`160120_09`" for Jan 20 2016 9am report.
- The format of this file is not very user friendly and users are listed by their user ID.

Examples

- `quota+` - show quota values:

```
% quota+
Disk quotas for user sylvain (uid 10541):
Muonted on                             Used   Quota   Limit   Grace   Files   Quota   Limit   Grace
----------                          ------- ------- ------- ------- ------- ------- ------- -------
/home                                11.00G  50.00G  100.0G       0  73.13k   2.00M   2.00M       0
/data/sao                             1.92T   7.60T   8.00T       0  37.53M  78.00M  80.00M       0
/pool/sylvain                         8.79T  12.50T  14.00T       0  57.93M  71.00M  75.00M       0
/scratch/sao                         10.00G  11.00T  12.00T       0       2  25.17M  26.21M       0
/scratch/sylvain                      6.63T  50.00T  50.00T       0   1.89M  99.61M  104.9M       0
/store/admin                          1.00G    none    none
/store/sylvain                        8.39T    none    none
```

Use `quota+ -h,` or read the man page (`man quota+`), for the complete usage info.

- `parse-disk-quota-reports` will parse the disk quota report file and produce a more concise report:

```
% parse-disk-quota-reports
Disk quota report: show usage above 85% of quota, (warning when quota > 95%), as of Wed Nov 20 21:00:05 2019.

Volume=NetApp:vol_data_genomics, mounted as /data/genomics
                     --  disk   --     --  #files --     default quota: 512.0GB/1.25M
Disk                 usage   %quota    usage  %quota     name, affiliation - username (indiv. quota)
-------------------- ------- ------    ------ ------     -------------------------------------------
/data/genomics       512.0GB 100.0%     0.17M  13.4% *** Paul Frandsen, OCIO - frandsenp

Volume=NetApp:vol_data_sao, mounted as /data/admin or /data/nasm or /data/sao
                     --  disk   --     --  #files --     default quota:  2.00TB/5M
Disk                 usage   %quota    usage  %quota     name, affiliation - username (indiv. quota)
-------------------- ------- ------    ------ ------     -------------------------------------------
/data/admin:nasm:sao  1.88TB  94.0%     0.01M   0.1%     uid=11599

Volume=NetApp:vol_home, mounted as /home
                     --  disk   --     --  #files --     default quota: 100.0GB/2M
Disk                 usage   %quota    usage  %quota     name, affiliation - username (indiv. quota)
-------------------- ------- ------    ------ ------     -------------------------------------------
/home                 96.5GB  96.5%     0.41M  20.4% *** Roman Kochanov, SAO/AMP - rkochanov
/home                 96.3GB  96.3%     0.12M   6.2% *** Sofia Moschou, SAO/HEA - smoschou
/home                 95.2GB  95.2%     0.11M   5.6% *** Cheryl Lewis Ames, NMNH/IZ - amesc
/home                 95.2GB  95.2%     0.26M  12.8% *** Yanjun (George) Zhou, SAO/SSP - yjzhou
/home                 92.2GB  92.2%     0.80M  40.1%     Taylor Hains, NMNH/VZ - hainst

Volume=NetApp:vol_pool_genomics, mounted as /pool/genomics
                     --  disk   --     --  #files --     default quota:  2.00TB/5M
Disk                 usage   %quota    usage  %quota     name, affiliation - username (indiv. quota)
-------------------- ------- ------    ------ ------     -------------------------------------------
/pool/genomics        1.71TB  85.5%     1.23M  24.6%     Vanessa Gonzalez, NMNH/LAB - gonzalezv
/pool/genomics        1.70TB  85.0%     1.89M  37.8%     Ying Meng, NMNH - mengy
/pool/genomics        1.45TB  72.5%     4.56M  91.3%     Brett Gonzalez, NMNH - gonzalezb
/pool/genomics       133.9GB   6.5%     4.56M  91.2%     Sarah Lemer, NMNH - lemers

Volume=NetApp:vol_pool_kistlerl, mounted as /pool/kistlerl
                     --  disk   --     --  #files --     default quota: 21.00TB/52M
Disk                 usage   %quota    usage  %quota     name, affiliation - username (indiv. quota)
-------------------- ------- ------    ------ ------     -------------------------------------------
/pool/kistlerl       18.35TB  87.4%     0.88M   1.7%     Logan Kistler, NMNH/Anthropology - kistlerl

Volume=NetApp:vol_pool_nmnh_ggi, mounted as /pool/nmnh_ggi
                     --  disk   --     --  #files --     default quota: 15.75TB/39M
Disk                 usage   %quota    usage  %quota     name, affiliation - username (indiv. quota)
-------------------- ------- ------    ------ ------     -------------------------------------------
/pool/nmnh_ggi       14.78TB  93.8%     8.31M  21.3%     Vanessa Gonzalez, NMNH/LAB - gonzalezv

Volume=NetApp:vol_pool_sao, mounted as /pool/nasm or /pool/sao
                     --  disk   --     --  #files --     default quota:  2.00TB/5M
Disk                 usage   %quota    usage  %quota     name, affiliation - username (indiv. quota)
-------------------- ------- ------    ------ ------     -------------------------------------------
/pool/nasm:sao        1.78TB  89.0%     0.16M   3.2%     Guo-Xin Chen, SAO/SSP-AMP - gchen
```

reports disk usage when it is above 85% of the quota.

> Use `parse-disk-quota-reports -h`, or read the man page (`man parse-disk-quota-reports`). for the complete usage info.

Note

- Users whose quotas are above the 85% threshold will receive a warning email one a week (issued on Monday mornings).
    - This is a warning, as long as you are below 100% you are OK.
    - Users won't be able to write on disks on which they have exceeded their hard limits.

Last Updated 15 May 2024 SGK/PBF

How to Recover Old or Deleted Files using Snapshots

Some of the disks on the NetApp filer and the NAS have the so called "snapshot mechanism" enabled:

- This allow users to recover deleted files or access an older version of a file.
- Indeed, the NetApp filer makes a "snapshot" copy of the file system (the content of the disk) every so often and keeps these snapshots up to a given age.
- So if we enable hourly snapshot and set a two weeks retention, you can recover a file as it was hours ago, days ago or weeks ago, but only up to two weeks ago.
- The drawback of the snapshot is that when files are deleted, the disk space is not freed until the deleted files age-out, like 2 or 4 weeks later.

How to Use the NetApp Snapshots:

To recover an old version or a deleted file, foo.dat, that was (for example) in `/data/genomics/frandsen/important/results/`:

- If the file was deleted:

```
% cd /data/genomics/.snapshot/XXXX/frandsen/important/results
% cp -pi foo.dat /data/genomics/frandsen/important/results/foo.dat
```

- If you want to recover an old version:

```
% cd /data/genomics/.snapshot/XXXX/frandsen/important/results
% cp -pi foo.dat /data/genomics/frandsen/important/results/old-foo.dat
```

- The "`-p`" will preserve the file creation date and the `"-i"` will prevent overwriting an existing file.
- The `"XXXX"` is to be replaced by either:
  - `hourly.YYYY-MM-DD_HHMM`
  - `daily.YYYY-MM-DD_0010`
  - `weekly.YYYY-MM-DD_0015`
    where `YYY-MM-DD` is a date specification (i.e., `2015-11-01`)
- The files under `.snapshot` are read-only:
  - they be recovered using `cp`, `tar` or `rsync`; but
  - they cannot be moved (`mv`) or deleted (`rm`).

How to Use the NAS/ZFS Snapshots:

- The snapshots on the `/store` disks are:
  - located under `/store/XXX/.zfs/snapshot` (where XXX is, for example, `public`) and
  - in sub-directories named `auto-YYMMDD.0230-8w` where YYYYMMDD represent the date of the snapshot.
- Content of NAS/ZFS snapshots can be recovered as described above.

Last Updated 08 Oct 2021 SGK/PBF.

Scrubber and How to Request Scrubbed Files to be Restored

1.Introduction - What is Scrubbing?

In order to maintain free disk space on the public disks, we use a disk scrubber to remove old files and old empty directories.

The scrubber is run on a weekly basis, it deletes old empty directories, but old files are, at first, moved away in a staging location, then *permanent*ly deleted some 10 days later.

Please Note

Since the scrubber moves old files away at first, and delete them later,

- there is a grace period between the scrubbing (move) and the *permanent* deletion to allow users to request for some scrubbed files to be restored;
- reasonable requests to restore scrubbed files must be be sent no later than the Friday following the scrubbing, by 5pm;
- scrubbed files still "count" against the user quota until they are *permanent*ly deleted;
- one *permanent*ly deleted (aka zapped) the files can no longer be restored.

Requests to restore scrubbed file should be

- rare,
- reasonable (*i.e.* no blanket request), and,
- can only be granted while the scrubbed files are not yet *permanently* deleted.

Past the grace period, the files are no longer available, hence users who want their scrubbed files restore have to act *promptly*.

2. What Disks Are Scrubbed

The disks that are scrubbed are:

- `/pool/biology       - 180 days old files/empty directories`
- `/pool/genomics      - 180 days`
- `/pool/sao           - 180 days`
- `/scratch/genomics   - 180 days`
- `/scratch/sao        - 180 days`

3. How to Access the Scrubber's Tools

To access the scrubber tools, you need to load the module:

```
module load tools/scrubber
```

- to get the list of tools, use:

```
module help tools/scrubber
```

- to get the man page, accessible after loading the module, use:

```
man <tool-name>
```

4. How to Check what Will Be Scrubbed

- To check what files will be scrubbed, use:

  ```
  find-scrub [-in <dir>] [-age <age>]
  ```

  this will look for files older than <age> days in <dir>, by default dir=current working directory, and age=173 or 83 days.
- This search taxes the file system (aka disk server), especially if you have a lot of files, so use as needed only.

5. How to Find Out what Was Scrubbed

You will receive an email if any of your files were scrubbed.

- To look at the report for what was scrubbed on Jul 21 2016 under `/pool/genomics/frandsenp`:

  ```
  show-scrubber-report /pool/genomics/frandsenp 160721
  ```

- To find out which old empty directories where scrubbed:

  ```
  list-scrubbed-dirs [-long|-all] /pool/genomics/frandsenp 160721 [<RE>|-n]
  ```

where the <RE> is an optional regular-expression to limit the printout, w/o an RE your get the complete list, unless you specify -n and you get the number of scrubbed directories.

The -long or -all option allows you to get more info (like age, size and owner)

- To find out which old files where scrubbed:

  ```
  list-scrubbed-files [-long|-all] /pool/genomics/frandsenp 160721 [<RE>|-n]
  ```

where again the <RE> is an optional regular-expression to limit the printout, w/o an RE your get the complete list, unless you specify -n and you get the number of scrubbed files;

the `-long` option will produce a list that includes the files' age and size, -all will list age, size and owner.

- 💡 The <RE> (regular expressions) are PERL-style RE:
  - `.` means any char,
  - `.*` means any set of chars,
  - `[a-z]` means any single character between `a` and `z`,
  - `^` means start of match,
  - `$` means end of match, etc (see gory details here).
- for example:

  ```
  '^/pool/genomics/blah/project/.*\.log$'
  ```

means all the files that end in `'.log'` under `'/pool/genomics/blah/project/'`

6. How to Request Scrubbed Files to be Restored

In order to request that some of your scrubbed files be restored, you need to create a list of files, trimmed it down to what you really need and verify that list. We do not accept bulk restore requests.

To produce the list of files to restore (that in this example were under `/pool/genomics/frandsenp/big-project`), follow these 4 steps:

1. Create a list
   ```
   list-scrubbed-files /pool/genomics/frandsenp 160721 /pool/genomics/frandsenp/big-project/ > restore.list
   ```
   this will lists all the scrubbed files under `'big-project/'` and save the list in `restore.list`

   ⚠️ Note that `/pool/genomics/frandsenp/big-project` means `/pool/genomics/frandsenp/big-project*`, not `/pool/genomics/frandsenp/big-project/`

2. edit the file '`restore.list`' to trim it down to what you really need, with any text editor;

3. verify
   ```
   verify-restore-list /pool/genomics/frandsenp  160721 restore.list
   ```
   or use
   ```
   verify-restore-list -d /pool/genomics/frandsenp  160721 restore.list
   ```
   if the verification produced an error.

4. Only then, and if the verification produced no error, submit your scrubbed file restoration request as follow:
   - SAO users: email the location of the list file(s) to Sylvain at hpc@cfa.harvard.edu
   - non-SAO users: email the location of the list file(s) to SI-HPC@si.edu

     You can email the list file(s) themselves, but it is more convenient for us of the files are somewhere on Hydra.

Last Updated 17 Nov 2021 SGK/PBF.

How to Use Local SSD Space

1. Introduction

We have added SSDs, solid state disks, on a set of compute nodes:

- These are local fast disks that can speed up applications that preform a lot of intensive I/O operations,
    - hence such jobs should complete faster when using SSDs.
- Jobs that do not perform intensive I/Os should not use the SSDs - this is a limited shared resource.

- Since these disks are local to the compute nodes:
    - you cannot see the SSD from either login nodes,
    - your job will be able to use the SSD *only while the job is running*, hence
        - you need to prepare the files needed for a job prior to submitting the job, using /pool or /scratch, and
        - request the right amount of SSD disk space:

            - this is the maximum amount of disk space your job will need on the SSD at run-time, (similarly to the maximum of memory it will need).
            - You need to add something like `-l ssd_res=2560G` when submitting a job to request
                1. a compute node with (local) SSD,
                2. a quota of 2560GB (for example) on the SSD.
    - Your job script will have to
        - copy those files to the SSD before processing them,
        - be adjusted to use the SSD,
        - upon completion, copy the results from the SSD elsewhere (/pool or /scratch), and
        - delete what you wrote on the SSD.
    - If you exceed the amount of disk space, i.e. your quota, your job won't be able to write any longer to the SSD,
      hence your job script should stop when encountering such error.
    - If the job completes normally and there is less than 50GB of 'stuff' left on the SSD, that stuff will be archived in a tar-compress set,
        - otherwise the content of the SSD is deleted (including if the job gets killed).

This remains a limited resource, so used it only if your application benefit from using it.

2. How To Use SSDs

Since you can't access the SSDs from a login node, you must prepare the data the job will need somewhere else, like on `/pool` (or on `/scratch`) before submitting a job.

Like for memory, you need to *guestimate* how much SSD space your job will need. You will not be able to use more SSD space than you requested.

💡 Remember, your job will still be able to access the `/home, /data, /pool,` and `/scratch` disks, hence you don't have to copy everything on the SSD,

only the I/O intensive part of the analysis should use the SSD.

How to Prepare my Data

1. Create a subdirectory in `/pool` (or `/scratch)` and move or copy the data you will need, for example (as user smart1)
   ```
   cd /pool/genomics/smart1
   mkdir -p great/project/wild-cat
   ```
   Now is have a directory for this case, and would copy the I/O intensive part of the required data set in it.

2. While not required, you can pack these data in a compressed tar-ball
   ```
   cd /pool/genomics/smart1/great/project/wild-cat
   tar cfz ../wild-cat.tgz .
   ```
   The file `/pool/genomics/smart1/great/project/wild-cat.tgz` now holds you input data set,
   being compressed it is likely to be smaller than the content of `/pool/genomics/smart1/great/project/wild-cat,`
   That directory can be deleted, unless you will need it later.

3. Ancillary data and/or configuration files that are not causing intensive I/O can stay on a location under `/pool` (or `/scratch`)

How to Adjust a Job Script to Use the SSD

Your jobs script will need the following 4 parts

Part 1: Copy the Data to the SSD

- At the top of your job script, load the `tools/ssd` module and copy or extract your data set as follows:

example using recursive copy

```
module load tools/ssd
cp -pR /pool/genomics/smart1/great/project/wild-cat/* $SSD_DIR/.
```

or

example using compressed tar-ball

```
module load tools/ssd
cd $SSD_DIR
tar xf /pool/genomics/smart1/great/project/wild-cat.tgz
```

💡 The advantage of the compressed tar-ball is that the `.tgz` file is likely to be smaller than the content of the directory, hence less I/O transfer from the `/pool` disk, while un-compressing and writing to the SSD is fast,

Part 2: Adjust the Script or a Configuration File

- You need to replace all references to `/pool/genomics/smart1/great/project/wild-cat` by $SSD_DIR,
- this can be easily done at the shell script level, but not in a configuration file, i.e., for flags/options
  `execute -o /pool/genomics/smart1/great/project/wild-cat/result.dat`
  is replaced by
  `execute -o $SSD_DIR/result.dat`

- Here is a simple trick to modify a configuration file:
  - Let's assume that your analysis uses a file `wow.conf`, where for instance the full path of some files must be listed, like:

wow.conf

```
# this is the configuration file of the fabulous WOW package
input=/pool/genomics/smart1/great/project/wild-cat/wiskers.dat
output=/pool/genomics/smart1/great/project/wild-cat/tail.dat
paws=4
eyes=2
```

  - Replace the `wow.conf` file by a `wow.gen` file as follows:

wow.gen

```
# this is the configuration file of the fabulous WOW package
input=XXXX/wiskers.dat
output=XXXX/wild-cat/tail.dat
paws=4
eyes=2
```

  - And create the `wow.conf` file from the `wow.gen` at run-time by adding the following in the job script:

```
sed "s=XXXX=$SSD_DIR=" wow.gen > wow.conf
```

As long as `XXXX` is not used for anything else, this will replace every occurrence of `XXXX` by the value of the environment variable `SSD_DIR.`

Part 3: Run the Analysis

- With your data copied to the SSD and with your commands and configuration files adjusted to use the SSD, run your analysis.

Part 4: Copy the Results from the SSD

- At the end of the job script, you must add instructions to copy the results of your analysis back to `/pool`(or `/scratch`, or `/data`).

  If/when the results are easily identifiable, you can use `the commands` `mv` or `tar`, and `find`, here are a few examples:

1. Move the directory where all the results are stored and the log file, delete the rest.

   moving identifiable results, delete the rest

   ```
   # move results and log file back
   cd $SSD_DIR
   mv results /pool/genomics/smart1/great/project/wild-cat/.
   mv wow.log /pool/genomics/smart1/great/project/wild-cat/.
   #
   # delete the rest
   rm -rf *
   ```

2. Move the directory where all the results are stored and the log file, delete the input (conservative approach, in case you missed something).

   moving identifiable results, delete known input sets

   ```
   # move results and log file back
   cd $SSD_DIR
   mv results /pool/genomics/smart1/great/project/wild-cat/.
   mv wow.log /pool/genomics/smart1/great/project/wild-cat/.
   #
   # delete input set and other stuff
   rm -rf input
   rm wow.gen wow.conf
   ```

3. Move using the `--update` flag of `mv` (see `man mv`)

   moving using --update

   ```
   # move results using --update
   cd $SSD_DIR
   mv --update * /pool/genomics/smart1/great/project/wild-cat/.
   #
   # delete the rest
   rm -rf *
   ```

   Note, you can use `mv --update` on an explicit list (of files, directories, or file specification), not just * (everything), and you do not have to remove the rest, but can only remove what you know you can safely remove (conservative approach).

4. Find newer files and move them: the trick is to create a 'timestamp' file ***before*** starting the analysis.
   That file can be used later to find any newer file with the `--newer=` option of `tar` (see `man tar`):

Using a timestamp file and tar --newer=

```
# set the timestamp
date > $SSD_DIR/started.txt
# run the analysis
...
# copy the new files in the subdir data/ to a compressed tar-ball
cd $SSD_DIR
tar --newer=$SSD_DIR/started.txt -cfz /pool/genomics/smart1/great/project/wild-cat-results.tgz
data/
# now remove it
rm -rf data/
# etc...
# delete everything, unless
rm -rf *
```

See previous comments and what to tar and what to remove: once you've `tar`'d new stuff in `data/`, remove `data/`, etc.

5. Using the timestamp file and the `find` command (see `man find`):

Using find and a timestamp file

```
# set the timestamp
date > $SSD_DIR/started.txt
# run the analysis
...
# find the new files in the subdir data/
cd $SSD_DIR
find data/ -newer $SSD_DIR/started.txt -type f > /tmp/list
# do the same on logs/, append to the list
find logs/ -newer $SSD_DIR/started.txt -type f >> /tmp/list
# etc...
# now save what is in the list with one tar
tar --files-from=/tmp/list -cfz /pool/genomics/smart1/great/project/wild-cat-results.tgz data/
# now remove data/ and logs/
rm -rf data/ logs/
# etc...
# delete everything, unless
rm -rf *
```

✓There are many more ways to accomplish this ....

💡 BTW, the advantage of writing a `.tgz` file, rather than moving files is two fold, assuming your stuff is compressible:

a. You write less in the .tgz file, so it should be done faster (reading and compressing should be fast, writing is the slow step)
b. you need less disk space for your output (since it is compressed).

The drawback being that you need to know how to handle/view/deal with a `.tgz` file.

Controlling Saving the Content of the SSD at the End of the Job

You can control where and how much of the content of the SSD to save at the end of the job to overwrote the default behavior with two variables:

1. `SSD_SAVE_DIR` to specify where to save the content of the SSD when the job finishes, or not to save anything, and
2. `SSD_SAVE_MAX` to specify the max size to save, namely if there is more than the given size left on the SSD, not to save it.

NOTE:

- `SSD_SAVE_DIR` must specify an existing directory in which you can write, or you can set the value to '-' (w/out the quotes) to disable the saving.
- `SSD_SAVE_MAX` must be a number (integer or float) followed by an optional unit - like k,K,M,G or T - for example `SSD_SAVE_MAX=21.4M`.
- Setting `SSD_SAVE_MAX` to 0 is equivalent to setting `SSD_SAVE_DIR` to '-', namely do not save what is left.
- ⚠ Avoid setting `SSD_SAVE_MAX` to a value too large (i.e., > 80G), since saving the content of the SSD will take too long, instead save the contend of SSD in your job script.
- ⚠ If you kill your job, the content of the SSD is *never* saved.

How to Specify these:

These must be passed via the `-v` flag of `qsub`, either explicitly to `qsub` or as an embedded directive in the job file, as in

Explicitly

```
% qsub -v SSD_SAVE_DIR=/pool/sao/hpc/save -v SSD_SAVE_MAX=10M demo.job
```

or

Embedded directive in job file

```
#
#$ -l ssd_res=10G -v SSD_SAVE_DIR=/home/hpc/tmp -v SSD_SAVE_MAX=10M
```

3. Examples

1. **A trivial example is available on Hydra in `~hpc/examples/ssd` as `test-ssd.job` , i.e.:**

Trivial Example

```
#
#$ -cwd -j y -o test-ssd.log -N test-ssd
#$ -l ssd_res=10G
#
echo + `date` $JOB_NAME started on $HOSTNAME in $QUEUE with id=$JOB_ID
echo NSLOTS = $NSLOTS
#
module load tools/ssd
ls -ld $SSD_DIR
ls -l  $SSD_DIR
#
date >  $SSD_DIR/date
dd if=/dev/zero of=$SSD_DIR/100M count=1024 bs=102400
#
ls -lh $SSD_DIR/*
#
echo = `date` $JOB_NAME done.
```

2. Here is what a more sophisticated job script might look like:

Pseudo Example

```
#
#$ -N example
#$ -o example.log -cwd -j y
#$ -l ssd_res=2560G
#
# pseudo example using a fake package WOW, on the SSD
#
echo $JOB_NAME started `date` on $HOSTNAME in $QUEUE jobID=$JOB_ID
#
module load tools/ssd
module load special/wow
#
# create a wow config file from a generic version, to insert the SSD temp dir value
sed "s=XXXX=$SSD_DIR=" ~/wow/wild-cat.gen > ~/wow/wild-cat.conf
#
# cd to the SSD temp dir and copy the data set to it, using the existing .tgz file
cd $SSD_DIR
tar xf /pool/genomics/smart1/great/project/wild-cat.tgz
#
# create some sub dirs for output and logs
mkdir output
mkdir logs
#
# run the wow analysis (note how some files are not on the SSD)
wow --type=m --params=$HOME/wow/parameters.dat --config=$HOME/wow/wild-cat.conf -o $SSD_DIR/output -l $SSD_DIR
/logs
#
# save the output and the logs in a tar compressed file
# (assumes wow did not change current working directory)
# otherwise insert: cd $SSD_DIR
tar -cfz /pool/genomics/smart1/great/project/wild-cat-results.tgz output/ logs/
#
# remove everything (in $SSD_DIR), or remove what you know you can (conservative option)
rm -rf *
#
echo $JOB_NAME done `date`
```

4. SSD Usage Monitoring Tools

- We have two tools to monitor SSD usage, one on a per-job basis, and one to view usage summary.
- To access them, you to load the `tools/local` module.

Per Job Basis

plot-qssduse.pl

```
% module load tools/local
% plot-qssduse -x 7420073
```

This example plots the SSD usage of job 7420073 to the screen, assuming you have an X-windows capable connection,

- drop the `-x` to plot to a file,
- and use `NNNN.TTT`, instead of `NNNN` to show usage for a given task (`TTT`) of a job array (`NNNN`).
- Try `plot-qssduse -help` or `man plot-qssduse` for more information.

Usage Summary

plot-qssduse-summary.pl

```
% module load tools/local
% plot-qssduse-summary -x
```

As above:

- drop the `-x` to plot to a file.
- Try `plot-qssduse-summary -help` or `man plot-qssduse-summary` for more information.

Last Updated 18 Jan 2024 SGK

How to Use NAS Storage and the I/O Queue

Near Line Storage or NAS

- A large near-line storage of about 950TB has been added to Hydra in July 2019 as NAS and mounted via NFS.
- This storage is mounted as `/store`, *but only* on both login nodes. the head node and the interactive nodes.
- It is not (*and will not be*) mounted to the rest of the compute nodes.
    - Hence data stored on the NAS are to be copied to active storage (e.g. `/data`, `/pool` or `/scratch`) before processing and/or analyzing your data, and vice-versa (active storage can be offloaded to `/store`).
- Most of that disk space is project-specific space, but there is a small amount of space on that system available to all other users upon request and at no cost.
- Approved users will receive up to 5TB of un-scrubbed space, with a daily snapshot up to 14 days.
    - Note that we reserve the right to clean up old stuff that is likely to accumulate in the future, with proper notification, once it fills up.

SAO users interested in receiving an allocation in `/store/public` should contact Sylvain (hpc@cfa.harvard.edu) and Biology users should contact Rebecca (DikowR@si.edu).

Additional Technical Details

- This NAS is storage is build on a zfs file system and runs FreeNAS (a dedicated Linux version for NAS/NFS).
- It supports quotas and snapshots, although not the full gamut that NetApp and GPFS offers.
- This system is fault tolerant (when a disk fails it keeps running fine) but not high availability (i.e. there is no full hardware redundancy), so it is intrinsically less robust (hence less expensive) than the NetApp and the soon to arrive GPFS.
- Like all the disks on Hydra, its content is NOT backed up, and while we do not expect catastrophic failures, keep all this in mind.
- It is to be thought of as a "*cheap bucket*" to keep things around awaiting processing, not for backup, archiving or any other form of reliable & long term storage.

Also, the Linux command `quota` does work with the NAS/zfs, instead use `quota+.pl` (part of the `tools/local` module).

Accessing `/store`

- You can access `/store` from either login node. You can also access via the interactive queue, using `qrsh`.
    - Remember that the limits on login node usage and on `qrsh` jobs are in effect.
- Alternatively, you can also access `/store` via the IO queue with its specific limits.
- Submitting jobs to the IO queue allows you to (1) have larger limits, (2) queue a slew of IO jobs and (3) chain IO and processing jobs using the scheduler.

The IO Queue

- We have added an IO queue, called `lTIO.q`, so users can do copy data to/from the NAS through batch jobs (`qsub`).
- The IO queue runs on the interactive compute nodes, using a limited number of slots.

To submit an IO job, specify `-q lTIO.q -l ioq` as arguments to `qsub` or as an embedded directive in the job file.

Here is a trivial IO job file:

```
#
#$ -cwd -j y -N testIO
#$ -o testIO.log
#$ -q lTIO.sq -l ioq
#
echo + `date` $JOB_NAME running on $HOSTNAME in $QUEUE with jobID=$JOB_ID
set echo
ls -ld /store/sylvain
ls -l /store/sylvain/*
df -h /store/sylvain
unset echo
echo = `date` $JOB_NAME done
```

Limits

- Jobs in the IO queue can run for 72 hours, and are limited to consuming no more than 12h of CPU and 2GB of memory (per slot).
    - hence IO jobs are not meant to be used to run computations on data stored on the NAS
- Users can run only 2 IO jobs concurrently and use up to 6 slots.
- You can submit as many IO jobs as needed, keeping in mind the total limit of 2,000 running and queued jobs per user.

Hints

- You can tell the scheduler to chain jobs to run sequentially, using the `-hold_jid NNN` argument to `qsub`, where NNN is a job number, the number of the job to wait for completion).
- Here is a conceptual example to chain an analysis job to start only after an IO job completes, and then run a save and clean job.

```
% qsub getData.job
Your job 7437744 ("getData") has been submitted
% qsub -hold_jid 7437744 analyze.job
Your job 7437745 ("analyze") has been submitted
% qsub -hold_jid 7437745 saveNClean.job
Your job 7437746 ("saveNClean") has been submitted
```

- and `qstat+.pl +a%` returns:

```
Total running (PEs/jobs) = 1/1, 2 queued (jobs) for user 'hpc'.
   jobID name                     stat     age nPEs      cpu% queue     node taskID
 7437744 getData                     r   00:01    1           lTIO.sq  8-31
 7437745 analyze                   hqw   00:00    1           sThC.q
 7437746 saveNClean                hqw   00:00    1           lTIO.sq
```

where `hqw` indicates a wait in the queue on a hold.

- Or you can use `qchain` to do this for you:

```
% qchain getData.job  analyze.job saveNClean.job
qsub getData.job
Your job 7437747 ("getData") has been submitted
qsub -hold_jid 7437747 analyze.job
Your job 7437748 ("analyze") has been submitted
qsub -hold_jid 7437748 saveNClean.job
Your job 7437749 ("saveNClean") has been submitted
```

- resulting as above in:

```
% q+ +a%
Total running (PEs/jobs) = 1/1, 2 queued (jobs) for user 'hpc'.
   jobID name                     stat     age nPEs      cpu% queue     node taskID
 7437747 getData                     r   00:00    1          lTIO.sq   8-32
 7437748 analyze                   hqw   00:00    1          sThC.q
 7437749 saveNClean                hqw   00:00    1          lTIO.sq
```

- Use `man` qchain for more info.

Last updated 04 Jul 2019 SGK

How to Use "bigtmp" - Access to Large Temporary Disk Space

Introduction

- We have set aside some temporary disk space on the GPFS for jobs who need large temporary disk space, that will not fit on `/tmp` (aka `bigtmp`).
- To manage that space we've implemented a consumable so the job scheduler will not start more jobs requesting space than there is.
- A user can't request more than 25G at a time, in either a single job or as the total of all the job from that user requesting `bigtmp` space.

How To

- To request large temporary disk space, use `-l bigtmp=XX`, where XX is the amount of disk space needed in GB, like in `-l bigtmp=10` (no unit).
- Use the module `tools/bigtmp` to store the location of the temporary disk space in the `BIGTMP` environment variable.
- Use `$BIGTMP` to specify the temporary disk space location.
- The content of the temporary disk space location is deleted when the job is done.

Example

Let's assume that `doMyThing` is some application that needs lots of temporary space, and whose location cab be specified with the `-tmp` flag:

```
# /bin/csh
#$ -cwd -j y -o test-bigtmp.log -N test-bigtmp
#$ -l bigtmp=10
#
echo + `date` $JOB_NAME started on $HOSTNAME in $QUEUE with id=$JOB_ID
#
module load tools/bigtmp
#
doMyThing -tmp $BIGTMP
#
echo = `date` $JOB_NAME done.
```

Last Updated 21 Nov 2021 SGK

Software

1. Introduction

Hydra is a Linux cluster running Rocky 8.9, while the software installation is managed using Bright Cluster management (BCM or CM).

Like any Linux machine, your Un*x environment on Hydra can be configured to your liking.

How to configure a Un*x environment is beyond the scope of this set of documentation.

Unix Basic Configuration

A set of configuration files, located in your home directory, sets up your Un*x environment - a default set of such files is provided when a new account is created:

Login shell			
Bash shell (`bash`)	**Bourne shell (`sh`)**	**C-shell (`csh` or `tcsh`)**	**Action**
`.bash_profile`	`.profile`	`.cshrc`	read & executed at startup to configure your environment
`.login`	`.login`	`.login`	read & executed at startup, next, but only by a login shell
`.emacs`	`.emacs`	`.emacs`	configures the `emacs` editor
`.bash_logout`	`.logout`	`.logout`	read & executed when logging out (login shell)

Note

`qsub`'ed job scripts are not started as a login shell, hence unless you fully understand the idiosyncrasies of the bash shell startup rules,

it is recommended that you use the Bourne shell (`sh`) or the C-shell (`csh`), and not the bash shell (`bash`) when submitting jobs.

Module

The command `module` is available on Hydra:

- Instead of editing your `.bash_profile` file (or `.profile` or `.cshrc`) to configure your `PATH` (and `MANPATH` and `LD_LIBRARY_PATH`, etc), use the command `module` (as explained below).

EMail Forwarding

- Email sent on the cluster is delivered to the head node (that you should not use),
- To access these emails (like job notifications), a `~/.forward` file was created with your "canonical/home" email address, i.e.:

```
% echo DoeJ@si.edu > ~/.forward
```

- While you are welcome to edit this file and change the forwarding email:

  - do not delete it, and
  - use an email address you will read.
- Note that by SD931, we must communicate with users via their work email, not their private one
- You "on file" email (used for things like password reset and HPCC-L listserv) will remain you "canonical/home" email (the one that ends in `.edu`)

The plan/project files

- the content of the files `~/.plan` and `~/.project` are displayed by the command `pinky` (`man pinky`);
  - feel free to put relevant/pertinent information in them.
- The command finger is no longer available under Rocky 8:
  - the `tools/local-user` module offers a `finger` replacement.
- Try

  ```
  % pinky -l hpc
  ```

  or

  ```
  % finger hpc
  ```

Last updated 15 May 2024 SGK

The "module" Command

1.Introduction

- The purpose of a module file is to simplify how to configure/modify your Un*x environment to run or have access to some specific set of tools /applications/etc....
- A module file also allows users to change their configuration without worrying what shell is being used (i.e., `bash` or `csh`),
- Module files to access general purpose or supported tools/applications are written and maintained by the HPC support team.

So, instead editing your ~/`.bash_profile`, ~/`.profile` or `~/.cshrc file(s)` to configure your Un*x environment (`PATH, MANPATH,LD_LIBRARY_PATH`, etc), you should use the command `module.`

- For example, to use the GNU compiler version 4.9.2, one simply needs to execute:
  ```
  % module load gcc/4.9.2
  ```
- The command `module` uses module files, files that lists what is needed to be done to your Un*x environment to use a given tool, including the location of the tool.
- A module can be loaded, unloaded, or you can switch to a different version without having to edit and/or source a configuration file.
- Also, if the location of a tool changes, one only has to edit the corresponding module file:
  - configuration changes are transparent to the users and to any script or job that uses that tool via the command `module`.
- There is no need to type long paths that include version numbers.
- Module files also allow to check for conflict: you can't use different versions of the same tool simultaneously.
- The command `module` works the same way whether you use use the `bash` or the `csh` shell,
  - so there is no need to explain what to do for each shell.
- Module files can also be used by `perl, cmake` or `python.`
- Users can augment the module files we offer by writing their own module files (written in `TCL)` .
- To learn about the command module, read the module man page:
  ```
  % man module
  ```

⭐ The key ways to use `module` are:

```
module help                 - show help on the command module itself
module load   XXX           - load the module XXX
module unload XXX           - unload the module XXX
module switch XX/YYY        - switch to module XX/YYY
module list                 - list which modules are currently loaded
module avail                - show which modules are available
module -t avail             - show which modules are available, one per line
module whatis               - show one line information about all the available modules
module whatis ZZZ           - show one line information about the module ZZZ
module help   ZZZ           - show the help information on the module ZZZ
module show   ZZZ           - show what loading the module ZZZ does to your Unix environment.
```

- You can easily find out what software is available by "`grep`'ing" the output of `module`, i.e.:
  ```
  % ( module -t avail ) |& grep bioinformatics
  ```
- You can access the documentation built into a module file with:
  ```
  % module help gcc/4.9.2
  ```
- You can view how a module file will change your Un*x environment with:
  ```
  % module show gcc/4.9.2
  ```

Module shortcut and sticky modules

The command module has been upgraded to version 5.3.1. It works as before but has a few improvements:

- explicit sticky modules: two modules are preloaded and are sticky, i.e. you cannot unload them
- `ml` shortcut:
  - `module list, module load` and `module unload` can be shorten using `ml` as follows:

`ml`	`module list`
`ml tools/ffsend`	`module load tools/ffsend`
`ml -tools/ffsend`	`module unload tools/ffsend`

- customization
  - `module list` is by default more verbose and in color

- this can be customized and what is shown when loading a module can be also customized
- the following modules customize the output of the module command:

`ml module-nocolor`	do not use colors
`ml module-nowarn`	disable some warning
`ml module-simple-format`	simplify the output of module list
`ml module-simple`	load the 3 module above
`ml module-color`	specify a color scheme (red for sticky, green for auto-loaded)
`ml module-verbose`	set module in verbose mode, equiv to using the -v flag

- You can load and unload these module to your liking.

Details about the new version of module can be found in the module man page or the documentation about module here.

Module conflict

⚠ The command `module` implements the concept of conflict: i.e., you cannot load simultaneously `gcc/4.9.1` and `gcc/4.9.2`:

- to change versions use switch, i.e.:
  ```
  % module load gcc/4.9.1
  [do your stuff]
  % module switch gcc/4.9.2
  ```

- to use a different (b/c of conflict) tool, use `unload` first
  ```
  % module load gcc/4.9.1
  [do your stuff that needs gcc ver 4.9.1]
  % module unload gcc/4.9.1
  % module load nvidia/23.9
  [do the stuff that needs NVIDIA ver 23.9]
  ```

Module default and manpath

Module also implements the concept of default value:

- so `module load nvidia` is equivalent to `module load nvidias/23.9` if NVIDIA version 23.9 is set as the default nvidia module.

💡 Loading/unloading module(s) may set your MANPATH variable in such a way as to prevent you from accessing default man pages locations.

Loading the module `tools/manpath` will solve this.

2. Available Module Files

For a current list of all available module files on Hydra, see here (or as plain text here).

⚠ In the *bioinformatics* and *tools* categories, the list of module files does not include every version of each software package installed. For these sections, you will need to run `module avail` on Hydra to see every available version.

3. How to Write your Own Modules Files

Users can write their own module files to configure/modify their Un*x environment by loading or unloading a module.

This page describe how to write your own (private) module(s), with examples.

Last updated 15 May 2024 SGK

How to Write your Own Module File

Rationale

- The purpose of a module file is to simplify how to configure/modify your Unix environment to run or have access to some specific set of tools /applications/etc....
- A module file also allows users to change their configuration without worrying what shell is being used (bash or csh),
- Module files to access general purpose or supported tools/applications are written and maintained by the HPC support team.

This being said:

- Users can create their own module file(s) to facilitate the use of software they have installed themselves (like in their own directories) or to simplify how to configure your environment for a type pf jobs.
- This page explains how to write your own, private, module file.

Introduction

- Module files are simple text files written in `TCL` - the Tool Command Language.
- TCL is a very powerful programming language on its own, but in most cases you do not need to know it, you just need to know a few very simple command,
- simple module files can be created with minimal code, to simply update `PATH` or other environmental variables,
- You can read a complete set of instructions on module file with the command `man modulefile,`

Examples

Here are a few simple examples:

1. To change your `PATH` variable to add `/home/USER/miniconda3/bin`

   ```
   #%Module1.0
   prepend-path PATH /home/USER/miniconda3/bin
   ```

   This example is equivalent to

   `PATH=/home/USER/miniconda3/bin:$PATH` (in `bash` or `sh`)

   or

   `setenv PATH /home/USER/miniconda3/bin:$PATH` (in `csh` or `tcsh`).
2. To change your `LD_LIBRARY_PATH` variable to add `/home/USER/geos/lib`

   ```
   #%Module1.0
   prepend-path LD_LIBRARY_PATH /home/USER/geos/lib
   ```

   The instruction `prepend-path` will add the given path to the beginning of the specified variable.
3. To change both PATH and `LD_LIBRARY_PATH` variables, use an internal variable (for convenience) and set a variable:

   ```
   #%Module1.0
   set base /home/USER/crunch
   prepend-path PATH            $base/bin
   prepend-path LD_LIBRARY_PATH $base/lib
   setenv CRUNCH                $base
   ```

   The variable `$base` is only set and used by the TCL, it will **not** be passed to the shell as `$base`, but in this example passed as `$CRUNCH` by the `setenv` instruction.
4. Useful instructions (see `man modulefiles` for explanations):

   - `append-path/prepend-path`
   - `setenv/unsetenv`
   - `module`
   - `remove-path`
   - `conflict`

⚠ Note that in these examples (and the ones below) `/home/USER` should be replaced by a real directory name.

💡 A module file **must** start with the so-called *magic* first line `#%Module1.0`

⭐ Books have been written about TCL, you can use Google to get more information, or go to https://www.tcl.tk/ to learn everything about TCL (and Tk)

Implementation

Once you have written a module file, you can load it with the command `module`, followed by the full path to the module file, like in:

```
module load /home/USER/modulefiles/miniconda
```

You can make things easier, by storing all your module files in a single location and instructing the command `module` where to look, by either

- modifying the environment variable MODULEPATH to include the location(s) where to look,

or by

- creating a `~/.modulerc` file that adds to the directories where modules are searched for

example of ~/.modulerc

```
#%Module1.0
module use --append /home/USER/modulefiles
```

With this `~/.modulerc` file, you can then load your module file `miniconda` with the command

```
module load miniconda
```

More Examples

- The system-wide modules are located in `/share/apps/modulefiles`
- You can take a look there for more examples of what can be done with modules and see `man modulefile` for more information.

Last modified 14 Jun 2018 MK/SGK

Compilers & Libraries, and MPI or Multi-threaded Programs

1. Compilers

We support the following three different compilers:

1. The GNU compilers (`gcc, g++, gfortran`)
2. The Intel compilers (`icc, icpc, ifort`, and the new LLVM ones: `icx, icpx` and `ifx`)
3. The NVIDIA compilers (`nvcc, nvc++, nvfortran`).

Some form of MPI is available for each compiler (although not all flavors for all versions of each compiler).

To access a compiler, use the corresponding module:

	GNU	Intel	NVIDIA
	`module load gcc`	`module load intel`	`module load nvidia`
default version	8.5.0	2024.0	23.9
also available	4.9.1, 4.9.2, 5.3.0, 6.1.0, 7.3.0, 9.2.0, 9.3.0, 10.1.0, 11.2.0, 12.2.0, 13.2.0	2021.3, 2021.4, 2022.1, 2022.2, 2023.1	21.9, 22.9, 23.11, 23.5, 24.3
Note		As of 2021 Intel has repackaged their compilers as `OneAPI` and has changed the compilers names in their most recent releases.	

To use a specific version, add the version number as in

```
% module load gcc/12.2.0
```

or

```
% module load intel/2021.4
```

etc.

Note

- The default values and the list of available values might change before this documentation page is updated.
- To check what versions are available, use something like
  `% ( module -t avail ) |& grep gcc`
- In most cases, you cannot mix and match compilers, their respective libraries, and the associated run-time environment, doing so may lead to unpredictable results.
- As of 2020 NVIDIA has acquired PGI and repackaged their compilers as the NVIDIA compilers.
    - The old PGI compilers are no longer available since Hydra upgrade to Rocky 8.9 (May 2024)

2. Libraries

The following libraries are available:

Library	Description	Where to find examples
BLAS & LAPACK	Linear Algebra libraries	`~hpc/examples/lapack`
MKL	Intel's Math Kernel Library	`~hpc/examples/lapack/intel`

GSL	GNU Scientific Library	`~hpc/examples/gsl`

⚠ The NVIDIA LAPACK library crashes or hangs in some situations (see README under `~hpc/examples/lapack/nvidia).`

Last updated 16 May 2024 SGK

Building & Running MPI

How to Build and Run MPI Programs

	GNU	Intel	NVIDIA

modules	`gcc/V.R/openmpi`
	`gcc/V.R/mvapich`
		`intel/YY/mpi` (vendor's version)
		`intel/YY/openmpi`
		`intel/YY/mvapich`
			`nvidia/YY/mpi` (vendor's version)
			`nvidia/YY/openmpi`
			`nvidia/YY/mvapich`

Notes	`where V.R` is the version and release numbers:
	`gcc/8.5/openmpi`
	you can also specify the OpenMPI version:
	`gcc/8.5/openmpi4`
	or use the full OpenMPI and GNU versions:
	`gcc/8.5/openmpi4.1.6-8.5.0`
		Where YY is the version (year):
		`intel/24/mpi`
		you can also specify the OpenMPI version:
		`intel/24/openmpi4`
		or use the full OpenMPI and Intel versions:
		`intel/24/openmpi5.0.1-24.0`
			Where YY is the version (year):
			`nvidia/24/mpi`
			you can also specify the OpenMPI version:
			`nvidia/24/openmpi4`
			or use the full OpenMPI and NVIDIA versions:
			`nvidia/24/mmvapich2.3.7-24.3`

Examples
under
~hpc/examples
	`mpi/openmpi3/gcc`
	`mpi/openmpi4/gcc`
	`mpi/openmpi5/gcc`
	`mpi/mvapich/gcc`
		`mpi/intel` (vendor's version)
		`mpi/openmpi3/intel`
		`mpi/openmpi4/intel`
		`mpi/openmpi5/intel`
		`mpi/mvapich/intel`
			`mpi/nvidia` (vendor's version)
			`mpi/openmpi3/nvidia`
			`mpi/openmpi4/nvidia`
			`mpi/openmpi5/nvidia`
			`mpi/mvapich/nvidia`

Note

- MPI jobs must request either the `orte, mpich` or `ompi` parallel environment with the number of slots (CPUs, computing elements, etc)
  - OpenMPI uses `orte`,
  - MVAPICH uses `mpich`,
  - *except* that NVIDIA supplied openmpi uses/needs `ompi`.
- The job script should use the environment variable `NSLOTS` (via `$NSLOTS`) to access the assigned number of CPUs (slots), that number should not be hardwired.
- the list of nodes set aside for your MPI job is compiled by the jobs scheduler (GE) and passed to the job script via a machine file that file is either
  - `$PE_HOSTFILE`

    or
  - `$TMPDIR/machines`
- Whether you build or run an MPI program, you must first load the corresponding module, before invoking `mpirun`.
- I recommend to log what computes nodes your MPI job is using with commands listed on the "info" line

	ORTE	MPICH	OMPI
qsub	`-pe orte N`	`-pe mpich N`	`-pe ompi N`

info	`echo using $NSLOTS slots on:`
	`cat $PE_HOSTFILE`
		`echo using $NSLOTS slots on:`
		`sort $TMPDIR/machines | uniq -c`
			`echo using $NSLOTS slots on:`
			`sort $TMPDIR/hostfile`

module	`module load XXX/YYY/ZZZ`	`module load XXX/YYY/ZZZ`	`module load nvidia/YY/mpi`
run	`mpirun -np $NSLOTS ./code`	`mpirun -np $NSLOTS -machinefile $TMPDIR/machines ./code`	`mpirun -np $NSLOTS -hostfile $TMPDIR/hostfile ./code`

where

- XXX/YYY/ZZZ is the right module name, and
- N is the number of slots you want your code to use,
  - it can also be specified as "`N-M`", meaning at least `N` and at most `M` CPUs (slots, ...)

⚠️ This can be confusing, so look at the examples for the compiler/mpi-flavor you use. ⚠️

💡 You can find more information under Submitting Distributed Parallel Jobs with Explicit Message Passing.

Last updated 16 May 2024 SGK

Building & Running Multi-threaded Programs

How to Build Multi-Threaded Programs

- You can build and run multi-threaded programs on the cluster;
- You can either write, or use, a program that starts separate threads to parallelize tasks, or
- you can use the compilers to produce multi-threaded code, using OpenMP directives, known as pragmas.
  A pragma is a directive that looks like a comment, but get parsed by the compiler when invoking it with the appropriate flag.
- or you can write code that explicitly create multiple threads (via fork), etc.

- A multi-threaded code (application) must run on a single compute node and usually uses a shared memory model;
  - The total cumulative available memory of a multi-threaded code is thus limited to the largest amount of memory available on any compute node;
    by contrast MPI code uses a distributed memory model, and can thus access a much larger cumulative amount of memory.
  - Similarly, the total number of threads (CPUs) available to a multi-threaded code is thus limited to the largest amount of CPUs available on any compute node;
    by contrast MPI code uses a distributed model, and can thus make use of a much larger total number of CPUs.

How to Run a Multi-Threaded Program

- To submit a job that will use a multi-threaded application, you must
  - request a number of CPUs via the qsub option `-pe mthread N`, where `N` is the number of CPUs
  - The number of requested CPUs can also be specified as `N-M`, meaning at least `N` and at most `M` CPUs
  - The more CPUs you request, the less likely it is that many machines will have that many CPUs and/or that many free CPUs,
  - The maximum number of available CPUs (in the regular queues) is 64, and drops to 40 or 24 for some special nodes.

- The job script should use the environment variable `NSLOTS` (via `$NSLOTS`) to access the allocated number of CPUs (slots), that number should not be hardwired.

Compiling Using OpenMP

- The following compiler flags enable `OpenMP` pragmas in your code:

Compiler	Flag	
GNU	`-fopenmp`	
Intel	`-qopenmp`	`-openmp is deprecated`
PGI/NVIDIA	`-mp`	

  How to parallelize a code using `OpenMP` directives is beyond the scope of this set of documentation.
- `OpenMP` code uses the environment variable `OMP_NUM_THREADS` to specify the number of threads, it should thus be set to `NSLOTS`:

C-shell (csh) syntax	Bourne shell (sh) syntax
`setenv OMP_NUM_THREADS $NSLOTS`	`export OMP_NUM_THREADS=$NSLOTS`

You can find more information under Submitting Parallel Jobs.

Last updated 17 Nov 2021 SGK

Packages and Tools

The following Packages and Tools are supported on Hydra:

1. IDL & FL
2. Java
3. Julia
4. Matlab
5. Python
6. R
7. Conda: Anaconda, Miniconda (& Mamba)

Consult the pages Genomics Software for additional information.

Last update 16 May 2024 SGK/MPK

IDL, GDL & FL

- We have 5 interactive licenses and 128 run-time licenses for IDL (up to v 9.0.0), and have installed FL (up to version 0.79.54).
- GDL and FL are open-source (read no licenses needed) idl-like package.
    - ⚠️ GDL is no longer available on Hydra.

1. IDL

- The interactive licenses are available for all versions of IDL, from v8.7 to v9.0, and should be used only on the login nodes for pre- or post processing.
- to access IDL, simply load the `idl` module:
  `% module load idl`
- To view all the available versions, use:
  `% ( module -t avail ) |& grep idl`
- Running IDL the normal way (i.e., interactively) in jobs submitted to the queue will quickly exhaust the available pool of licenses,
    - 💡 instead use the run-time licenses.

Using IDL with Run-Time Licenses

- To run IDL with a run-time license you must first compile your IDL code, and all the ancillary procedures and functions it may use and save it in a save set.
- The following example shows how to compile a procedure called `reduce`, stored in the file `reduce.pro`, to a complete save set that will be called `reduce.sav`

How to compile reduce.pro and save it as a save set

```
% module load idl/rt
% idl
IDL> .run reduce
IDL> resolve_all
IDL> save, /routine, file='reduce.sav'
```

- After creating a save set, changes to any segment of the code will not be reflected in the `.sav` file; you must recompile the code each time you modify it.
- To run IDL in run-time mode, you load the `idl/rt` module and use the `-rt=XXX` flag when invoking IDL.
- To let the GE know that you will pull an IDL run-time license, use the `-l idlrt=1` flag when `qsub`'ing.
  The GE will keep track of the number of licenses used and will limit accordingly the number of concurrent jobs using them to the number of available licenses.
- Here is an example of a job file `reduce.job` that will run the `reduce` procedure:

IDL run-time example job file

```
# /bin/csh
#
#$ -cwd -j y -o reduce.log -N reduce
#$ -l idlrt
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
#
module load idl/rt
#
idl -rt=reduce.sav
#
echo = `date` job $JOB_NAME done
```

- You then run that job with:
  `% qsub reduce.job`

Notes

- The name of the save file must match the name of the top procedure to run, and
- there is no way to pass arguments to that top level procedure, when started in with `-rt=name.sav`, i.e.:
  If you want to run the procedure `make_my_model`, you should write `make_my_model.pro` and compile it to a `make_my_model.sav` file.
- IDL procedures can read parameters from a file, or from standard input (`stdin`), for example:

Example of reading from stdin (standart input)

```
idl -rt=make_my_model.sav<<EOF
5943.
124.
sun
EOF
```

the corresponding procedure would look like this

Corresponding procedure `make_my_model.pro`

```
procedure make_my_model, temperature, density, name
;;
;; this procedure computes a model using a given temperature and density
;;   and saves it to a file whose name is derived from given string
;;
if n_params() eq 0 then
  ;;
  ;; no parameters passed, so we need to read the parameters from stdin
  ;; let's initialize them to set the variable type (double and string)
  ;;
  temperature = 0.D0
  density = 0.0D0
  name = ''
  ;;
  read, temperature
  read, density
  read, name
endif
;;
fullName = name + '.dat'
print, 'running a model for T=', temperature,', rho=', density
print, 'saving it to "'+fullName+'"'
;;
[the code goes here]
;;
end
```

  - If the reference to the `.sav` file is to a different (sub)directory, IDL will execute a `cd` to that directory, namely
    ```
    idl -rt=src/make_my_model
    ```
    is in fact equivalent to
    ```
    cd src
    idl -rt=make_my_model
    ```

  - Some IDL procedures will use more than one thread (more than one CPU) if some operations can be parallelized.
    In fact IDL queries how many CPUs there are on the current machine to decide on the maximum number of threads it may start, assuming effectively that you are the sole user of that computer.
    ⚠️ This is not appropriate for a shared resource and it should not be used.
    To avoid this you must add the following instruction:
    CPU, TPOOL_NTHREADS = 1
    to limit IDL to using only one thread (one CPU).

    Alternatively, you can request several slots (CPUs, threads) on a single compute node when submitting a job (`qsub -pe mthread N`, where N is a number between 2 and 64),
    and tell IDL to use that same number of threads with the `CPU` command.
    Of course, if your IDL code is not using N threads all the time, you will be grabbing more resources than you will be using, something that should be avoided.

  - You can check if IDL was started in run-time mode with
    ```
    IF lmgr(/runtime) THEN message, /info, 'runtime mode is on'
    ```

- Some IDL instructions are not allowed in run-time mode to prevent emulating interactive mode, consult the manual.
- You can check how many run-time licenses the GE thinks are still available with
  ```
  %  qhost -F idlrt -h global
  ```

- All versions of IDL are now using the new IDL license manager.

2. ~~GDL &~~ FL

- GDL & FL are an open source packages compatible with IDL:
  - GDL is compatible with version 7.1, see http://sourceforge.net/projects/gnudatalanguage, but is no longer available on Hydra;
  - FL is compatible with version 8, see https://www.flxpert.hu/fl
- These are free software:
  - you get what you paid for, but
  - there is no licensing nor run-time limitations.

- The version 0.79.54 (and a few other) of FL are available on Hydra, and is accessible by loading the respective modules:
  ~~% module load tools/gdl~~
  ~~or~~

```
% module load tools/fl
```

- ~~The module will set variable GDL_STARTUP to either ~/.gdl.startup.0.9.5 or ~/.gdl.startup, if either file exists (checked in that order).~~
  ~~Any GDL commands in the startup file will be executed every time GDL is started as if typed at the prompt.~~
- ~~Like IDL, some GDL procedures will use more than one thread (more than one CPU) if some operations can be parallelized.~~
  ~~GDL queries how many CPUs there are on the current machine to decide on the maximum number of threads it may start, assuming effectively that you are the sole user of that computer.~~
  ⚠ ~~This is not appropriate for a shared resource and it should not be used. To avoid this you must add the following instruction:~~
  ~~CPU, TPOOL_NTHREADS = 1~~
  ~~to limit GDL to using only one thread (one CPU).~~

  Alternatively, you can request several slots, as described for IDL above, with the same caveats.

Last update 16 May 2024 SGK

Java

- Java versions 17, 18 & 21 are available on Hydra by loading the appropriate module:

  ```
  % module load tools/java
  ```

  or

  ```
  % module load tools/java/18
  ```

or

```
% module load tools/java/21
```

- ⚠️ java is now under tools/
- ⚠️ java 18 does not play nice with the GridEngine:

  - Java, by default, wants to starts as many threads and grab as much memory as possible.
  - By not specifying some memory related parameters, java fails in every submitted job, with the following message:

    ```
    Error occurred during initialization of VM
    Could not reserve enough space for object heap
    ```

  - 💡 You should always start java with the following options:

    ```
    java -d64 -server -XX:MaxHeapSize=1g
    ```

    where the value "`1g`" should be adjusted to the memory needed by the application and for the job to fit within the queue and the requested resources memory constraints.

    The total amount of memory used by java is not just the maximum heap size.
  - If you need more memory, be sure to adjust the memory resource request accordingly (`-l memres=X,h_data=X,h_vmem=X`), see the section about memory reservation in the Available Queues page.

💡 The complete documentation for all of `java` options (all versions) is posted at Oracle's web site..

Last update 16 May 2024 SGK/MPK

Julia

- Several versions of julia (up to 1.10.2, see julialang.org) are available by loading the corresponding module:

  ```
  % module load tools/julia
  ```

  ```
  % module load tools/julia/1.6.3
  ```

  etc...

- `Check with`

  ```
  % module whatis julia
  ```

  % module help tools/julia

  ```
  % man julia
  ```

  and at `julialang.org & juliaplots.org`

Last update 16 May 2024 SGK/MPK

Matlab

- Full fledged MATLAB is not available on `Hydra`
  - we would need to purchase licenses for Hydra,
  - Hydra users can use compiled MATLAB, since we installed the MATLAB run-time environment,
  - SAO users can use the ITS supported `MATLAB` compiler to produce run-time version of their MATLAB code.

- The `MATLAB` run-time environment is available on `Hydra`
  - to access it, load the right module:

Module	Description
`matlab/R2014a`	2014 first release
`matlab/R2017b0`	2017 second release (SAO/CF equivalent)
`matlab/R2017b`	2017 second release, with (latest) update (# 9)
`matlab/R2019a`	2019 first release
`matlab/R2019b`	2019 second release
`matlab/R2020a`	2020 first release
`matlab/R2020b`	2020 second release
`matlab/R2021a`	2021 first release
`matlab/R2021b`	2021 second release
`matlab/R2022a`	2022 first release
`matlab/R2022b`	2022 second release
`matlab/R2023a`	2023 first release
`matlab/R2023b`	2023 second release
`matlab/rt  R2023b`	default run-time is set to use R2023b

Note that:

- you must compile your `MATLAB` application to run it on Hydra,
- SAO has a single (concurrent) seat license for the `MATLAB` compiler, available on all ITS-managed machines.

Last update 16 May 2024 SGK/MPK

Python

- The default Python with Rocky 8.x is version 3.6.8;
  - so unless you load a specific module, you will run these versions of Python.

- Additional versions of Python are available as follows:

Module	Version	Comment
none	3.6.2	`python` under Rocky 8.9
none	2.7.18	`python2` under Rocky 8.9
`python3`	3.9.18	BCM version
`python39`	3.9.18	BCM version
`tools/python`	3.11.4 :: Anaconda, Inc.	current default
`tools/python/2.7`	2.7.16 :: Anaconda, Inc.	
`tools/python/3.7`	3.7.3 :: Anaconda, Inc.	Anaconda built/full support
`tools/python/3.8`	3.8.8 :: Anaconda, Inc.	Anaconda built/full support
`tools/python/3.9`	3.9.8	Built from source
`tools/python/3.9a`	3.9.13	Built from source
`tools/python/3.10`	3.10.0	Built from source
`tools/python/3.11`	3.11.4 :: Anaconda, Inc.	Anaconda built/full support
`tools/python/3.11b`	3.11.7 :: Anaconda, Inc.	Anaconda built/full support
`intel/python`	3.9.7 :: Intel Corporation	Intel's version
`intel/python/37-21.4`	3.7.11 :: Intel(R) Corporation	Intel's version w/ 21.4
`intel/python/39`	3.9.7 :: Intel(R) Corporation	Intel's version
`intel/python/39-22.1`	3.9.7 :: Intel(R) Corporation	Intel's version w/ 22.1
`intel/python/39-23.1`	3.9.26 :: Intel(R) Corporation	Intel's version w/ 23.1
`intel/python/39-24.0`	3.9.18 :: Intel(R) Corporation	Intel's version w/24.0

💡If you are looking for the versions with all the usual packages, use the `tools/python` one. This list will vary as we install new versions.

Package Installation and Management

We have installed a long list of packages with the Anaconda versions.

You can view that list with (after loading the appropriate module)

```
% module load tools/python

% pip list
```

You can install additional packages without elevated (admin/root) privileges with

```
% pip install --user <package-name>
```

Alternatively you can use `conda` to manage your Python packages and beyond that manage your environment - rather than using module files.

- If you only need a handful of packages that have no incompatibilities with the ones already installed and
- you can use one of the available python version you may chose not to bother with `conda,` and use `pip install --user.`
- For further explanations on Anaconda and Miniconda and instructions on using `conda`, consult the Conda: Anaconda & Miniconda page.

Python Multi-Threading (NUMPY and others)

- By default, `NUMPY,` as well as other packages, are built to use multi-threading, namely some numerical operations will use all the available CPUs on the node it is running on.
    - ⚠ This is *NOT* the way to use a shared resource, like Hydra,
    - The symptom is that your job is oversubscribed.
- The solution is to tell `python` how many threads to use, using the respective module:

  serial case

  ```
  module load tools/python/use-single-thread
  ```

  or

  multi-thread case

  ```
  module load tools/python/use-multi-thread
  ```

  use `module show <module-name>` to see what is done, the multi-thread version will use the value stored in the environment variable NSLOTS to match the requested resource (`-pe mthread`).

  If you distribute your python code with mpi, use the single thread version.

  You can use

    - `tools/python-single-thread-numpy` or `tools/single-thread-numpy` instead of `tools/python/use-single-thread`

  and

    - `tools/python-mthread-numpy` or `tools/mthread-numpy` instead of `tools/python/use-multi-thread`

  for backward compatibility.

Example

demo-mthread-numpy.job

```
#
# this example uses 4 threads
#$ -pe mthread 4
#$ -cwd -j y -o demo-mthread-numpy.log -N demo-mthread-numpy
#
echo + `date` $JOB_NAME started on $HOSTNAME in $QUEUE with id=$JOB_ID
echo NSLOTS = $NSLOTS
#
module load tools/python/3.8
module load tools/python/use-multi-thread
python my-big-data-cruncher.py
#
echo = `date` $JOB_NAME done.
```

Last update 16 May 2024 SGK/MPK

R

- R version 4.4.0 (built with Rocky 8 default gcc, i.e. 8.5.0) is available by loading the corresponding module:
  `% module load bio/R`
- The `tools/R` module is now equivalent to `bio/R` and `bioinformatics/R`, and will default to version 4.4.0
- Users who need a more recent version of R should consider using Anaconda to install R and needed packages in their space. See below for info on Anaconda on Hydra.

- Installing Packages:
  Users can install packages their own packages in a user-specific library. When the `install.packages()` command is used, there will be a notification that the system-wide library is not writable and you will prompted to create a personal library in your home folder. All future packages that you install will be installed into your personal library. You will then be prompted to choose a repository.

  Packages only need to be installed one time to be accessible to all of your R jobs on all nodes on the cluster. Because compute nodes don't have internet access, you will need to run the `install.packages()` command interactively from the login node.
- $NSLOTS in R scripts
  It is best practice to use the environmental variable `NSLOTS` in your scripts to specify the number of cores to use for R commands that supports multiple cores. By using `NSLOTS` rather than hard-coding the number of cores in your R script, your job will utilize the requested number of slots, even if you change your qsub submission parameters.

  💡 Use the R Base function `Sys.getenv()` to access the value of `$NSLOTS` from within your script.

  ```
  ## Script using Sys.gentenv() to read the value of $NSLOTS

  numcores <- Sys.getenv("NSLOTS")
  ```

- Proper parallelization of `makeCluster()`:
  R packages that incorprate the `makeCluster()` function must specify `type="FORK"` as an argument. Without this, Hydra's scripts that kill zombie jobs will terminate the R processes created by `makeCluster()`.

  ```
  cl <- makeCluster(no_cores, type="FORK")
  ```

Last update 16 May 2024 SGK/MPK

Conda: Anaconda & Miniconda

1. Introduction

Conda is a free & open-source package and environment management system, see conda.io:

- It was started by the Python community (hence the snake name reference) to help handling the plethora of Python packages,
    - but it has been extended to support any languages (*Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, FORTRAN*) and to manage the OS environment.
- It is used by a large community in data science and biology.
- It requires no elevated (admin/root) privileges to install software and/or create "environments", if used properly and works under different OSes.
- Conda main development is carried out by a commercial company, Anaconda, that offers free and paid versions; the free version works great.
- There are two free versions:
    1. The full fledge version, Anaconda.
    2. A small, bootstrap version of Anaconda, called Miniconda.

2. Anaconda or Miniconda?

- Users may find it convenient to use Anaconda or Miniconda to install and manage software in their user space.
- It is acceptable to use these systems, but like compiling and running any software on Hydra, the user should be familiar with how the software functions and uses resources (memory, CPUs, disk) so as not to overutilize cluster resources.
- User can either:
    - install Miniconda,
    - install the full-fledged Anaconda,
    - use the pre-installed full-fledged Anaconda we support.
- Installing Miniconda or using the pre-installed Anaconda are the preferred options for Hydra, since Miniconda initially installs minimal packages, hence reducing disk usage (required dependencies will be installed by `conda`).
- Using the pre-installed Anaconda requires the least efforts from the user, but in some cases might not offer the user full control:
    - users cannot modify or update the "base" installation,
    - instead user should create their own `conda` environment(s) and adjust them as needed.
- If the "base" needs to be updated, please contact us.

a. Do NOT Mix and Match

⚠ **WARNING**

You cannot mix and match Miniconda, Anaconda or pre-installed Anaconda

⚠ If you have installed your own Miniconda or Anaconda, you may have executed:

```
      % conda init
```

If you did, you will have stuff like this

```
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!

...

# <<< conda initialize <<<
```

in your `~/.bashrc` file.

⚠ While this is convenient if you always use that same `conda` (your Miniconda, your Anaconda or the pre-installed Anaconda), it does not allow you to *safely* use a different one.

- If you want to be able to switch to a different `conda`, delete these lines from your `~/.bashrc` file.
- You can re-execute `conda init` after switching.
- If you want to be able to switch back and forth, delete these lines and do not execute `conda init` (see below for recommended usage of the pre-installed full-fledged Anaconda).

b. Pre-installed Anaconda

⭐ This is the most direct way to start using `conda` on Hydra as it uses a module that is already setup on the system.⭐

- You can access the pre-installed full-fledged Anaconda simply by loading the `tools/conda` module and enabling `conda` with the command `start-conda`.
- The recommended procedure to enable `conda` each the time you log on Hydra is to add these two lines

  ```
  module load tools/conda

  start-conda
  ```

  to either your `~/.bashrc` or your `~/.cshrc` file (i.e. whether your login shell is `bash` or `csh` respectively).

  You don't have to enable `conda` for each time you login, tho.

Job files

- To enable `conda` for your jobs that use the Bourne shell (`-S /bin/sh`)
  - either add these two lines in your job file, or
  - add the fist line in a file called `~/.profile`, and the command `start-conda` in the job file.
- To enable `conda` for your jobs that use the C shell (`-S /bin/csh` or no `-S` option passed):
  - either add these two lines in your `~/.cshrc`, or
  - add these two lines in you job file, or
  - the first one in your `~/.cshrc` and the second in the job file

⭐ The most flexible method is to type the two lines each time you log in, and adding them to the jobs that need it. ⭐

c. Miniconda

To install Miniconda:

- Download the latest 64-bit installer.
- Follow these instructions to install it in your user space (or the Installing Miniconda on Hydra instructions).
  - The default location to unpack the software is your home directory.
  - This works well because `/home` is not scrubbed and the disk space needed by miniconda typically is within the user quotas for `/home`.
- After installation, you can use `conda` to install software and create `conda` environments.

Job Files

- To use software installed via `conda` in submitted jobs, you can create a personal module file to add your miniconda `bin` directory to your `PATH`.
  - Follow these instructions.

3. Using the `conda` command

- Instructions on how to use `conda` are in the Using Conda page.

Last update 19 Nov 2021 SGK/MPK

Installing Miniconda on Hydra

Although using the pre-installed Anaconda with the `tools/conda` module requires the least efforts from the user, there may be cases where you want full control of the installation such as modifying the "base" environment. If that is the case, installing your own copy of Miniconda in your user space is an option.

Installing

- Where to install? We suggest your home directory: it's not scrubbed

1. Download: We want the Linux, 64-bit, Python3 installer found on this page: https://docs.conda.io/en/latest/miniconda.html (Note: We recommend getting the Python3 installer even if you need Python2 for some programs. You can setup a separate Python2 environment when needed.)

```
$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

2. Run installer. Make sure to enter "yes" to "running conda init":

```
$ sh Miniconda3-latest-Linux-x86_64.sh
...
Do you accept the license terms? [yes|no]
[no] >>> yes
...
[/home/user/miniconda3] >>> [press enter key]
PREFIX=/home/user/miniconda3
...
Do you wish the installer to initialize Miniconda3
by running conda init? [yes|no]
[no] >>>yes
```

3. Log off Hydra and log back in.
4. Look for `(base)` at the beginning of your command prompt.
5. Test with: `which conda`

```
(base) $ which conda
~/miniconda3/bin/conda
```

⚠ If you don't see `(base)`, and your shell is `bash` (true for biology users) there may be an issue with a config file in your home directory.

You can fix this with: `nano ~/.bash_profile` and then append this text to the bottom of the file:

```
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi
```

Log out and back in again to see if `(base)` is now at the beginning of your command prompt.

What did the installer do?

- `~/miniconda3/`: where all the conda programs are installed
- `~/.bashrc` (for `bash` users, this file is run when you start a new `bash` shell): a command has been appended to enable conda (modifying your `$PATH`) and changing your prompt.
- `~/.conda/` and `~/.condarc`: hidden directory and file with settings

Last updated 19 Nov 2021 MPK/SGK

Using Conda

1. Packages

- A conda package is the precompiled program and a link to all of the required packages that will also be downloaded and needed.
- A whole pipeline can be defined in a package: all the scripts and dependent software (e.g. qiime2 or phyluce)
- Packages are created by people in the community and shared publicly. Sometimes they're created by the software developer, but they're often by users of the software.
- You can create your own packages which a great method for reproducible science and collaboration. We're not covering that, but we can direct you to resources.

`conda` command

We'll be using the `conda` command for managing packages and controlling conda

Help with conda

- `conda help`
- `conda install --help`
    - You can consults the 'conda' command reference documentation, or
    - the conda "cheat sheet."

Listing installed packages

Some packages come pre-installed with either Anaconda or Miniconda (a lot more if you're using Anaconda), for Minconda:

```
(base)$ conda list
# packages in environment at /home/user/miniconda3:
#
# Name                    Version                   Build  Channel
_libgcc_mutex             0.1                        main
asn1crypto                1.2.0                    py37_0
ca-certificates           2019.10.16                    0
certifi                   2019.9.11                py37_0
cffi                      1.13.0           py37h2e261b9_0
chardet                   3.0.4                 py37_1003
conda                     4.7.12                   py37_0
conda-package-handling    1.6.0            py37h7b6447c_0
cryptography              2.8              py37h1ba5d50_0
...
```

How to find `conda` packages

You can search the public package repositories at https://anaconda.org/ ⭐ **What we suggest using**

- Use the anaconda.org search feature.
- Copy the install command from the package's page

Or do a web search: "conda r" or "bioconda mafft"

Or use `conda search **"blast"**` (finds packages with blast anywhere in their name in your current channels)

- Search for admixtools on anaconda.org
  - Only available on bioconda
  - `conda install -c bioconda admixtools`
- Search for mafft on anaconda.org
  - Available on multiple channels
  - `bioconda`, `conda-forge`, `anaconda`, `r` are reliable channels. You can try other popular ones.

Installing a package

`conda install ...` installs a packages and its dependencies

- It doesn't matter what your current directory is, it will install in your "conda" directory!
- ⚠️ if you use the pre-installed version of Anaconda, you won't be able to install (or update) anything in the base environment,
  - 💡 you will need to create and active your own `conda` environment(s) first (see below) and you will see that name in lieu of `(base)` in the following examples.

```
(base)$ conda install -c bioconda mafft

The following packages will be downloaded:
...
The following NEW packages will be INSTALLED:
...
Proceed ([y]/n)? y
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Test it:

```
(base)$ mafft --help
------------------------------------------------------------------------------
  MAFFT v7.471 (2020/Jul/3)
  https://mafft.cbrc.jp/alignment/software/
  MBE 30:772-780 (2013), NAR 30:3059-3066 (2002)
------------------------------------------------------------------------------
```

Adding `bioconda` and `conda-forge` to your channels

```
(base)$ conda config --add channels defaults
(base)$ conda config --add channels bioconda
(base)$ conda config --add channels conda-forge
```

`conda config --get channels` shows your current channels

Which channel is the highest and which is the lowest?

```
$ conda config --get channels
--add channels 'defaults'   # lowest priority
--add channels 'bioconda'
--add channels 'conda-forge'   # highest priority
```

`conda-forge` is the highest, `defaults` is the lowest.

Note: the more channels you have, the increased time to "solve" installation.

```
(base)$ conda install iqtree

...
The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2020.6.20  |       hecda079_0         145 KB  conda-forge
    certifi-2020.6.20          |   py37hc8dfbb8_0         151 KB  conda-forge
    conda-4.8.3                |   py37hc8dfbb8_1         3.0 MB  conda-forge
    iqtree-2.0.3               |       h176a8bc_0         2.8 MB  bioconda
    openssl-1.1.1g             |       h516909a_0         2.1 MB  conda-forge
    python_abi-3.7             |          1_cp37m           4 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         8.2 MB
...
```

`conda` install tricks

- Installing multiple items
  - `conda install mafft gblocks`
  - ⭐“It is best to install all packages at once, so that all of the dependencies are installed at the same time.” https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-pkgs.html
- Specifying program versions
  - Show available versions with: `conda search python`
  - `conda install python=3`
  - `=3` matches the most recent version starting with `3`
  - `==3.7.5` matches exactly the version specified
  - `">3.7"` matches most recent version greather than 3.7 (in quotes because of the > which will interfere with the shell's redirection features)
  - `"<3.8"` matches most recent version less than 3.8 (in quotes because of the < which will interfere with the shell's redirection features)
  - See the conda documentation for more information

2. Using conda in submitted jobs

There are a couple steps needed to utilize `conda` in your submitted jobs.

⚠️ By default the installed programs won't be found.

If you want to use your Miniconda installation:

Create a module file

There is Hydra documentation about creating module files here.

We'll set one up for your miniconda install. Module files are text files with instructions on how the current environment should be modified.

We suggest creating a directory in your home folder called `modulefiles` and creating a module called miniconda in that directory.

```
(base)$ mkdir ~/modulefiles
(base)$ cd ~/modulefiles
(base)$ nano miniconda
```

and then enter into the new text file:

```
#%Module1.0
prepend-path PATH /home/your_username/miniconda3/bin
```

⚠️ Make sure to change `your_username` to your Hydra username!

Job file using your module file

Use the `module load` command followed by the path and name of your module file to load your module: `module load ~/modulefiles/miniconda`

Example job file using:

```
# /bin/sh
# ----------------Parameters---------------------- #
#$ -S /bin/sh
#$ -q sThC.q
#$ -l mres=2G,h_data=2G,h_vmem=2G
#$ -cwd
#$ -j y
#$ -N minicondatest
#$ -o minicondatest.out
#
# ----------------Modules------------------------ #
module load ~/modulefiles/miniconda
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo + NSLOTS = $NSLOTS
#
echo "Conda location:"
which conda

echo
echo "Installed packages:"
conda list

#
echo = `date` job $JOB_NAME done
```

Excerpt of `minicondatest.out`:

```
...
Conda location:
/home/user/miniconda3/bin/conda

Installed packages:
# packages in environment at /home/user/miniconda3:
#
# Name                    Version                   Build  Channel
...
```

If you use the pre-installed Anaconda or Miniconda

- No need to create a module
  - but you need to load the `tools/conda` module and run `start-conda`

  Simply replace, in the examples above, the line

  `module load ~/modulefiles/miniconda`

  by the following two lines

  `module load tools/conda`

```
start-conda
```

Even if you have these two lines in your `~/.bashrc` file. 💡 You can save typing the first line in your job file if you insert it in a file called `~/.profile`

- To enable `conda` for your jobs that use the C shell (`-S /bin/csh` or no `-S` option passed):
  - either add these two lines in your `~/.cshrc,` or
  - add these two lines in you job file, or
  - the first one in your `~/.cshrc` and the second in the job file.

⚠️ If you need access to packages installed in an environment (see below) you will need to add the corresponding `conda activate` command, like in

```
conda activate iqtree-v1
```

3. Wait, there's one more concept to keep things clean... `environments`

So far all installs we've done have done into one set of packages called `(base).`

What if programs have conflicting dependencies or you need different versions of the same program?

A `conda` Environment is a compartmentalized set of packages, and if you use the pre-installed Anaconda, it allows you to modify it (since you can't modify the pre-installed Anaconda base environment)

Also, the more packages you have, the longer it will take for conda on the "Solving Environment" step.

What if you wanted to have both iqtree1 and iqtree2 installed? Above we showed how to installed iqtree.

It installed the latest version: 2.0.3. If you also need iqtree 1.x installed, the conda command: `conda install iqtree=1` will downgrade your current iqtree from 2.x to 1.x, i*t won't allow you to have more than one version installed.* You can have environments to have access to the two versions. (Using `iqtree=1` will install the latest version of iqtree that starts with a 1).

Creating an environment

```
(base)$ conda create -n iqtree-v1
Collecting package metadata (current_repodata.json): done
Solving environment: done
## Package Plan ##
  environment location: /home/user/miniconda3/envs/iqtree-v1
Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate iqtree-v1
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

Adding packages to the environment

Although the output says to use `conda activate...`, we recommend using `source activate...` because it is compatible with job files submitted through `qsub`

```
(base)$ source activate iqtree-v1
(iqtree-v1)$ conda install iqtree=1
```

Note: you can create an environment and add packages do it in one step with: `conda create -n iqtree-v1 iqtree=1`

Using an environment in your job file

When submitting a job that uses a specific conda environment, you need to use `source activate ...` *after* you load your miniconda module.

Example script:

```
...
# ----------------Modules------------------------ #
module load ~/modulefiles/miniconda
source activate iqtree-v1
#
# ----------------Your Commands------------------- #
...
```

Removing an environment

To remove an environment and all of its packages:

⚠ *You have to deactivate the environment before removing it*

```
(iqtree-v1)$ conda deactivate
(base)$ conda remove -n iqtree-v1 --all

Remove all packages in environment /home/user/miniconda3/envs/iqtree-v1:

## Package Plan ##

  environment location: /home/user/miniconda3/envs/iqtree-v1

The following packages will be REMOVED:
...
Proceed ([y]/n)? y
```

Useful environments: python2 and R

If you need to a use a program written in python2, you can create an environment with that version of Python

```
(base)$ conda create -n python2 python=2
(base)$ source activate python2
```

You can use conda to install R and R packages

```
(base)$ conda create -n tidyverse r r-tidyverse
(base)$ source activate tidyverse
```

When there isn't a `conda` package for a pipeline

Some pipelines don't have a conda package with all the requirements. You can use the program's documentation to install dependencies.

For example the Hybpiper pipeline doesn't have a conda package (you could create one and contribute it to bioconda...), but The install instructions list these requirements:

- Python 2.7 or later [We suggest python3 when possible -Hydra team]
- BIOPYTHON 1.59 or later
- EXONERATE
- BLAST command line tools
- SPAdes
- GNU Parallel
- BWA
- samtools

Use `conda` to install the dependencies for HybPiper in your `hybpiper` environment. You may need to look up package names at anaconda.org or your favorite search engine. (Hint: pay special attention to GNU Parallel)

```
(base)$ conda create -n hybpiper install python=3 "biopython>1.59" exonerate blast spades parallel bwa samtools

Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/user/miniconda3/envs/hybpiper

  added / updated specs:
  - biopython[version='>1.59']
  - blast
  - bwa
  - exonerate
  - parallel
  - python=3
  - samtools
  - spades

The following packages will be downloaded:
...
The following NEW packages will be INSTALLED:
...
Proceed ([y]/n)? y
...
Downloading and Extracting Packages
...
```

Now that the dependencies are installed, you can download the scripts for HybPiper:

```
cd ~
git clone https://github.com/mossmatters/HybPiper.git
```

⭐ When you're done using an environment, you can go back to `(base)` with: `conda deactivate`

4.Troubleshooting

Finding online help:

- Official conda website (it’s good): https://docs.conda.io
    - Specifically this page called Conda Package Manager
- Web search with “conda” in it, e.g.: conda remove environment (this usually brings me to docs.conda.io)
- si-hpc@si.edu email
- Thursdays Brown Bags (12-1pm, W107 conference room)

Reinstalling

- `mv ~/miniconda3 ~/miniconda3-old` (if you want to keep the old install) or `rm -rf ~/miniconda3` (to delete the old install)
- `rm -rf .condarc .conda`
- Edit ~/.bashrc
    - delete lines from `# >>> conda initialize >>>` to `# <<< conda initialize <<<`
- logout/back in

5. Advanced

- Installing packages using pip if not available on conda
- Exporting environment as environment.yml, and creating a simplified version for working on local computer
    - export: `conda env export > environment.yml`
    - import: `conda env create -f environment.yml`
    - This is how qiime2 install works

6. Appendix

Definitions

- conda: open source package management tool
- Anaconda®: commercial company that develops conda
- Anaconda: distribution of conda with many data science tools pre-bundled
- Miniconda: minimal distribution of conda, other packages can be added
- Channel: grouping of packages managed by one group (e.g. bioconda, conda-forge)

Channels

Major conda channels:

- `Main`: built and maintained by anaconda (setup by default)
- `R`: R and packages for it (setup by default)
- `Conda-Forge`: community contributions
- `Bioconda`: contributions from biological community, find packages on https://bioconda.github.io

7. Additional commands

Remove packages

To remove a package from the current environment use `conda remove [package]`

Update packages

To update a specific package use `conda update [package]`. To update all packages in the current environment use: `conda update --all`

You can prevent `conda update --all` from updating some packages using these instructions

---

Last updated 19 Nov 2021 MPK/MT/SGK

Genomics Software

- You can see a current list of the available genomics and biology software here (or as plain text here).
- We have written detailed instructions for
  - ALLPATHS-LG
  - BEAST
  - BLAST
  - BLAST2GO
  - BUSCO
  - GeneMark-ES
  - MaSurCA
  - SPAdes

If there is a package that you would like to use for genomics that is not installed please contact `si-hpc@si.edu` and we can advise you on how you can compile it for Hydra.

💡 To find out more about a program, including the names of the program's executables and instructions for running on Hydra, log into Hydra and type the command:

```
module help bioinformatics/program
```

💡 The list of module files does not include every version of the software installed. If there are multiple versions, it gives the default version, which is typically the more recent version. To see all modules and versions, log into Hydra and use the command:

```
module avail
```

Last updated 17 Jan 2024 PBF/MGC/MPK/SGK

ALLPATHS-LG

<under construction>

ALLPATHS-LG is a genome assembly program that requires at least two different kinds of Illumina libraries (e.g. paired-end and mate-pair).

Here is a link to the software and documentation: http://software.broadinstitute.org/allpaths-lg/blog/

Parallel Support

Parts of ALLPATHS-LG can run in parallel and performance seems to improve up to 32 threads. More than that does not seem to help.

Memory

ALLPATHS-LG requires running on at least the high-memory queue (512GB), and sometimes the ultra-high-memory queue (1TB). Genome size and coverage depth will determine the amount of RAM you need.

Invoking the program

ALLPATHS-LG runs with two separate commands:

1. **PrepareAllPathsInputs.pl (required arguments)**

Resources: Can usually run on sThM.q, unless you have very high coverage data, then you might need mThM.q

Module: module load bioinformatics/allpaths-lg/52415

Options: PLOIDY=2 for diploid, PLOIDY=1 for haploid

HOSTS=$NSLOTS (# threads)

DATA_DIR=

IN_GROUPS_CSV=

IN_LIBS_CSV=

PHRED_64=True/False depending on type of quality scoring

2. RunAllPathsLG (required arguments)

PRE=

DATA_SUBDIR=

REFERENCE_NAME=

RUN=

THREADS=$NSLOTS

OVERWRITE=True

Sample in_groups.csv file

```
file_name,library_name,group_name
SRR946954_*.fastq,G10cloDADDLAAPE,SRR946954
SRR946955_*.fastq,G10cloDAODBAAPE,SRR946955
SRR946957_*.fastq,G10cloDAODIAAPE,SRR946957
SRR946958_*.fastq,G10cloDAODIABPE,SRR946958
SRR946959_*.fastq,G10cloDAODLAAPE,SRR946959
SRR946960_*.fastq,G10cloDAODMAAPE,SRR946960
SRR946961_*.fastq,G10cloDAODTAAPE,SRR946961
SRR946962_*.fastq,G10cloDAODUAAPE,SRR946962
SRR946963_*.fastq,G10cloDAODWAAPE,SRR946963
SRR946964_*.fastq,G10cloDAODWAAPE,SRR946964
```

Sample in_libs.csv file

```
library_name,project_name,organism_name,type,paired,frag_size,frag_stddev,insert_size,insert_stddev,
read_orientation,genomic_start,genomic_end
G10cloDAODBAAPE,Manakin,Manacusvitellinus,fragment,1,170,17,,,inward,,
G10cloDADDLAAPE,Manakin,Manacusvitellinus,jumping(sheared),1,,,5000,500,outward,,
G10cloDAODIAAPE,Manakin,Manacusvitellinus,jumping(sheared),1,,,500,50,outward,,
G10cloDAODIABPE,Manakin,Manacusvitellinus,jumping(sheared),1,,,500,50,outward,,
G10cloDAODLAAPE,Manakin,Manacusvitellinus,jumping(sheared),1,,,500,50,outward,,
G10cloDAODMAAPE,Manakin,Manacusvitellinus,jumping(sheared),1,,,800,80,outward,,
G10cloDAODTAAPE,Manakin,Manacusvitellinus,jumping(sheared),1,,,10000,1000,outward,,
G10cloDAODUAAPE,Manakin,Manacusvitellinus,jumping(sheared),1,,,20000,2000,outward,,
G10cloDAODWAAPE,Manakin,Manacusvitellinus,jumping(sheared),1,,,2000,200,outward,,
```

Sample qsub scripts

PrepareAllPathsInputs.pl step:

```
# /bin/sh
# ----------------Parameters---------------------- #
#$ -S /bin/sh
#$ -pe mthread 4
#$ -q sThM.q
#$ -l mres=30G,h_data=30G,h_vmem=30G,himem
#$ -j y
#$ -N allpaths_prepare
#$ -cwd
#
# ----------------Modules------------------------- #
module load bioinformatics/allpaths-lg/52415
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo + NSLOTS = $NSLOTS
#
perl PrepareAllPathsInputs.pl DATA_DIR=/path/to/data/ IN_GROUPS_CSV=/path/to/in_groups_manakin.csv IN_LIBS_CSV=
/path/to/in_libs_manakin.csv PHRED_64=True PLOIDY=2 HOSTS=$NSLOTS
#
echo = `date` job $JOB_NAME done
```

RunAllPathsLG step:

```
# /bin/sh
# ----------------Parameters---------------------- #
#$ -S /bin/sh
#$ -pe mthread 8
#$ -q lThM.q
#$ -l mres=60G,h_data=60G,h_vmem=60G,himem
#$ -j y
#$ -cwd
#$ -N allpaths
#
# ----------------Modules------------------------- #
module load bioinformatics/allpaths-lg/52415
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo + NSLOTS = $NSLOTS
#
RunAllPathsLG PRE=/path/to/data/ DATA_SUBDIR=/ REFERENCE_NAME= RUN=run1 THREADS=$NSLOTS OVERWRITE=True
#
echo = `date` job $JOB_NAME done
```

Special notes

BEAST

Performance

Please see: http://www.beast2.org/performance-suggestions/index.html for information about optimizing performance. The BEAGLE libraries will be available when you load the BEAST module and BEAGLE is enabled used by default. There are BEAGLE options such as -beage_SSE which may help performance.

⚠Confirm version of BEAST that is running

Make sure that the intended version of BEAST shows in the log file. The first time beast is run, the program makes a copy of beast.jar and beast.src.jar into `~/.beast/2.x/BEAST/lib`. A separate subdirectory in `~/.beast` is created for versions 2.6 and 2.7. Minor revisions within these releases are not automatically updated when you re-run 2.6 or 2.7. For example, if you run beast 2.6.6, a copy of this jar file is put into `~/.beast/2.6/BEAST/lib`. If you subsequently use the 2.6.7 module, the 2.6.6 version in your `~/.beast/2.6` will be used instead of 2.6.7. To resolve this remove `~/.beast/2.6/BEAST` before running to get a new copy of the minor revision.

Add-ons

To add a beast add-on use the following command after you've loaded the beast module (`module load bioinformatics/beast`) first list available add-ons:

```
packagemanager -list
```

and then add one with

```
packagemanager -add SNAPP
```

The add-on will be downloaded to ~/.beast

Tips on running SNAPP

The SNAPP add-on documentation for BEAST has information on running analyses on a cluster.

1. Add `doNotRun='true'` to the `run` section of your BEAST XML file.
2. Start an interactive session with enough memory to start the BEAST job: `qrsh -pe mthread 8`
3. In the interactive session, load the BEAST module and start the execution of your XML file. In the `-threads` option, specify the number of SNAPP steps you specific in your XML (in the example it is 48)

   ```
   module load bio/beast
   beast -threads 48 your_file.xml
   ```

4. This creates 48 step directories and a `run<number>.sh` script for each one. You can submit a separate job for each of the bash scripts. A sample job file (beast_step_run.job) and a bash script to submit the jobs are below.

beast_step_run.job

```
# /bin/sh
# ----------------Parameters---------------------- #
#$ -S /bin/sh
#$ -q lThM.q
#$ -l mres=16G,h_data=16G,h_vmem=16G,himem
#$ -cwd
#$ -j y
#$ -N beast_step_run
#$ -o beast_step_run.log
#
# ----------------Modules------------------------- #
module load bioinformatics/beast
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo + NSLOTS = $NSLOTS

#
if [ -z $1 ]; then
  echo "Give the run number as an argument: e.g. qsub beast_step_run.job 0"
  exit 1
fi

source ./run${1}.sh
#
echo = `date` job $JOB_NAME done
```

submit.sou

```
for x in $(seq 0 47); do
  qsub -o run${x}.log beast_step_run.job ${x}
done
```

⚠️ Change the `seq` command to match the range of steps. Shown here is for 48 steps which are numbered `run0.sh` to `run47.sh`

5. Submit the steps as separate cluster jobs with: `source submit.sou`
6. When all the jobs are complete, run the `PathSampleAnalyser` to get the final output. In the below example, the output is redirected to the file `pathsampler.out` . You should of course adjust the arguments (like `alpha` and `burnInPercentage` to suite your analysis.

PathSampleAnalyser

```
qrsh -pe mthread 8
module load bio/beast
applauncher PathSampleAnalyser -nrOfSteps 48 -alpha 0.3 -rootdir /path/to/your/runs -burnInPercentage 25
>pathsampler.out 2>&1
```

You can see all the `PathSampleAnalyser` arguments with: `applauncher PathSampleAnalyser -help`

BLAST

BLAST (Basic Local Alignment Search Tool) is a greedy aligner used to match query nucleotide/amino acid sequences against a database. Databases include the complete non-redundant protein/nucleotide databases (nr/nt), single genome databases, and custom databases.

Nucleotide Query

Nucleotide queries use the 'blastn' command (available in the module bioinformatics/blast). Important options include:

```
-task # Specifies the type of query (e.g. "megablast")

-db # Specifies the database to query (e.g.  nt)

-query # Specifies the input file in fasta format

-outfmt # Specifies the format to output hits in (e.g. 5 for xml)

-out # Specifies the name of the output file
```

Example commands:

```
module load bioinformatics/blast

blastn -help # Print help file for all options

blastn -task "megablast" -db nt -query <yourfastafile> -outfmt 5 -out <yourfastahitsinxml> # Megablasts your sequences against the nt database and outputs hits in xml
```

Example qsub file

```
#!/bin/sh
# ----------------Parameters---------------------- #
#$ -S /bin/sh
#$ -j y -cwd
#$ -q lThM.q
#$ -l mres=32G,h_data=32G,h_vmem=32G,himem
#$ -N blast_against_nt
#
# ----------------Modules------------------------- #
module load bioinformatics/blast
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
blastn -task "megablast" -db nt -query <yourfastafile> -outfmt 5 -out <yourfastahitsinxml>
gzip <yourfastahitsinxml>
echo + `date` job $JOB_NAME done
```

Protein Query

Protein queries use the command 'blastp'. Other syntax is similar to blastn. See 'Nucleotide Query' section above for more details for Hydra job creation.

Generating a Custom Database

Custom blast databases can be generated from fasta files using 'makeblastdb' (available in the module bioinformatics/blast).

Example commands:

```
module load bioinformatics/blast

makeblastdb -help # Print help file for all options
makeblastdb -in <yourfastafile> -parse_seqids -dbtype <nucl or prot> # Use 'nucl' for a nucleotide database,
'prot' for a protein database
```

Example qsub file

```
#!/bin/sh
# ----------------Parameters---------------------- #
#$ -S /bin/sh
#$ -j y -cwd
#$ -q sThM.q
#$ -l mres=12G,h_data=12G,h_vmem=12G,himem
#$ -N custom_db
#
# ----------------Modules------------------------- #
module load bioinformatics/blast
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
makeblastdb -in <yourfastafile> -parse_seqids -dbtype <nucl or prot>
echo + `date` job $JOB_NAME done
```

Parallelization

Although BLAST jobs can be natively parallelized using the -num_threads option, often it is better to split large BLAST jobs into smaller files and concatenate the results afterwards. Files containing 1000 to 10,000 sequences each have performed well on Hydra. The jobs can be run using job arrays o r using a bash or csh loop to submit a separate job for each sequence file.

Submitting separate jobs for each input file

Example qsub file

```
# /bin/sh
# ----------------Parameters---------------------- #
#$ -S /bin/sh
#$ -q mThC.q
#$ -l mres=4G,h_data=4G,h_vmem=4G
#$ -cwd
#$ -j y
#
# ----------------Modules------------------------- #
module load bioinformatics/blast
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
#
blastp -db nr -query $1 -outfmt 5 -out $1.out
#
echo = `date` job $JOB_NAME done
```

The `$1` variable in the blastp command above will contain the name of the sequence file when called with the bash command below.

Example bash script to execute qsub

```
for x in *.fa; do qsub -N blast-${x} -o ${x}.log blast.job ${x}; done
```

This script should be run from the Hydra login node command line. In the example it will submit a seperate qsub for each file ending in .fa in the current directory and the qsub job file is named `blast.job`. These should be adjusted for your case. The input file name will be available to the qsub file as `$1`. The qsub command specifies the job name (`-N`) and log file (`-o`) for the scheduler.

Memory and Disk Space Considerations

It is often best to remove duplicate molecules BEFORE BLAST analysis to reduce run-time and results file size.

BLAST jobs are not memory efficient and will easily exceed the capabilities of even high-memory nodes if there are sufficient hits within a query file. If you have a file where you expect most sequences to be identified (e.g. a barcoding project), you should reduce the number of sequences per individual blast job and run more jobs.

BLAST output files are highly redundant. Always compress your files afterwards to reduce disk usage.

Further Information

Detailed explanations of all the BLAST parameters are available from the NCBI: http://www.ncbi.nlm.nih.gov/books/NBK279675/

BLAST2GO

Hydra has a license to run BLAST2GO Command Line (manual here). A key advantage of BLAST2GO on Hydra over workstation versions is that the GO information is stored on a local database rather than using a server on the internet, this greatly increases the speed of mapping and annotation.

License Information

The Hydra BLAST2GO license allows a **single execution** of the command line version of the program on one licensed node. If there is another job using the license, your job will wait in queue until it is available.

Limiting Use to Essential Stages

Because of licensing limitations it is essential to limit the use of BLAST2GO to essential steps, that is, mapping and annotation, but not BLAST.

The BLAST step can be run on Hydra independently of BLAST2GO. A good strategy for running BLAST on Hydra is to split a fasta file and run each part on a different compute node. With this strategy the BLAST output format 5 (BLAST XML) works well. Combine multiple XML output files with the python script `blastXMLmerge.py` that is available when you load the BLAST2GO module. This script takes the name of output file as the first argument and then the list of XML files to be merged: `blastXMLmerge.py combined.xml *.xml` Use the BLAST2GO option `-loadblast <path>` to load your BLAST results.

How to Submit Jobs

Queue specification

- We have created a special queue for BLAST2GO: `-q lTb2g.q`
- You must also request the resource "b2g": `-l b2g`
- Job time limits are the same as other 'long' queues. Memory is limited to 24GB

Command line

- Use the module `bioinformatics/blast2go` to load the dependencies for BLAST2GO
- This will create an alias `runblast2go` which incorporates the java options needed for the program.
- A java `maxheapsize` of 2048m for JAVA is used by default. This can be overridden by setting the environmental variable `BLAST2GO_HEAP_SIZE`
- The `-tempfolder` (where logs and temporary files are put) is set to the current working directory. This can be changed with the environmental variable `BLAST2GO_TEMP`

```
# /bin/sh
# ----------------Parameters---------------------- #
#$ -S /bin/sh
#$ -q lTb2g.q
#$ -l b2g,mres=32G,h_data=32G,h_vmem=32G
#$ -cwd -j y -N b2g-test -o b2g-test.log
#
# ----------------Modules------------------------- #
#
module load bioinformatics/blast2go
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
#
#

# Make local copy of cli.prop
# (this overwrites any existing cli.prop file in the current directory and only needs to be done once)
hydracliprop

# Run example dataset:
runblast2go \
  -properties cli.prop \
  -loadfasta /share/apps/bioinformatics/blast2go/1.5.1/example_data/plant_nucleotide.fasta \
  -loadblast31 /share/apps/bioinformatics/blast2go/1.5.1/example_data/plant_blast_31.zip \
  -mapping \
  -annotation \
  -savebox example.box \
  -savereport example.pdf

#
echo = `date` job $JOB_NAME done
```

cli.prop and Local Database Access

The file `cli.prop` gives the settings for the execution of the program. A template configured with the database access information for running BLAST2GO on the Hydra cluster can be copied to your current directory with the command `hydracliprop` after you load the BLAST2GO module.

Local Database Policy

The local database is large and system constraints limit us from keeping old versions. When the database is updated, the old version will no longer be available.

Outputting Graphs and Statistics

The command line version of BLAST2GO can produce many types of graphs as well as a summary report. The option `-statistics all` will produce all available statistics as png images, csv and .b2g files. Start BLAST2GO with `-statistics` (committing any options) to see a list of available charts. The option `--savereport <name>` creates a PDF with common summary statistics. Creating a combined graph can only be done with the GUI based BLAST 2GO Basic which has a free license. This program can also be used to work with the .b2g files created by the command line version.

18 Nov 2019 MPK

BUSCO

BUSCO (Benchmarking Universal Single-Copy Orthologs) assesses genome assembly completeness by searching for universal and lineage-specific single-copy orthologs in OrthoDB. It can also be used to train gene prediction models for AUGUSTUS. BUSCO requires a genome assembly (in fasta format) and an ortholog database from OrthoDB (see documentation here for best database choices). It also requires access to a writable AUGUSTUS configuration path. To prepare such a path, recursively copy the default path to your partition:

```
cp -r /share/apps/bioinformatics/augustus/gcc/4.9.2/3.3/config /path/to/your/workspace/config
```

You will then have to export this modified configuration path in your job file using:

```
export AUGUSTUS_CONFIG_PATH="/path/to/your/workspace/config"
```

BUSCO is executed using the run_BUSCO.py script. A list of required and common parameters are given below. See documentation here for further details. These can also be given in the config.ini file for BUSCO.

Required Arguments

-i, --in SEQUENCE_FILE: Your genome assembly in fasta format

-o, --out NAME: Name of your output file stems

-l, --lineage_path LINEAGE: Lineage database from OrthoDB

-m, --mode MODE: BUSCO analysis mode (geno/genome for genomes, tran/transcriptome for transcriptomes, prot/proteins for annotated protein gene sets)

Common Arguments

-c, --cpu NUMBER: Number of CPUs. Make sure this matches the requested number of threads in your job file!

-h, --help. Show program help

--long: Turn on AUGUSTUS optimization mode for self-training.

-sp: AUGUSTUS species to use for initial gene predictions. Default is human.

Example Job File

```
# /bin/sh
# ----------------Parameters---------------------- #
#$  -S /bin/sh
#$ -pe mthread 12
#$ -q lThM.q
#$ -l mres=10G,h_data=10G,h_vmem=10G,himem
#$ -cwd
#$ -j y
#$ -N your_genome
#$ -o your_genome.log
#$ -m bea
#$ -M your_email@somewhere.edu
#
# ----------------Modules------------------------- #
module load bioinformatics/augustus/3.3
module load bioinformatics/busco/3.0
module load bioinformatics/bioperl/1.6.924
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo + NSLOTS = $NSLOTS
#
export AUGUSTUS_CONFIG_PATH="/path/to/your/workspace/config"
run_BUSCO.py -i your_genome.fa -o your_genome -l /path/to/your/workspace/your_database -m geno --cpu 12 --long -
sp chicken
#
echo = `date` job $JOB_NAME done
```

GeneMark-ES

GeneMark-ES is a eukaryotic gene prediction software. In order to run it, the user must copy the license from the software install into their home directory as `.gm_key` in order to run the software. You can do this with:

```
$ cp gm_key_64 ~/.gm_key
```

MaSuRCA

MaSuRCA is an assembly algorithm that relies on creating "super-reads" then assembling them with other assemblers such as Celera (an overlap, layout, consensus assembler). It also has the ability to create "hybrid" assemblies using both PacBio and Illumina data.

This is meant to be a cookbook example of how to run MaSuRCA on Hydra, not a comprehensive guide.

To generate a MaSuRCA assembly, you will need to follow several steps.

1. Load your data (FASTQ files for Illumina, FASTA for PacBio) into either your `/pool` or `/scratch` directory (see Transferring files to or from Hydra for more info).
2. Generate a sample MaSuRCA config file. You can do this by entering the following commands into your terminal:
   ```
   module load bioinformatics/masurca
   masurca -g masurca.config
   ```
3. Now you will edit the `masurca.config` so that it includes the paths to your sequence files and values for the program parameters.
   a. DATA:
      i. You need to enter data for the different options given in the config file. E.g. 'PE' stands for 'Paired-end' and 'JUMP' stands for mate-pairs or jumping libraries. You'll see below that the PacBio line is commented out. If you have PacBio data, remove the #. Likewise, if you don't have Jumping libraries, add a # to the start of those lines. Make sure that you use full file paths for your read locations. A good way to get this right is to navigate to the directory that holds your files, use the command `pwd`, copy and paste the path, then add a `/` and copy and paste the filename. If your paths are complicated, this is a good way to avoid typos.
      ii. When you give names to the paired-end or mate pair libraries, make sure that you use a unique two letter code for each separate library. It also asks for both the average read length and the standard deviation for each library. Keep in mind that Illumina reads should be in FASTQ format and PacBio reads should be in FASTA format. Here is an example of how that might look:

      Example library config

      ```
      DATA

      #Illumina paired end reads supplied as <two-character prefix> <fragment mean> <fragment stdev> <forward_reads> <reverse_reads>
      #if single-end, do not specify <reverse_reads>
      #MUST HAVE Illumina paired end reads to use MaSuRCA
      PE= pe 500 50  /FULL_PATH/frag_1.fastq  /FULL_PATH/frag_2.fastq
      #Illumina mate pair reads supplied as <two-character prefix> <fragment mean> <fragment stdev> <forward_reads> <reverse_reads>
      JUMP= sh 3600 200  /FULL_PATH/short_1.fastq  /FULL_PATH/short_2.fastq
      #pacbio OR nanopore reads must be in a single fasta or fastq file with absolute path, can be gzipped
      #if you have both types of reads supply them both as NANOPORE type
      #PACBIO=/FULL_PATH/pacbio.fa
      #NANOPORE=/FULL_PATH/nanopore.fa
      #Other reads (Sanger, 454, etc) one frg file, concatenate your frg files into one if you have many
      #OTHER=/FULL_PATH/file.frg
      END
      ```

   b. PARAMATERS
      i. Most parameters have comments (lines that start with '#') that will guide you in selecting the appropriate values for each. Do not change the USE_GRID option - we do not have MaSuRCA set up to run with that option on Hydra. Make sure that the number of threads you put for the NUM_THREADS option is reflected in your job file.

```
PARAMETERS
#set this to 1 if your Illumina jumping library reads are shorter than 100bp
EXTEND_JUMP_READS=0
#this is k-mer size for deBruijn graph values between 25 and 127 are supported, auto will compute the optimal
size based on the read data and GC content
GRAPH_KMER_SIZE = auto
#set this to 1 for all Illumina-only assemblies
#set this to 0 if you have more than 15x coverage by long reads (Pacbio or Nanopore) or any other long reads
/mate pairs (Illumina MP, Sanger, 454, etc)
USE_LINKING_MATES = 0
#specifies whether to run mega-reads correction on the grid
USE_GRID=0
#specifies queue to use when running on the grid MANDATORY
GRID_QUEUE=all.q
#batch size in the amount of long read sequence for each batch on the grid
GRID_BATCH_SIZE=300000000
#use at most this much coverage by the longest Pacbio or Nanopore reads, discard the rest of the reads
LHE_COVERAGE=25
#set to 1 to only do one pass of mega-reads, for faster but worse quality assembly
MEGA_READS_ONE_PASS=0
#this parameter is useful if you have too many Illumina jumping library mates. Typically set it to 60 for
bacteria and 300 for the other organisms
LIMIT_JUMP_COVERAGE = 300
#these are the additional parameters to Celera Assembler.  do not worry about performance, number or processors
or batch sizes -- these are computed automatically.
#set cgwErrorRate=0.25 for bacteria and 0.1<=cgwErrorRate<=0.15 for other organisms.
CA_PARAMETERS =  cgwErrorRate=0.15
#minimum count k-mers used in error correction 1 means all k-mers are used.  one can increase to 2 if Illumina
coverage >100
KMER_COUNT_THRESHOLD = 1
#whether to attempt to close gaps in scaffolds with Illumina data
CLOSE_GAPS=1
#auto-detected number of cpus to use
NUM_THREADS = 16
#this is mandatory jellyfish hash size -- a safe value is estimated_genome_size*estimated_coverage
JF_SIZE = 200000000
#set this to 1 to use SOAPdenovo contigging/scaffolding module.  Assembly will be worse but will run faster.
Useful for very large (>5Gbp) genomes from Illumina-only data
SOAP_ASSEMBLY=0}
```

MaSuRCA runs with 2 job files. The first uses your .config file to generate an sh script called assemble.sh. Then you just execute the sh script to complete the actual assembly.

- Sample parameters for a job file for the first part of MaSuRCA.
  - Queue: sThC.q
  - Threads & RAM: single thread, 2GB RAM
  - Module: `module load bioinformatics/masurca`
  - Commands: `masurca <CONFIG_FILE>`
- This job should complete in a few seconds and result in a file called assemble.sh
- Create a second job file for the second part of MaSuRCA.
  - Queue: lThM.q or uTxlM.rq (the latter is restricted to users that request access from SI-HPC@si.edu)
  - Threads & RAM: 16 threads, 30GB RAM each (RAM usage will depend on the genome size - you are able to select up to 512GB on the lThM.q. If you need more than that, please request access to uTxlM.rq)
  - Module:`module load gcc/4.9.2`
  - Command: `./assemble.sh`
- Submit this second job. It can take a very long time depending on genome size and coverage.

SPAdes

The SPAdes assembler is often used for microbial and organelle genome assembly as well as target enrichment methods such as UCE or exon capture.

Specifying memory

the `-m <int>` option for `spades.py` should be used to specify the amount of memory (in GB) you have reserved in your `qsub` submission (`mres`). The default value of `-m` if you do not specify a value is 250. By specifying a value for `-m` that matches what you reserved, SPAdes will adjust buffer size usage to reduce RAM usage.

SPAdes temporary files

`spades.py` can create many temporary files, more that one million in some cases. By default, the temporary files are placed in a `tmp` directory in the output directory you specify with `-o`. The large number of files can cause you to exceed your inode quota or adversely affect the operation of the GPFS (`/scratch` et al.) and NetApp (`/pool` et al.).

To avoid this, it is recommended to use the local SSD space for the SPAdes temporary files.

This examples requests 200GB of local SSD from the scheduler (`-l ssd_res=200G`) and then uses this for the `spades.py` tmp directory by adding `--tmp-dir $SSD_DIR`

The addition of `-v SSD_SAVE_MAX=0` directs the job scheduler to not save any of the files from the SSD storage space. The temporary files `spades.py` creates do not need to be retained.

Example spades.py script using local SSD

```
# /bin/sh
# ----------------Parameters---------------------- #
#$ -S /bin/sh
#$ -pe mthread 8
#$ -q lThM.q
#$ -l mres=96G,h_data=12G,h_vmem=12G,himem
#$ -l ssd_res=200G -v SSD_SAVE_MAX=0
#$ -cwd
#$ -j y
#$ -N spades
#$ -o spades.log
#
# ----------------Modules------------------------- #
 module load bio/spades
 module load tools/ssd
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo + NSLOTS = $NSLOTS

spades.py  \
  -o sample \
  --pe1-1 sample_R1_PE_trimmed.fastq.gz \
  --pe1-2 sample_R2_PE_trimmed.fastq.gz \
  --tmp-dir $SSD_DIR \
  -m 96 \
  -t $NSLOTS

#
echo = `date` job $JOB_NAME done
```

Last updated 17 Jan 2024 MPK

Bioinformatics Software and Modules

- As of May 2024, about 100 bioinformatics software packages have been installed on Hydra-7 , a list we had to prune down from the 200+ that were over the years installed on Hydra-6, and 8 new packages were added,
- As we reorganized things, we also moved some general use packages from `bio/` to `tools/`
- The list of packages that we pruned is also down this page, here.

💡 ***Job files will need to be adjusted to reflect these changes.***

- Users looking for software not on this list can compile/install in their own space or request installation by emailing SI-HPC@si.edu.
- Note that with our limited time, we prioritize installing software that many users need and is well-documented.
- Please contact SI-HPC@si.edu if you notice any issues with bioinformatics software and modules.

General Packages Moved from bio/ to tools/

- The following packages had their modules under bio/ and have been moved to tools/:

```
awscli, ffsend, julia, rclone, ruby
```

New Packages in bio/

- The following packages are now available on Hydra-7, they were not available on Hydra-6:

```
admixtools, basespace, blobtoolkit, mitos, paup, quast, raxml-ng, rsem
```

Pruned Packages

- The following packages, that were available on hydra-6, *are no longer* available on Hydra-7:

```
anuga, aster, atram, autoparts, bayesass, beagle_5.1, blasr, blast_taxa_backfill, blat, centrifuge,
comp_popgen_sjsw, delimitr, discovista, dna-algorithm, dotnet-core, edirect, edta, eems, ervin, ete3, exabayes,
fatotwobit, fsct, gadma, gatb-minia-pipeline, gblocks, genomescope, gnuparallel, goleft, gsutil, haystac,
hifiadapterfilt, hmmer, hybphylomaker, hyde, hyphy, ibpp, imagemagick, insector, ivis, jags, jbrowse, kakapo,
makehub, manta, mauve, mcscanx, megadetector, metamaps, metamorpheus, metapop2, mitogeneextractor, mitohifi,
ml_derkarabetian, mpboot, mptp, msmc2, multiqc, ngsadmix, ngstools, opt-sne, orthomcl, paml, partitionfinder,
patchwork, pbsuite, pcangsd, phast, phrapl, phylobayes, phylobayes-mpi, phyloflash, phylomad, phylonet, phyparts,
phyx, pilon, prokka, proteinortho, pyrad, ragout, relernn, rjags, rsem-eval, salmon, salsa, seq-seq-pan, shapeit,
sharkmer, smc++, spruceup, sspace, stampy, star, strauto, stringtie, supernova, tabix, transrate_1.0.3,
trinotate, vcflib, velvet, viridal, w2rap-contigger, wgd
```

- ⚠️ If you need one of these tools, contact us at `si-hpc@si.edu`

Complete List of `bio/` Packages

- Here is the full list of `bio` packages available on Hydra-7 as we migrated to Rocky 8 (May 2024):

⚠️ **For many packages the default version has changed. In some cases. Please review your job files to ensure that the intended version is being used.** ⚠️

Module	**Version(s) on Hydra-7** ***"(def)" denotes default version***	**Version(s) on Hydra-6** ***"(def)" denotes default version***	**Change in default version?**	**Note**
`abyss`	2.3.7(def)	2.2.4(def)	yes	
`admixtools`	7.0.2(def)			
`admixture`	1.3.0(def)	1.3.0(def)		
`angsd`	0.941(def)	0.921, 0.931, 0.937(def), 0.94	yes	
`assembly_stats`	0.1.4(def)	0_1_3(def)	yes	
`astral`	5.7.8(def), 5.15.5-MP	5.7.8(def), mp-5.15.5		
`augustus`	3.5.0(def)	3.3.2(def)	yes	

basespace	1.5.4(def)			
bayescan	2.1(def)	2.1(def)		
bbmap	39.06(def)	38.67(def)	yes	
bcftools	1.19(def)	1.9(def)	yes	
beagle	4.0.1(def)	3.2(def), 4.0.1	yes	
beast	2.6.7, 2.7.6(def)	1.10.4, 2.6.0, 2.6.0-no-beagle, 2.6.1, 2.6.3, 2.6.6, 2.6.7(def), 2.7.5-no-beagle, 2.7.6-beagle	yes	
bedtools	2.31.1(def)	2.28.0(def)	yes	
bioawk	1.0(def)	1.0(def)		
bioperl	1.7.8(def)	1.7.2(def)	yes	
biopython	1.83(def)	1.74(def), 1.74_python2.7	yes	
blast	2.15.0(def)	2.6.0, 2.6.0_old_db, 2.8.1, 2.8.1-precompiled, 2.9.0, 2.9.0_v5, 2.10.1(def), 2.13.0	yes	
blast2go	To be installed	1.4.4, 1.5.1(def)		⚠️ This will be installed after the deployment of Hydra77
blobtoolkit	3.3.10(def)			
blobtools	1.1.1(def)	1.0.1(def), 2, 2.6.3	yes	This is for blobtools 1.x, later versions are in the module bio/blobtoolkit
bowtie2	2.5.3(def)	2.3.5(def), 2.5.1	yes	
bpp	4.7.0(def)	1, 4.6.1(def)	yes	
busco	5.7.0(def)	3.0.2(def), 4.0.2, 5.1.3, 5.2.2, 5.3.2, 5.4.3	yes	
bwa	0.7.17(def)	0.7.17(def)		
cactus	2.8.0(def)	2019.03.01, 2020(def), cactus_test	yes	
canu	2.2(def)	1.8(def), 2.1.1	yes	
cd-hit	4.8.1(def)	4.8.1	yes	
cutadapt	4.7(def)	2.4(def)	yes	
dates	753(def)	753(def)		
diamond	2.1.9(def)	1.0(def)	yes	
exonerate	2.4.0(def)	2.4.0(def)		
fastp	0.23.4(def)	0.23.4	yes	
fastqc	0.12.1(def)	0.11.8(def)	yes	
faststructure	1.0(def)	1.0(def)		
fasttree	2.1.11(def)	2.1.11(def)		
gatk	3.8.1.0, 4.5.0.0(def)	3.8.1.0(def), 4.1.3.0	yes	
gemoma	1.9(def)	1.0(def), 1.7.1, 1.9	yes	

getorganelle	1.7.7.0(def)	1.6.2d(def), 1.7.5	yes	
guppy	6.5.7(def)	5.0.16(def)	yes	
hifiasm	0.19.8(def)	0.14.2(def), 0.16.1	yes	
hisat2	2.2.1(def)	2.2.1(def)		
htslib	1.19.1(def)	1.9(def)	yes	
hybpiper	1.3.1_final, 2.1.6(def)	1.3.1(def), 2.0.1, 2.1.6	yes	
illumiprocessor	2.10(def)	2.0.9_trimgalore, 2.10_trimgalore, 2.10_trimgalore_2023-02-17(def)	yes	
ipyrad	0.9.94(def)	0.7.30(def), 0.9, 0.9.74	yes	
iqtree	1.6.12, 2.3.1(def)	1.6.12, 2.0.4, 2.0.6, 2.1.1, 2.1.2, 2.1.3(def), 2.rc.1	yes	
jellyfish	2.3.1(def)	2.3.0(def)	yes	
kraken	2.1.3(def)	2.0(def), 2.1.2	yes	
krakenuniq	1.0.4(def)	1.0.1(def)	yes	
mafft	7.525(def)	7.407(def)	yes	
maker	3.01.03(def)	2.31.10(def)	yes	
mapdamage2	2.2.2(def)	2.2.1(def)	yes	
masurca	4.1.1(def)	3.2.2, 3.3.3, 3.3.3_test, 3.3.8(def), 3.3.9, 4.0.0, 4.0.9, 4.1.0	yes	
merqury	1.3(def)	1.3(def)		
migrate	3.7.2, 5.0.6(def)	3.6.11, 3.7.2, 4.4.4		
minimap2	2.28(def)	2.24(def)	yes	
mitofinder	1.4.1(def)	1.2(def), 1.4	yes	
mitos	2.1.8(def)			
mitoz	3.6(def)	3.5(def)	yes	
mrbayes	3.2.7a(def)	3.2.7_beagle, 3.2.7a(def)	yes	
nucmer	3.2.3, 4.0.0rc1(def)	3.23(def)	yes	
oma	2.6.0(def)	2.3.1, 2.4.1, 2.4.2		
orthofinder	2.5.5(def)	2.0.9, 2.5.4		
paup	4a168(def)			
phyluce	1.6.8, 1.7.3(def)	1.5_tg, 1.6.7(def), 1.7.0, 1.7.1	yes	
picard-tools	2.20.6, 3.1.1(def)	2.20.6(def)	yes	
plink	1.90b7.2(def)	1.90b4(def)	yes	

psmc	0.6.5(def)	0.6.5(def)		
purge_dups	1.2.6(def)	1.0(def)	yes	
qiime2	2024.2-amplicon(def), 2024.2-shotgun	2019.7, 2019.1, 2020.8, 2021.11(def)	yes	qiime2 now has separate amplicon and shotgun sequencing packages
quast	5.2.0(def)			
R	4.3.3(def)	3.5.2_rgdal, 3.6.0_conda, 3.6.1(def)	yes	
raxml	8.2.13(def)	8.2.12(def), 8.2.12-mpi, ng-0.9.0	yes	
raxml-ng	1.2.1(def)			
repeatmasker	4.1.5, 4.1.6(def)	4.0.9(def)	yes	
repeatmodeler	2.0.5(def)	1.74(def)	yes	
revbayes	1.2.2(def)	1.0.11(def), 1.0.13	yes	
rohan	1.0.1(def)	1.0.1(def)		
rsem	1.3.3(def)			
samtools	1.19.2(def)	1.9, 1.17(def)	yes	
seqkit	2.8.1(def)	0.13.2, 2.3.1(def)	yes	
seqtk	1.4(def)	1.0(def)	yes	
spades	3.15.5(def)	3.11.1, 3.12.0, 3.14.0(def), 3.15.5	yes	
sratoolkit	3.1.0(def)	2.9.6, 2.11.0(def)	yes	
stacks	2.66(def)	1.48, 2.6.5, 2.41(def)	yes	
structure	2.3.4(def)	2.3.4(def)		
transabyss	2.0.1(def)	2.0.1(def)		
transdecoder	5.7.1(def)	5.5.0(def)	yes	
transrate	1.0.3	1.0.3, 1.0.3.2, 1.0.3_mamba, 1.0.3_orp		
trim_galore	0.6.10(def)	0.6.4(def)	yes	
trimmomatic	0.39(def)	0.39(def)		
trinity	2.15.1(def)	2.8.5(def), 2.9.0, 2.9.1, 2.13.2, r2013_2_25	yes	
vcftools	0.1.16(def)	0.1.16(def)		
wtdbg2	2.5(def)	2.5(def)		

Last updated 17 Apr 2024 MPK

Additional Tools

1. Introduction

- We offer a set of tools, written for Hydra, to help monitor jobs and the cluster and do some simple operations.
    - These are split into `tools/local-user` and `tools/local-admin`;
    - The modules `uge/8.8.1` & `tools/local-user` are 'sticky' and this always loaded.
    - The module `tools/local` is available for backward compatibility and loads `tools/local-user` and `tools/local-admin`;
    - The module `tools/local+` gives access to more local tools.
- The commands

```
% module help tools/local-user
```

```
% module help tools/local-admin
```

and

```
% module help tools/local+
```

list the available tools (see table below).

- Each tool has a man page.
- A few of these tools are describe here in more details.
- The module `tools/local-bc` offers backward compatibility (hence the -bc) w/ the local tools on Hydra-4 and will be eventually phased out.

2. A "better" qstat: qstat+

- `qstat+` is a PERL wrapper around `qstat`: it runs `qstat` for you and parses its output to display it in a more friendly format.
    - `q+` is a shorthand for `qstat+.`
- How to use `qstat+` is explained in the man page (`man qstat+`) and is described by:

```
% qstat+ -help
```

or

```
% qstat+ -examples
```

- It can be used to
    - get useful info like the age and/or the cpu usage (in % of the job age) of running jobs,
    - get the list of nodes a parallel job is running on (if you haven't saved it),
    - get a filtered version of `qstat -j <jobid>`, and
    - get an overview of the cluster queues usage.

Namely:

qstat+ -help

```
usage: qstat+ [mode] [options]
  where modes are (exclusive list):
    -s|-sx|-sc      show summary (simple, extended or compact)
    -a              show all jobs
    -q              show queued jobs
    -r              show running jobs
    -X              show extra jobs (dr, Eqw, t)
    -j      JID     show info on a specific job
    -nlist  JID     show node list for a (set of) job(s)
    -hiload N       show node(s) with high load
    -gc             show global cluster status
    -es[x]          show empty slots, -esx: expanded info
    -down           show node(s) that are down
    -ores           show overreserved jobs, i.e.: resMem/maxVMem > 2.5, & age > 1 hr
    -osub           show oversubscribed jobs, i.e.: cpu% > 133%, & age > 1 hr
    -ineff          show inefficient jobs, i.e.: cpu% < 33%, & age > 1 hr
    -ssd            show SSD usage
    -h|-help--help  show help
    -examples       show examples

where options are:
    -u USER       limit to the specified user, you can use "*" or "all" to list everybody's jobs
                  or a coma-separated list
    -njobs        show counts in no. of jobs
    -npes         show counts in no. of PEs (slots)
    -nqpe         show jobs/PEs not jobs/tasks for queued jobs
    -nqxx         show jobs/tasks/PEs for queued jobs
    -load         show the nodes' load
    -sua          show the nodes' used/avail no of slots
    -age          show the age of the jobs, i.e., elapsed time vs starting/submit time
    -cpu          show the amount of CPU used by running job(s)
    -cpu%         show the amount of CPU/age/#PE in % (job efficiency)
    -cpur         show the ratio CPU/AGE, not scaled by #PE
    -mem          show the mean memory usage for running jobs, the total requested memory for queued jobs, in GB
    -memx         show more memory info for running jobs: reserved, mean, vmem and maxvmem (slow)
    -memr         show more memory info for running jobs: reserved, mean, maxvmem and res/mxvmem (slow)
    -io           show the I/O usage
    -iow          show the I/O wait usage (slow)
    -ioops        show the IOPs usage (slow)
    +lic          show license info (default), although not in detailed outputs
    -lic          do not show license info

    -noheader     do not show the header
    -nofooter     do not show the footer
    -raw          print raw values (for easier parsing)
    -queue QSPEC  limit to jobs in queue QSPEC (RE ok)
    -oresF  VAL   change the overreserved factor to VAL
    -osubT  VAL   change the oversubscribed threshold to VAL, in %
    -osubE  VAL   set the oversubscribtion excess threshold to VAL, in hr x slots
    -ineffT VAL   change the inefficient threshold to VAL, in %
    -ageT   VAL   change the minimum age threshold to VAL, in hour

    -v|-verbose   set verbose mode
    -warn         show warnings
    -wide         wide output (132 cols, implies -notty)
    -notty        do not use the width of the terminal, as returned by stty
    -check-load   check the instantanous load (-hiload only)

shorthands:
    +a   is expanded to -a -u $USER -nofooter             show all your jobs
    +a%                -a -u $USER -nofooter -age -cpu%  ibidem, with cpu on % of age
    +ax%               -a -u $USER -nofooter -age -cpu% -load -sua -mem -io
    +ar%               -a -u $USER -nofooter -age -cpu% -memr -io
    +r                 -r -u $USER -nofooter             show all your running jobs
    +r%                -r -u $USER -nofooter -age -cpu%  ibidem, with cpu on % of age
    +rr%               -r -u $USER -nofooter -age -cpu% -memr -io -iow -ioops
    +rx%               -r -u $USER -nofooter -age -cpu% -load -sua -memx -io -iow -ioops
    +q                 -q -u $USER -nofooter             show all your queued jobs
    +q%                -q -u $USER -nofooter -age        ibidem but show age
    +qx%               -q -u $USER -nofooter -age -mem   ibidem plus mem info
    +X                 -X -u $USER -nofooter             show all your extra jobs
    +n                 -notty -wide -noheader -nofooter

qstat+: Ver 5.3/4 - May 2024
```

qstat+ -examples

```
examples:
 qstat+ -a -u hpc                   show all of hpc's jobs
 qstat+ -r -cpu% -u hpc             show all of hpc's running jobs, cpu in %
 qstat+ -r -cpu% -load -sua -u hpc  show all of hpc's running jobs, cpu in %, and
                                the nodes' load and slot usage/availability
 qstat+ -q                          show all the queued jobs
 qstat+ -j 8683280,8683285          show info on specific job IDs
 qstat+ -nlist 8683280,8683285      show nodes list for specific job IDs
 qstat+ -hiload 1.5                 show nodes whose load is 1.5 greater than the number of slots used
 qstat+ -ineffT 50 -ineff           show jobs that are below a 50% efficiency threshold
 qstat+ -osubT 200 -ageT 48 -osub   show jobs that are above a 200% usage threshold and are older than 48 hours
 qstat+ -gc                         show the global cluster status
 qstat+ -es                         show the empty slots
 qstat+ -down                       show which nodes are down

        +ax% -u all                 is equiv to -a -u all -cpu% -load -sua -mem
                             etc... for the +XXX shorthands

qstat+: Ver 5.3/4 - May 2024
```

3. A "better" qacct: qacct+

- The accounting information is ingested into a SQL compatible data base by the GridEngine (aka ARCo - for Accounting and Reporting Console)
    - The ingestion is done at regular intervals, hence the database is not instantaneously synchronized (less than a minute lag usually).
    - As of Hydra-7, we use GE's dbwriter (aka ARCo) and PostgreSQL as the DB engine
        - although we discovered some bugs under 8.8.1, where some fields are not properly ingested.
- That database can be queried with `qacct+`, hence
    - `qacct+` is often faster than `qacct`
    - `qacct+` its output can be customized, and
    - `qacct+` computes derived values.
- How to use `qacct+` is explained in the man page (`man qacct+`) and is described by:

```
   % qacct+ -help
```

or

```
   % qacct+ -show help
```

Namely:

qacct+ -help

```
usage: qacct+ [options] where options are

                   to limit query to
  -j|--job_number  <jobid>    given job  ID(s),    single value, comma separated list, or range like in n:m
  -t|--task_number <taskid>   given task ID(s),    single value, comma separated list, or range like in n:m
  -o|--owner       <owner>    given owner,         exact match

  -b|--back        <number>   jobs submitted  >= (now - n) days,  (def. 92 days ago)
  -s|--since       <date>     jobs submitted  >= date,   like "4/27/2016 10:00AM"
  -u|--until       <date>     jobs submitted  <= date,   like "4/27/2016 11:00AM"

                   to specify what to show
  -show            <string>                                       (def.: -show simple)
                   where <string> can be:
                     "simple", "simple+", "tab", "tab+", or "raw",
                   or "fields", "explain", or "help", or a custom specification

                   to limit output to
  -m|--show_max    <number>   max number of entries to show (when querying more than one job)

  -w|--col_width   <number>   width of columns in tabular mode   (def.: 15)
  -d|--date_format <number>   how to format dates, n=-1,0,1 for YYYY/MM/DD, Mon DD YYY, MM/DD/YYYY, (def. 0)

  -n|--dry_run                dry run, use w/ -v to check the resulting SQL search
  -v[=value]                  verbose mode (repeat to increase verbosity, -v=2 equiv to -v -v)
  -i|--init         <file>    mysql initialization file(s),      (def.: /share/apps/tools/local-user/.qacct+.
cnf, /home/hpc/.my.cnf)

  -h|--help                   show this help
  use -show help to get additional help on how to use -show <string>

  Ver. 5.1/0 (Mar 2024/SGK)
```

- and by

```
% qacct+ -show help
```

qacct+ -show help

```
  -show  <string> specify what to show

  i.e.
    -show simple  for simple format (default)
    -show simple+ for extension of the simple format
    -show tab     for tabular format
    -show tab+    for extension of the tabular format
    -show raw     for raw format

  or a list of which keyword to show, with an optional format, where <string> is either
    +field1[=format1][,field2[=format2]]   for keyed format
  or
    %field1[=format1][,field2[=format2]]  for tabular format

  the format is either a C format specification like %d, %s, %.1f, or
  the values @DATE, @MEM, or @AGE to convert the numeric to a date, a memory (or volume) or an age

  like in:
    -show +qname,slots,wallclock=@AGE,cpu=%.1f
  or
    -show %qname,slots,wallclock,cpu=%-15.1f,granted_pe

  use -show fields           to list all the available fields
  use -show explain_failed   to list the failed codes explanations
```

4. A better "quota": quota+

- The Linux command `quota` is not working on the GPFS (/scrarch) or the NAS (/store).
- `quota+` will report disk quota information on all the disks (NFS, GPFS or NAS)

quota+ help

```
quota+ [options]
  where options are:
   -u|--user user          return quotas for given user (must be root)
   -v|--verbose            display quotas on filesystems where no storage is allocated
   -a|--all                display quotas on filesystems that are not mounted
   -%                      show Use% instead of Used
   +%                      show Use% as well as Used
   -f filesys              return quotas for given file system only
   -device                 show device name only
   +device                 show mount point and device name
   -terse                  show only Used/Use% and Quota Limit, disable +%

Ver 2.6/1 May 2024
```

5. Checking a Compute Node: rtop+

`rtop+`: a script to run the command `top` on a compute node (aka remote `top`.)

- The Un*x command `top` can be used to look at what processes are running on a given machine (it reports the "top" processes running at any time).
- To check what processes are running on a compute node, (to check CPU and/or memory usage, you can use:
  ```
     % rtop+ [-u <username>] [-<number>] NN-MM
  ```
  like in
  ```
     % rtop+ -u hpc -50 43-05
  ```

  and you will see the `<number>` lines listing the processes owned by `<username>` on the compute node `compute-NN-MM`, or (second example) the first 50 lines when running `top`, limited to user `hpc`, on `compute-43-50`.
  If you omit `-<number>` you will see only the first 10 lines, if you omit `-u <username>` you will see everybody's processes.

Check `man top` to better understand the output of `top`.

6. Checking Memory and CPU Usage of Jobs in the High Memory Queues

- `plot-qmemuse:` a tool to plot the memory and CPU usage of jobs that ran recently or are running in the high memory queues.
- We monitor the jobs running in the high-memory queue, taking a usage snapshot every five minutes. This tool only applies to jobs in the high-memory queues
- The resulting statistics can be used to visualize the resources usage of a given job with the command `plot-qmemuse`, using

```
  % plot-qmemuse <jobid>
```

or

```
  % plot-qmemuse <jobid>.<taskid>
```

For that command to run, you must load the `gnuplot module` first. By default this tool produces a plot in a 850x850 pixels `png` file.

You can specify the following options:

`-l <label>`	to add your own label on the plot
`-s <size>`	to specify the plot size, in pixel (-s 1200 for a 1200x1200 plot)
`-o <filename>`	to specify the name of the `png` file
`-x`	to plot on the screen (using `X11`, assuming your connection to hydra allows `X11`)

You can view the plot in the `png` file with the command `display <filename>`, assuming that your connection to hydra allows `X11`, or

you can copy that file to your local machine and view it with your browser or a png-compatible image viewer (like `xv`).

7. List of Local Tools

The following tools are always available since the module `tools/local-user` is *always* loaded:

`check-qwait`	show job(s) waiting in the queue and associated queue quota limits	
`finger`	replacement for CentOS7 `finger`	use `pinky -l`
`get-gpu-info`	print information about GPU	
`monitor-code-usage`	helps you monitor your code usage	
`plot-qmemuse`	plot memory and cpu usage as a function of time for a given jobID	
`plot-qssduse`	plot SSD usage as a function of time for a given jobID	
`qacct+`	a "better" `qacct`	show accounting information for completed jobs
`qchain`	chains a set of jobs by adding the -hold_jid <jobID> to qsub for you	
`qquota+`	a "better" `qquota`	show queue quota wrt queue limits
`qstat+`	a "better" `qstat`	show queue status
`quota+`	a "better" `quota`	show disk quota information for all type (NFS, GPFS, NAS)
`q-wait`	wait until some jobs are not found in the queue	
`rtop+`	remote `top`	
`ruptime`	replacement for `ruptime`	limited to head, login and NSDs
`rwho`	replacement for `rwho`	limited to head, login and NSDs
`show-qmemuse`	show memory use statistics	
`show-qslots`	shows how many slots are available in the queue(s)	
`show-qssduse`	log statistics for `plot-qssduse`	

The following tools are available when loading the module `tools/local-admin:`

`chage+`	substitute to `chage`, to query LDAP properties	ie: `chage+ $USER`
`check-disks-usage`	check disks usage, and print warning when usage exceed a threshold	ie: `check-disks-usage -w 10`
`check-gpuse`	print the cluster GPUs usage, for the hosts in the GPU host group	
`check-hi-memuse`	check for memory use versus reservation and CPU usage efficiency	
`check-memres`	check for jobs that use less memory than the amount reserved	
`check-memuse`	print the cluster usage (memory and CPUs)	
`check-qlogs`	return a report on either oversubscribed or inefficient jobs	
`disk-usage`	return information on disk usage	
`find-all-zombies`	find zombies	
`get-cpu_arch`	return CPU architecture	
`parse-disk-quota-reports`	return report by parsing the disk quota report	
`plot-disk-dev`	plot usage of a given device	
`plot-disk-usage`	plot the usage of a given disk (volume)	
`plot-gpfs-info`	plot GPFS information	
`plot-ibtraffic`	plot IB traffic	
`plot-qsnapshot`	plot a snapshot of cluster usage	
`plot-qssduse-summary`	plot SSD usage summary as a function of time	

plot-qstat	plot the queue status of the cluster	
plot-ruptime	plot the load (output of ruptime) of the non-compute nodes	
qhost+	a "better" qhost	
rkill	remote kill	
rkillall	remote killall	

Each tool has a man page, accessible after you load the module tools/local.

8. List of Local+ Tools

The following tools are available when loading the module tools/local+;

backup	backup a file	ie: mv or cp file to file.<n>
centos-version	print the OS flavor & version	
check-hosts	print the cluster usage (memory and CPUs), as aggregate by logical rack	
check-qacct	show statistics of resources usage for completed jobs	
dus-report	run and parse du to produce a disk usage report	
elapsed	print elapsed time between each call	ie; elapsed; ...do something...; elapsed
fixFmt	format a number with fixed number of digits	csh: @ n = 1; set N = `fixFmt 3 $n` [ba]sh: n=1; N=`fixFmt 3 $n` echo n=$n N=$N n=1 N=001
get-jobhr	tool to retrieve a job hard resources (mem_res h_data h_vmem) in a job script	
lsth	show most recent files	ie: lsth [-40] [<spec>] ls -lt <spec> | head -40
lswc	count number of files in directories	ie: lswc dir/ ls dir/ | wc -l
noX	tool to unset DISPLAY (saved in XDISPLAY)	
p-wait	wait until given PID has completed	ie: p-wait [check-time] <PID>
pawk	print with awk	ie: pawk 1,hello,3 awk '{print $1," hello",$3}'
print-proc-memory	print nicely content of /proc/memory	
procinfo	print properties of local machine (#CPUs, memory, OS)	
procinfo+	print properties of local machine (#CPUs, memory, OS, etc...)	
tails	run tail [options] on a set of files	ie: "tail -3 *pl" fails "tails -3 *pl" OK
total	compute the total of values at given column of a file	
useX	tool to reset DISPLAY (from XDISPLAY), revert effect of noX	
xterm-config	tool to configure xterm window properties	

Each tool has a man page, accessible after you load the module tools/local+.

Last Updated 16 May 2024 SGK.

How to Use GPUs

1. Introduction

A total of 4 GPUs on 2 nodes are available on Hydra.

- One node has four GPU cards (NVIDIA L40S)
- Two nodes have two GPU cards (NVIDIA GV100)
    - each GPU corresponds to:

Type	CUDA Cores	Memory	Mem b/w
L40S	18,176	48GB	864 GB/s
GV100	5,120	32GB	870 GB/s

note that CUDA cores are not like CPU cores.

2. GPU Configuration

- The GPUs are configured as follow:
    - The current driver is CUDA 12.4 (Driver Version 550.54.15)
    - persistence mode is ENABLED (NVIDIA driver remains loaded even when there are no active clients)
    - compute mode is set to EXCLUSIVE_PROCESS (only one context is allowed per device, usable from multiple threads at a time)
    - ⚠ This configuration is reset at reboots.
      (see `man nvidia-smi` - accessible on the gpu nodes.)

💡 This means that GPU applications will

- start faster (no need to re-load the driver), and
- only one process can use a GPU at a time (exclusive use.)

⚠ Only one process per GPU can run at the same time, each process gets a different GPU.

⚠ Starting one more process than available GPUs will fail with the following error message:

```
Error: all CUDA-capable devices are busy or unavailable
 in file XXX.cu at line no NNN
```

3. Available Tools

CUDA

- The CUDA compiler is now part of the NVIDIA compilers, and is accessible loading the NVIDIA module

  ```
  % module load nvidia
  ```

  that loads by default NVIDIA 23.9 and CUDA 12.2.

- Other version are available, check with

  ```
  % module whatis nvidia
  ```

NVSMI: The NVIDIA System Management Interface

- NVSMI v550.54.15 and dcgm v 3.35 are available on the GPU nodes.
- The following tools are available, but only on the nodes with GPUs, and by loading the `cuda10.2` module:

nvidia-cuda-mps-control	NVIDIA CUDA Multi Process Service management program
nvidia-modprobe	Load the NVIDIA kernel module and create NVIDIA character device files
nvidia-persistenced	A daemon to maintain persistent software state in the NVIDIA driver
nvidia-settings	configure the NVIDIA graphics driver
nvidia-smi	NVIDIA System Management Interface program
nvidia-xconfig	manipulate X configuration files for the NVIDIA driver

- Read the corresponding man pages to learn more, load the `cuda-dcgm` module to access `dcgm`.
- The man pages are accessible on the GPU nodes.

Query and Monitor

- The most useful tool is `nvidia-smi`, it allows you to query & monitor the status of the GPU card(s):

Try one of the following commands:

```
hpc@compute-79-01% nvidia-smi -l
hpc@compute-79-01% nvidia-smi dmon -d 30 -s pucm -o DT
hpc@compute-79-01% nvidia-smi pmon -d 10 -s um -o DT
hpc@compute-79-01% nvidia-smi \
         --query-compute-apps=timestamp,gpu_uuid,pid,name,used_memory \
         --format=csv,nounits -l 15
hpc@compute-79-01% nvidia-smi \
         --query-gpu=name,serial,index,memory.used,utilization.gpu,utilization.memory \
         --format=csv,nounits -l 15
hpc@compute-79-01% nvidia-smi -q -i 0 --display=MEMORY,UTILIZATION,PIDS
```

- Read the man page (`man nvidia-smi`)

~~GDK/NVML/pynvml~~

~~The GPU development kit (GDK), NVIDIA Management Library (NVML) and the python bindings to NVML (`pyNVML`) are available.~~

~~The GDK version,~~

~~the NVML documentation is available at NVIDIA's web site~~

~~`pyNVML` 7.352.0 is available via the `nvidia/pynvlm` module, and the documentation is on-line.~~

NVIDIA OpenACC/CUF

- The NVIDIA compilers support OpenACC.
- OpenACC, similarly to OpenMP, instructs a compiler to produce code that will run on the GPU
- It uses `pragmas`, i.e., instructions to the compilers that look otherwise like comments, to specify what part of the computation should be offset to the GPU.

  💡 A single pair of such `pragmas` produced a >300x speed up of the Julia set test case.

- This requires an additional license that is available on `Hydra` (not at SAO, tho).
- The NVIDIA compilers also support CUDA FORTRAN (aka CUF).
- You can write or modify existing FORTRAN code to use the GPU like you can using C/C++ & CUDA.
- Simple examples are available in `/home/hpc/examples/gpu/cuda`

Local Tool

- We provide two local GPU related tools (accessible by loading the `tools/local-admin` module):
  - `check-gpuse`: checks current GPU usage:

```
hpc@hydra-login% check-gpuse
hostgroup: @gpu-hosts (3 hosts)
                - --- memory (GB) ----  -  #GPU - --------- slots/CPUs ---------
hostname        -   total   used   resd -  a/u  - nCPU used   load - free unused
compute-50-01   -   503.3   19.6  483.7 -  4/0  -   64    0    0.0 -   64   64.0
compute-79-01   -   125.4   18.1  107.3 -  2/0  -   20    0    0.0 -   20   20.0
compute-79-02   -   125.5   46.2   79.3 -  2/1  -   20    1    2.1 -   19   17.9

Total #GPU=8 used=1 (12.5%)
```

- `get-gpu-info`: queries whether a node has a GPU, returns the GPU(s) properties and which process(es) use(s) the GPUs.

  It's a simple C-shell wrapper to run the python script `get-gpu-info.py`, that python script uses the pyNVML (python bindings to NVML).

  💡 the `get-gpu-info` wrapper checks if the first argument is in the form `NN-MM`, and if it is will run `get-gpu-info.py` on `compute-NN-MM`

  in other words, "`get-gpu-info 79-01 0 -d`" is equivalent to "`ssh -xn compute-79-01 get-gpu-info 0 -d`"

Here is how to use it:

```
usage: get-gpu-info.py3 [-h] [-i] [-d] [-l [LOOP]] [-c COUNTS]
                        [--ntstamp NTSTAMP]
                        [id]

get-gpu-info.py3: show info about GPU(s)

positional arguments:
  id                    specify the GPU id, implies --info

optional arguments:
  -h, --help            show this help message and exit
  -i, --info            show info for each GPU
  -d, --details         show details of running process, implies --info
  -l [LOOP], --loop [LOOP]
                        repeat every LOOP [in sec: 10 to 3600], default is 30,
                        implies --info
  -c COUNTS, --counts COUNTS
                        limits the no. of times to loop, implies --info
  --ntstamp NTSTAMP     specify how often to put a time stamp, by default puts
                        one every 10 readings

Ver 1.1/0 Oct 2021/SGK
```

For example:

Examples:

```
hpc@hydra-login-01% get-gpu-info
0 GPU on hydra-login-01
hpc@login01% get-gpu-info 50-01 -d
4 GPUs on 50-01
Thu May 16 15:30:06 2024
id               ------ memory ------  ------ bar1 --------  ---- usage ----
   --- name ---    used/total            used/total            gpu  mem #proc
0  NVIDIA_L40S   479.1M/44.99G   1.0%  1.688M/64.00G   0.0%    0%   0% 0
1  NVIDIA_L40S   479.1M/44.99G   1.0%  1.688M/64.00G   0.0%    0%   0% 0
2  NVIDIA_L40S   479.1M/44.99G   1.0%  1.688M/64.00G   0.0%    0%   0% 0
3  NVIDIA_L40S   479.1M/44.99G   1.0%  1.688M/64.00G   0.0%    0%   0% 0

hpc@login01% get-gpu-info 79-01 -d
2 GPUs on 79-01
Thu May 16 15:30:12 2024
id               ------ memory ------  ------ bar1 --------  ---- usage ----
   --- name ---    used/total            used/total            gpu  mem #proc
0  Quadro_GV100  276.5M/32.00G   0.8%  2.688M/256.0M   1.0%    0%   0% 0
1  Quadro_GV100  276.5M/32.00G   0.8%  2.688M/256.0M   1.0%    0%   0% 0

hpc@login01% get-gpu-info 79-02 -d
2 GPUs on 79-02
Thu May 16 15:30:19 2024
id               ------ memory ------  ------ bar1 --------  ---- usage ----
   --- name ---    used/total            used/total            gpu  mem #proc
0  Quadro_GV100  6.212G/32.00G  19.4%  5.188M/256.0M   2.0%    0%   0% 1
    pid=494189 name=b'python3' used_memory=5.939G
1  Quadro_GV100  279.1M/32.00G   0.9%  3.188M/256.0M   1.2%    0%   0% 0
```

4. Available Queues

- Four queues are available to access the GPUs:
  - an interactive queue, `qgpu.iq`, that has a 24hr elapsed time limit, and
  - three batch queues `sTgpu.q, mTgpu`.q & `lTgpu.q`, corresponding to a short, medium and long time limit.
- There also limits on (CPU) memory usage and how many concurrent GPUs can be used, see the Available Queues page.

Note:

- You must request to use a GPU to run in these queues, as follows:

  `% qsub -l gpu`

  or

  `% qrsh -l gpu`

  these are equivalent to:

  `% qsub -l gpu,ngpu=1`

  or

  `% qrsh -l gpu,ngpu=1`

- If your job will use two GPUs, use:

  `% qsub -l gpu,ngpu=2`

- You can specify what type of GPU to use with

  `-l gpu,gpu_arch=L40S`

  or

  `-l gpu,gpu_arch=GV100`

5. Examples

Trivial Examples

I wrote a trivial test case: computing a Julia set (fractals) and saving the corresponding image. It is derived from NVIDIA's own example.

You can find that example, and equivalent codes, under `/home/hpc/examples/gpu`

`cuda/`	CUDA and C++ code (.cu .cpp Makefile)
`cuda/gpu`	GPU example
`cuda/cpu`	CPU equivalent
`matlab/`	MATLAB, using standalone compiled code (available for now only at SAO)
`matlab/gpu`	GPU example
`matlab/cpu`	CPU equivalent
`idl/`	IDL CPU-only equivalent (for comparison)

Note:

I wrote more sophisticated alternative to that example, to achieve a 500:1 speed up compared to the equivalent computation running a single CPU.

💡 That's reducing a 7.5 hour long computation to less than 1 minute, in a case that is intrinsically fully "parallelizable."

It illustrates the potential gain, compared to the cost of coding using CUDA (an extension of C++)

NVIDIA Examples

~~You can find the NVIDIA examples under `/share/apps/nvidia/cuda/samples` and under `/cm/shared/apps/cuda10.0/sdk`.~~

~~I was able to build them, but not on the login nodes, as they use libraries that are currently only available on the GPU nodes~~

~~You can find this under `/home/hpc/tests/gpu/cuda/samples/`.~~

Timing

[This section is old and these tests have yet to be re-run]

I use a simple fractal computation (Julia set) to run timings. The computation is simply
`z =: z * z + c`

where `z` and `c` are two complex numbers.

The assignment is iterated `N` times and computed on a `M x M` grid, where `z = x + i y` and `x` and `y` are equispaced values between -1.5 and 1.5.

The final value of `z` is converted to an index `iz = 255*exp(-abs(z))` and the index `iz` converted to a color triplet using a rainbow color map.

the computation is repeated for a range of values of `c`.

I wrote various versions using

C++ CUDA (a basic one, an optimized one),

CUDA FORTRAN

C++ and FORTRAN (CPU-only as reference),

MATLAB (CPU) and MATLAB using GPU,

and IDL (CPU only).

In the single precision versions, `z` and `c` are each represented by `floats`,

in the double precision versions, they are represented by `doubles`.

In the tests I ran for timing purposes I did not save the resulted image.

The codes, job files and log files are available in `/home/hpc/examples/gpu/timing/`

I ran them on 201 values of `c and for N=150`, timings units are `[HH:]MM:SS[.S]` (*not yet complete*)

	single precision			speed		double precision				speed		
size M x M	**4k x 4k**	**8k x 8k**	**16x 16k**	**ratio**		**4k x 4k**	**8k x 8k**	**16k x 16k**		**ratio**	**Comments**	**Note**
GPU cases:											**GPU specific code**	
CUDA	03:05.0	12:20.8	49:23.6	9.202							NVIDIA CUDA - their example	
Optimized CUDA	00:05.1	00:17.6	01:09.3	371.6		00:07.6	00:27.8	01:50.3		238.8	NVIDIA CUDA - optimized	
MATLAB GPU	01:33.2	04:46.0	18:56.4	22.04		02:06.2	07:13.0			14.69	MATLAB equiv, using GPU	1
GPU cases, using `pragmas`:											**Regular code with special directives**	
PGI C++ OpenACC	00:54.3	02:09.4		42.30							C++ with OpenACC	2,3a
PGI C++ OpenACC (optimized)	00:12.1	00:42.2	02:46.2	155.5		00:13.8	00:52.1	03:27.6		128.5	C++ with OpenACC, optimized	2,3a
PGI C++ OpenACC (`myComplex`)	10:52:44			0.043							uses `myComplex` class	2,3b
PGI C++ OpenAC (`std::complex<>`)	10:52:45			0.043							uses `std::complex<float>`	2,3c
PGI F90 OpenACC	00:06.0	00:21.1	01:23.0	311.8		00:09.5	00:34.4	02:18.6		191.4	F90 with OpenACC pragma	2,4
PGI (CUF Fortran CUDA)	00:05.1	00:17.3	01:08.2	376.0		00:07.7	00:27.8	01:54.6		234.8	PGI FORTRAN CUDA (optimized)	2,4
CPU only cases:											**CPU equivalent cases**	
C++ (PGI)	28:13.7	01:55:10	07:30:38	1.000		27:49.2	01:51:15	07:25:02		1.021	C++, PGI compiler (v15.9)	
C++ (Intel)	25:35.6	01:42:21	06:49:39	1.109		25:32.9	01:42:08	06:48:22		1.112	C++, Intel compiler	
C++ (gcc)	28:24.6	01:53:37	07:35:41	0.999		28:36.9	01:54:11	07:36:53		0.994	C++, GNU compiler (v4.9.2)	
PGI C++	28:53.6	01:55:38		0.986		28:33.2	01:54:01			0.999	(duplicate)	3a
PGI C++ (`myComplex`)	28:11.0	01:52:02		1.015							uses `myComplex` class	3b
PGI C++ (`std::complex<>`)	03:53:08	15:29:09		0.123							uses `std::complex<float>`	3c
PGI F90	33:36.8	02:13:50		0.850		34:04.9	02:31:51			0.793	F90, PGI compiler (v15.9)	4
MATLAB (1 thread)	01:09:00	04:34:24		0.414		01:03:18	04:06:06			0.457	MATLAB on 1 CPU	1
MATLAB (multi-threaded)	14:05.7	56:09.6		2.027		13:25.4	53:25.9			2.129	MATLAB on multiple CPUs	1
IDL (1 thread)	01:32:13	06:24:22		0.303							IDL on 1 CPU	1
IDL (multi-threaded)	12:50.0	01:32:50		1.720							IDL on multiple CPUs	1

Notes:

Green means ran faster, red slower, grey is the reference (yellow: not run).

Optimized CUDA and CUDA FORTRAN ran 371 and 376 times faster than single CPU C++

MATLAB and IDL tests use the run-time environment, for now we only have MATLAB licenses at SAO (including compiler).

Access to the OpenACC and FORTRAN CUDA from PGI requires a license upgrade - our license was upgraded in May 2016.

C++ does not support complex arithmetic like FORTRAN does, so I used 3 approaches:

1. write out the complex arithmetic explicitly using only floating points operations,
2. use a C++ class `myComplex`, and let the compiler figure out how to port it to the GPU,
3. use the C++ `std::complex<>` class template, and also let the compiler figure out how to port it to the GPU.

Using OpenACC with C++ is more tricky than using it with FORTRAN.
The Julia set computation generates NaN, handling this in some of the C++ cases may explain the slow down.

FORTRAN supports complex arithmetic as a built-in data type `COMPLEX`.

Last updated 18 Nov 2021 SGK

How to Use Containers

- `singularity` is now available on all the compute nodes, the interactive nodes and the login nodes.
  - There is no longer any module or host-group needed tun use i@
- Examples are under `/home/hpc/examples/containers`

Last Updated 16 May 2024 SGK

FAQs

The list of installed packages is on our software page. You can also view the available modules with `module avail`, which will list the software that have modules.
Login to Hydra and issue the command `module avail`,

or go to the status page(s) (@si.edu or @cfa.harvard.edu), or the Hydra-7 tools page and click on link to display the list of available modules
Did you use the `-cwd` flag in your qsub script/embedded directives?

If you do not use the `-cwd` flag, which tells the scheduler to use your 'current working directory',

your job will run from your home folder and all log files will be written there.
Yes, a couple of nodes offer interactive access to compute nodes on Hydra.

- Use the command `qrsh` instead of `qsub` (not `ssh`);
- ⚠ like all queues, the interactive queue has limits (CPU time, elapsed time, memory and number of CPUs/threads/cores/slots);
- 💡 Read the documentation under Available Queues.

Cluster Upgrades

- 2024 Upgrade to Hydra-7
- Nov 2021 Updates: Compilers, Tools and More
- 2021 Upgrade to Hydra-6
- 2019 Upgrade to Hydra-5

Last Updated 27 Nov 2023 SGK

2024 Cluster Upgrade to Hydra-7

This page lists the upgrades that will take place during the April 22 - May 2, 2024 upgrade.

While we are making every effort to set up the new configuration as backward compatible as possible, there will be changes, see below.
We will adjust the Wiki once the changes are in place.

What Is Being Upgraded

1. The OS version (& *flavor*) of the cluster (i.e., Rocky 8.9)
2. The version of the Grid Engine (i.e., v8.8.1 aka 2023.1.1)
3. The cluster management tool, i.e, Base Command Management (aka BCM v10.0)
4. The version of most tools (compilers, etc.)
5. new Hardware: fifteen new compute nodes will be added:
    a. two nodes with 2x96c or 192 cores (CPUs) and 1.5TB of memory
    b. twelve nodes with 2x64c or 128 cores (CPUs) and 1.0TB of memory
    c. One quad GPU servers, with 4x L50S NVIDIA GPUs
6. We have/will decommission our oldest compute nodes, namely
    a. the `compute-43-xx` series, and
    b. the `compute-00-xx` series.
7. This will:
    a. increase the number of CPUs from about 4900 to about 5900,
    b. increase the total memory from about 40 to 40TB,
    c. decrease the number of compute nodes from about 87 to 78.
    d. The new compute nodes have the latest generation architecture.
8. As a result of the hardware upgrades, we will adjust system configuration (queues, disk quota, modules, etc.)
9. We will use the downtime to upgrade any needed firmware and make other adjustments as needed.

Details on Changes

Click on the links for details

1. New list of modules
    a. Some modules have moved
    b. New set of compilers versions
    c. MPI
2. Bio packages
    a. The list of packages has been pruned
    b. The list of modules has been trimmed
    c. Some generic tools have been moved
3. Queues: changes and new ones
    a. Changes in limits
    b. New GPU queues
    c. New `ompi` PE for NVIDIA MPI
    d. I/O queues changes, use `-l ioq`
4. GPUs
5. New version of the command `module`
    a. version 5.3.1 allows use of shortcut `ml`
    b. customization
6. Disk space changes
    a. Consolidation
    b. New quotas
7. Misc

Last updated 03 May 2024 SGK

2024 Upgrade Details

1. New list of modules
    a. Some modules have moved
    b. New set of compilers versions
    c. MPI
2. Bio packages
    a. Reduced packages from 200+ to ~100, adding 8 new ones.
    b. Moved some general use packages from `bio/` to `tools/`
3. Queues: changes and new ones
4. GPUs
5. New version of the command module
6. Disk space changes
    a. Consolidation
    b. New quotas
7. Misc
    a. not everything was rebuild (like `gdl`, `plplot`, some libraries), check the list of modules
    b. utilities not available under Rocky 8

New list of modules

The list of available modules has changed, and is available

- as a HTML document, or
- as a text file.

Worth pointing out:

- Some modules have moved from `bio/` to `tools/`
- New modules: we have installed `mamba`, an alternative to `conda` and `miniconda`
- New set of compilers versions
    - We still support the 3 compilers: GCC, Intel,and NVIDIA.
    - The list of available versions for each compiler have changed.
    - We installed the most recent stable versions available to date.
    - The latest Intel compilers names have changed, you will get warnings when loading some modules.
    - We no longer support the (old) PGI compilers, they are superseded by the NVIDIA ones.

> ⓘ We recommend recompiling codes built on Hydra-6 under CentOS 7.x with either the Intel or the NVIDIA compilers.
>
> Codes built with GCC are likely to work fine, unless they use dynamic libraries that are no longer available or no longer available for that specific version.

- MPI support and module naming
    - We still support various flavors and versions of the MPI libraries with each compilers:
        - GCC: `mvapich` and `openmpi`
        - Intel & NVIDIA: vendor supplied, `mvapich` and `openmpi`
            - `mvapich` versions 2.3.6 and 2.3.7p1
            - `openmpi` versions 3.1.6, 4.1.6 and 5.0.1
        - The PE (parallel environment specified via `-pe`) is different depending on the flavor and in some cases the compiler
            - This will be properly documented when the Wiki is up to date, but
            - the new examples, in `~hpc/examples/`, show what to use for which combo
    - We reorganized the MPI module naming
        - module names include now both the MPI and the compiler version,
        - shorter names are pointers (links) to the most recent and stable version,
        - check the new module list.

Bio packages

1. We have tested and validated on Hydra-7, under Rocky 8 some 100 "bio" packages, a list we had to prune down from the 200+ that were over the years installed on Hydra-6:
    - the list is on the Bio Packages page, here.
    - we added 8 new packages, the list is on that page, down here
2. As we reorganized things, we also moved some general use packages from `bio/` to `tools/`
    - that list is also on that page, here.
3. The list of packages that we pruned is also also on that page, here.

Queues: changes and new ones

- We have increased the limit on the virtual memory for both the hi-CPU and hi-MEM queues:
    - hi-CPU limit remains 8GB/slot for resident memory, but has been increased to 64GB/slot for virtual memory.

        - hi-MEM limit remains 450GB/slot for resident memory, but has been increased to 900GB/slot for virtual memory.
- New queues"
    - We added 2 queues to use GPUs,
        - the GPU queues are now: `sTgpu.q, mTgpu.q,lTgpu.q` and `qgpu.iq`
        - like other queues, the batch queues `sTgpu.q, mTgpu.q,` and `lTgpu.q` correspond to short/medium/long time
        - `qgpu.iq` remains the interactive GPU queue/
    - You still need to `-l gpu` when submitting to the GPU queues
- New PE: `ompi`
    - The NVIDIA MPI libraries need to run using the `-pe ompi N` specification
- I/O queues
    - ⚠ To use the I/O queues, you must now use
        - `-l use_ioq` or `-l ioq`
        - no longer `-l use_io` or `-l io`

GPUs

- We now have 8 GPUs on 3 compute nodes, two dual GV100 (4), one quad L40S (4)
    - you can specify what type of GPU to use with
        - `-l gpu,gpu_arch=L40S`

          or
        - `-l gpu,gpu_arch=GV100`
- These servers run CUDA Version 12.4 (Driver Version 550.54.15) ~~12.2 (Driver Version: 535.154.05)~~
- The NVIDIA compilers come with various version of CUDA for each version of the compiler.

New version of the command module

The command module has been upgraded to version 5.3.1. It works as before but has a few improvements:

- explicit sticky modules: two modules are preloaded and are sticky, i.e. you cannot unload them
- `ml` shortcut:
    - `module list, module load` and `module unload` can be shorten using `ml`as follows:

	is a shortcut to
`ml`	`module list`
`ml tools/ffsend`	`module load tools/ffsend`
`ml -tools/ffsend`	`module unload tools/ffsend`

- customization
    - `module list` is by default more verbose and in color
    - this can be customized and what is shown when loading a module can be also customized
    - the following modules customize the output of the module command:

`ml module-nocolor`	do not use colors
`ml module-nowarn`	disable some warning
`ml module-simple-format`	simplify the output of module list
`ml module-simple`	load the 3 module above
`ml module-color`	specify a color scheme (red for sticky, green for auto-loaded)
`ml module-verbose`	set module in verbose mode, equiv to using the -v flag

    - You can load and unload these module to your liking.

Details about the new version of module can be found in the module man page or here.

Disk space changes

- Consolidation:
    - the various public disks have been consolidated as single disks,
    - namely:
        - `/pool/sao`, `/pool/genomics`, etc.. are now on one single NetApp volume, called `/pool/public`
        - `/scratch/sao, /scratch/genomics`, etc... are now on one single GPGS fileset, called `/scratch/public`
        - `/data/sao, /data/genomics,` etc. are no one single NetApp volume `/data/public`
    - The new full paths are therefore `/pool/public/sao`, `/pool/public/genomics`, etc,
    - although the shorter names will still work and point to the right place/full path.

- New quotas:

- disk space quotas have been increased as follow:

disk space	**quota**	
location	**was**	**now**
`/pool`	5.0TB	7.5TB
`/scratch`	10.0TB	15.0TB
`/data`	2.0TB	4.5TB

Misc

- Not everything that was available on Hydra-6 (like `gdl, plplot`, and some libraries), was rebuild
  - check the list of modules.
- Some utilities are no longer available under Rocky 8, like
  - `finger`: check `pinky`, use `pinky -l` for an equivalent to `finger`
  - `ruptime`: we implemented a poor man version of `ruptime`, for the non compute nodes, since `qhost` returns the compute nodes load

---

Last updated 03 May 2024 SGK

Bio Packages

1. We have tested and validated on Hydra-7, under Rocky 8 some 100 "bio" packages, a list we had to prune down from the 200+ that were over the years installed on Hydra-6:
    - the list is down this page, here.
    - we added 8 new packages, the list is on this page, down here
2. As we reorganized things, we also moved some general use packages from `bio/` to `tools/`
    - that list is also down this page, here.
3. The list of packages that we pruned is also down this page, here.

General Packages Moved from bio/ to tools/

- The following packages had their modules under bio/ and have been moved to tools/:

```
awscli, ffsend, julia, rclone, ruby
```

New Packages in bio/

- The following packages are now available on Hydra-7, they were not available on Hydra-6:

```
admixtools, basespace, blobtoolkit, mitos, paup, quast, raxml-ng, rsem
```

Pruned Packages

- The following packages, that were available on hydra-6, *are no longer* available on Hydra-7:

```
anuga, aster, atram, autoparts, bayesass, beagle_5.1, blasr, blast_taxa_backfill, blat, centrifuge,
comp_popgen_sjsw, delimitr, discovista, dna-algorithm, dotnet-core, edirect, edta, eems, ervin,
ete3, exabayes, fatotwobit, fsct, gadma, gatb-minia-pipeline, gblocks, genomescope, gnuparallel,
goleft, gsutil, haystac, hifiadapterfilt, hmmer, hybphylomaker, hyde, hyphy, ibpp, imagemagick,
insector, ivis, jags, jbrowse, kakapo, makehub, manta, mauve, mcscanx, megadetector, metamaps,
metamorpheus, metapop2, mitogeneextractor, mitohifi, ml_derkarabetian, mpboot, mptp, msmc2, multiqc,
ngsadmix, ngstools, opt-sne, orthomcl, paml, partitionfinder, patchwork, pbsuite, pcangsd, phast,
phrapl, phylobayes, phylobayes-mpi, phyloflash, phylomad, phylonet, phyparts, phyx, pilon, prokka,
proteinortho, pyrad, ragout, relernn, rjags, rsem-eval, salmon, salsa, seq-seq-pan, shapeit,
sharkmer, smc++, spruceup, sspace, stampy, star, strauto, stringtie, supernova, tabix, transrate_1.
0.3, trinotate, vcflib, velvet, viridal, w2rap-contigger, wgd
```

- ⚠ If you need one of these tools, contact us at `si-hpc@si.edu`

Complete List of `bio/` Packages

- Here is the full list of `bio` packages available on Hydra-7 as we migrated to Rocky 8 (May 2024):

⚠ For many packages the default version has changed. In some cases. Please review your job files to ensure that the intended version is being used. ⚠

Module	Version(s) on Hydra-7 *"(def)" denotes default version*	Version(s) on Hydra-6 *"(def)" denotes default version*	Change in default version?	Note
`abyss`	2.3.7(def)	2.2.4(def)	yes	
`admixtools`	7.0.2(def)			
`admixture`	1.3.0(def)	1.3.0(def)		
`angsd`	0.941(def)	0.921, 0.931, 0.937(def), 0.94	yes	
`assembly_stats`	0.1.4(def)	0_1_3(def)	yes	
`astral`	5.7.8(def), 5.15.5-MP	5.7.8(def), mp-5.15.5		

augustus	3.5.0(def)	3.3.2(def)	yes	
basespace	1.5.4(def)			
bayescan	2.1(def)	2.1(def)		
bbmap	39.06(def)	38.67(def)	yes	
bcftools	1.19(def)	1.9(def)	yes	
beagle	4.0.1(def)	3.2(def), 4.0.1	yes	
beast	2.6.7, 2.7.6(def)	1.10.4, 2.6.0, 2.6.0-no-beagle, 2.6.1, 2.6.3, 2.6.6, 2.6.7(def), 2.7.5-no-beagle, 2.7.6-beagle	yes	
bedtools	2.31.1(def)	2.28.0(def)	yes	
bioawk	1.0(def)	1.0(def)		
bioperl	1.7.8(def)	1.7.2(def)	yes	
biopython	1.83(def)	1.74(def), 1.74_python2.7	yes	
blast	2.15.0(def)	2.6.0, 2.6.0_old_db, 2.8.1, 2.8.1-precompiled, 2.9.0, 2.9.0_v5, 2.10.1(def), 2.13.0	yes	
blast2go	To be installed	1.4.4, 1.5.1(def)		⚠️ This will be installed after the deployment of Hydra77
blobtoolkit	3.3.10(def)			
blobtools	1.1.1(def)	1.0.1(def), 2, 2.6.3	yes	This is for blobtools 1.x, later versions are in the module bio/blobtoolkit
bowtie2	2.5.3(def)	2.3.5(def), 2.5.1	yes	
bpp	4.7.0(def)	1, 4.6.1(def)	yes	
busco	5.7.0(def)	3.0.2(def), 4.0.2, 5.1.3, 5.2.2, 5.3.2, 5.4.3	yes	
bwa	0.7.17(def)	0.7.17(def)		
cactus	2.8.0(def)	2019.03.01, 2020(def), cactus_test	yes	
canu	2.2(def)	1.8(def), 2.1.1	yes	
cd-hit	4.8.1(def)	4.8.1	yes	
cutadapt	4.7(def)	2.4(def)	yes	
dates	753(def)	753(def)		
diamond	2.1.9(def)	1.0(def)	yes	
exonerate	2.4.0(def)	2.4.0(def)		
fastp	0.23.4(def)	0.23.4	yes	
fastqc	0.12.1(def)	0.11.8(def)	yes	
faststructure	1.0(def)	1.0(def)		
fasttree	2.1.11(def)	2.1.11(def)		
gatk	3.8.1.0, 4.5.0.0(def)	3.8.1.0(def), 4.1.3.0	yes	

gemoma	1.9(def)	1.0(def), 1.7.1, 1.9	yes	
getorganelle	1.7.7.0(def)	1.6.2d(def), 1.7.5	yes	
guppy	6.5.7(def)	5.0.16(def)	yes	
hifiasm	0.19.8(def)	0.14.2(def), 0.16.1	yes	
hisat2	2.2.1(def)	2.2.1(def)		
htslib	1.19.1(def)	1.9(def)	yes	
hybpiper	1.3.1_final, 2.1.6(def)	1.3.1(def), 2.0.1, 2.1.6	yes	
illumiprocessor	2.10(def)	2.0.9_trimgalore, 2.10_trimgalore, 2.10_trimgalore_2023-02-17(def)	yes	
ipyrad	0.9.94(def)	0.7.30(def), 0.9, 0.9.74	yes	
iqtree	1.6.12, 2.3.1(def)	1.6.12, 2.0.4, 2.0.6, 2.1.1, 2.1.2, 2.1.3(def), 2.rc.1	yes	
jellyfish	2.3.1(def)	2.3.0(def)	yes	
kraken	2.1.3(def)	2.0(def), 2.1.2	yes	
krakenuniq	1.0.4(def)	1.0.1(def)	yes	
mafft	7.525(def)	7.407(def)	yes	
maker	3.01.03(def)	2.31.10(def)	yes	
mapdamage2	2.2.2(def)	2.2.1(def)	yes	
masurca	4.1.1(def)	3.2.2, 3.3.3, 3.3.3_test, 3.3.8(def), 3.3.9, 4.0.0, 4.0.9, 4.1.0	yes	
merqury	1.3(def)	1.3(def)		
migrate	3.7.2, 5.0.6(def)	3.6.11, 3.7.2, 4.4.4		
minimap2	2.28(def)	2.24(def)	yes	
mitofinder	1.4.1(def)	1.2(def), 1.4	yes	
mitos	2.1.8(def)			
mitoz	3.6(def)	3.5(def)	yes	
mrbayes	3.2.7a(def)	3.2.7_beagle, 3.2.7a(def)	yes	
nucmer	3.2.3, 4.0.0rc1(def)	3.23(def)	yes	
oma	2.6.0(def)	2.3.1, 2.4.1, 2.4.2		
orthofinder	2.5.5(def)	2.0.9, 2.5.4		
paup	4a168(def)			
phyluce	1.6.8, 1.7.3(def)	1.5_tg, 1.6.7(def), 1.7.0, 1.7.1	yes	
picard-tools	2.20.6, 3.1.1(def)	2.20.6(def)	yes	

plink	1.90b7.2(def)	1.90b4(def)	yes	
psmc	0.6.5(def)	0.6.5(def)		
purge_dups	1.2.6(def)	1.0(def)	yes	
qiime2	2024.2-amplicon(def), 2024.2-shotgun	2019.7, 2019.1, 2020.8, 2021.11(def)	yes	qiime2 now has separate amplicon and shotgun sequencing packages
quast	5.2.0(def)			
R	4.3.3(def)	3.5.2_rgdal, 3.6.0_conda, 3.6.1(def)	yes	
raxml	8.2.13(def)	8.2.12(def), 8.2.12-mpi, ng-0.9.0	yes	
raxml-ng	1.2.1(def)			
repeatmasker	4.1.5, 4.1.6(def)	4.0.9(def)	yes	
repeatmodeler	2.0.5(def)	1.74(def)	yes	
revbayes	1.2.2(def)	1.0.11(def), 1.0.13	yes	
rohan	1.0.1(def)	1.0.1(def)		
rsem	1.3.3(def)			
samtools	1.19.2(def)	1.9, 1.17(def)	yes	
seqkit	2.8.1(def)	0.13.2, 2.3.1(def)	yes	
seqtk	1.4(def)	1.0(def)	yes	
spades	3.15.5(def)	3.11.1, 3.12.0, 3.14.0(def), 3.15.5	yes	
sratoolkit	3.1.0(def)	2.9.6, 2.11.0(def)	yes	
stacks	2.66(def)	1.48, 2.6.5, 2.41(def)	yes	
structure	2.3.4(def)	2.3.4(def)		
transabyss	2.0.1(def)	2.0.1(def)		
transdecoder	5.7.1(def)	5.5.0(def)	yes	
transrate	1.0.3	1.0.3, 1.0.3.2, 1.0.3_mamba, 1.0.3_orp		
trim_galore	0.6.10(def)	0.6.4(def)	yes	
trimmomatic	0.39(def)	0.39(def)		
trinity	2.15.1(def)	2.8.5(def), 2.9.0, 2.9.1, 2.13.2, r2013_2_25	yes	
vcftools	0.1.16(def)	0.1.16(def)		
wtdbg2	2.5(def)	2.5(def)		

Nov 2021 Updates: Compilers, Tools and More

Introduction

- New versions of the compilers (Intel, NVIDIA and GCC) and tools (Java, Python, IDL, MATLAB, Julia, CUDA), have been installed and tested on Hydra
- New modules are available to access these new compilers and tools
- The list of available modules has been updated and reorganized, it is now linked from the cluster status pages:
    - Hydra-6 Tools List of Available Modules
    - Status page (@si.edu) List of Available Modules
    - Status page (@cfa.harvard.edu) List of Available Modules
    - That list is updated nightly. You can get that list with the command `module avail`
- Access to GPUs has been simplified
- Singularity is now available on a set of compute nodes
- Password Requirement Change
    - We adjusted the password requirement on Hydra to conform to SI's policy (only one digit).
- We have updated the modules description (`whatis`) for some modules to be more informative and consistent, and
    - for 7 modules we will change the default versions when loading that module without version specification - on Monday November 29, 2021

Compilers

Intel

- Intel 2021.3 and 2021.4 have been installed, and 2021 2021.4
    - As of 2021.x, Intel is matching NVIDIA and releasing their compiler and more for free under the OneAPI name
    - Intel's modules are a bit of a mess, I've cleaned them up but they are very chatty
    - As before Intel also offers their version of Python
- Previous versions are still available (2015.x to 2021.x, check the list of available modules on the pages listed above or with with `module avail intel)`

PGI/NVIDIA

NVIDIA has acquired PGI and has repackaged these compilers as of 2020.

These compilers are now free and include CUDA support. This transition hasn't been the smoothest.

- The PGI compilers up to version 20.4 are still available (`module avail pgi`)
- As of 20.7 the compilers are NVIDIA (20.7 and 20.9), although they are available as pgi too
- NVIDIA versions 21.1, 21.2, 21.3, 21.5, 21.7 and 21.9 are now available (`module avail nvidia`)
- CUDA up to version 11.4 is available (CUDA 11.4 comes with NVIDIA 21.9)

GCC

- Versions 10.1. 10.2.0 and 11.2.0 are now available
- Previous versions are still available (`module avail gcc`)
- Version 8.2.0 is not longer available on Hydra-6, it has been substituted by 10.2.0 by BCM

MPI: Vendor, OpenMPI and MVAPICH

- Intel's MPI distributions are not working on Hydra - we've logged this with Intel's support.
- NVIDIA MPI distribution (OpenMPI 3.x) is working on Hydra (not their OpenMPI 4.x versions)
- Built from source versions of OpenMPI (4.x) and MVAPICH (2.3.x) are available for all 3 compilers/
- The following combinations are working and supported, other versions are available, look at the list of available modules for additional details:

```
    mvapich/intel/2019.5 - BFS | openmpi/intel/*     - VFV DNW | openmpi4/intel/2019.4 - BFS
                 /2020.4 - BFS |                     - VFV DNW |               /2020.4 - BFS
                 /2021.4 - BFS |                     - VFV DNW |               /2021.4 - BFS
    mvapich/pgi/19.9     - BFS | openmpi/pgi/19.9    - VFV     | openmpi4/pgi/19.9     - BFS
               /20.4     - BFS |            /20.4    - VFV     |             /20.4     - BFS RDW
    mvapich/nvidia/20.9  - BFS | openmpi/nvidia/20.9 - VFV     | openmpi4/nvidia/20.9  - BFS RDW
                  /21.9  - BFS |               /21.9 - VFV     |                /21.9  -
BFS
    mvapich/gcc/4.8.5    - BFS | openmpi/gcc/4.8.5   - BFS     | openmpi4/gcc/* - use openmpi/gcc instead
               /4.9.1    - BFS |            /4.9.1   - BFS     |
               /4.9.2    - BFS |            /4.9.2   - BFS     |
               /5.3.0    - BFS |            /5.3.0   - BFS     |
               /6.1.0    - BFS |            /6.1.0   - BFS     |
               /7.3.0    - BFS |            /7.3.0   - BFS     |
               /9.2.0    - BFS |            /9.2.0   - BFS     |
               /10.1.0   - BFS |            /10.1.0  - BFS     |
               /10.2.0   - BFS |            /10.2.0  - BFS     |
               /11.2.0   - BFS |            /11.2.0  - BFS     |
```

Notes:

- BFS: built from source; VFV: version from vendor; DNW: do not work; RDW: runs despite warnings
  - mvapich uses -pe mpich
  - openmpi uses -pe orte,
    - except for pgi & nvdia VFV (openmpi) that needs -pe mpich
- Examples are on hydra under `/home/hpc/examples/mpi`

Tools

Java

- Java 17.0.1 is available, as well as version 1.8.0_45

Python

- Versions 3.8, 3.9. 3.10 are available:
  - 3.8 is the most recent Anaconda version (2021.05),
  - 3.9 and 3.10 were built from source.
- Intel's version 3.7.11 is also available, distributed by Intel's OneAPI 2021.4

IDL

- version 8.8.1 is available
- IDL licensing mechanism is currently a simpler one, hence `idl` issues the following message:

```
Licensed for use by: Harvard-Smithsonian Astrophysical Observatory (Main)
License: 100554-5516875-BUF
License expires 30-Nov-2022.
```

MATLAB

- Matlab runtime version 2021a and 2021b are available

Julia

- Version 1.6.2 and 1.6.3 are available

CUDA

- CUDA is now included with the NVIDIA compilers,
- Version 11.4 comes with the 21.9 version of the compilers

GPU

- Access to the nodes with GPUs has been simplified, just specify `-l gpu` to `qsub` or `qrsh`
- More details on this in the GPU section of the Reference pages

Containers: singularity

- Singularity has been installed on a handful of node, you can access it by specifying the `@container-hosts` host group
- More details on this in the Containers section of the Reference pages

Password Requirement Change

- We adjusted the password requirement on Hydra to conform to SI's policy (only one digit).
- Passwords must conform to SI password policies and meet the following requirements:
    - at least 12 characters in length, and include at least:
        1. *one digit*,
        2. one upper case,
        3. one lower case, and
        4. one special character;
    - moreover, new passwords cannot be too similar to old passwords.

Modules Description and Default Version

- We have updated the modules description (`module whatis`) for some modules to be more informative and consistent, and
- We will change the default version when loading a module without version specification (7 cases).
    - These 7 changes are:

```
     idl/8.8 -> 8.8.0  -> 8.8.1
   matlab/rt -> R2020a -> R2021b
intel/python -> 36     -> 37
       intel -> 2020   -> 2021
 tools/julia -> 1.0.5  -> 1.6.3
tools/python -> 3.7    -> 3.8
         pgi -> 19.9   -> 20.4
```

and will occur on Monday November 29, 2021**.**

Last updated 19 Nov 2021 SGK

2021 Cluster Upgrade to Hydra-6

This page lists the upgrades that took between August 30th 2021 and September 14th.

- While we are making every effort to set up the new configuration as backward compatible as possible, there will be changes, see below.
- We anticipate a rewrite of the Wiki (organization and content) soon.

What Has Been Upgraded

1. The OS version of cluster (CentOS 7.9)
2. The Grid Engine (v8.6.18);
3. The cluster management tool, i.e, Bright Cluster Management (v9.1);
4. The NetApp controller (FAS8300); we replaced some of the oldest disks with new ones;
5. The firmware in some components of the GPFS;
6. Etc

Access

- Access to Hydra has not changed, ssh into `hydra-login01.si.edu` or `hydra-login02.si.edu`.
- Your credentials (username and password) have not changed.
- The web pages under `https://hydra-5.si.edu` have been moved to `https://hydra-6.si.edu`.
  - In most cases you should be redirected, but if you are not, please update your bookmarks accordingly.

List of Changes

Main Software Upgrade

What: upgrade of the operating system, the cluster management tool and the grid engine versions

Impact: access to a more recent software release and updated versions of the associated tools.

Details: upgraded O/S to CentOS 7.9 (release upgrade), UGE to v8.6.18 (newer release), BCM to v9.1 (new version).

NOTE: users will see a warning when logging on either login node that the host key has changed.

This is normal. You can either edit the known hosts file, delete it or hit OK when prompted to update it.

On most systems the known host file is `~/.ssh/known_hosts` .

Also, some mail readers (like GMail) have marked email coming from `hydra-6` as spam.

Check your spam folder and set your mail reader to accept all emails from `hydra-6.si.edu` and from `hydra-6a.si.edu`.

Installed a New NetApp Controller with Some New Disks

What: replaced the aging NetApp controller by a new one, and added some disks.

Impact: faster access to files under `/home`, `/data` and `/pool`. We also increased the capacity of `/home`,

and of the public space under `/data` and `/pool`. We have (or will) also increase(d) some of the quotas.

Details: upgraded the FAS8040 with a FAS8300 with some new disk shelves, while migrating most of the older

disk shelves from the old controller to the new one, increasing the NetApp capacity to 600TB.

Also, the filesystems `/data/biology` and `/data/nasm` and now on separate volumes, while we added

`/data/data_science` , `/pool/data_science` , `/data/fellows` and `/pool/fellows` to match

`/scratch/data_science` and `/scratch/fellows` .

NOTE: We are now backing up the content of `/home` to Amazon's Glacier storage.

This is a low cost backup option aimed at disaster recovery (i.e., recover the content of `/home` if the storage system that holds the content of `/home` fails in our data center). In other words, what is stored currently under `/home` is quite safe, because /home is on a highly reliable disk system (NetApp) and a copy of it exists on Amazon Glacier.

Snapshots are still enabled for the `/home` file system, hence files under `/home` deleted within 4 weeks can, in most cases, be restored by the users. Beyond that period, we plan to keep backups of `/home` for up to a year, but restoring files from these backups are costly, both in terms of support manpower and actual billing by Amazon. Users who would need to recover data from this backup would need proper justification and may need to contribute to the actual recovery costs. Feel free to contact us if need be.

We plan next to investigate the feasibility of backing up `/data` to Glacier as well.

Queue Changes

What: removed the uTSSD.tq and uTGPU.tq queues.

Impact: access to local SSD storage is now integrated with the high-CPU and high-memory queues.

GPU servers and associated software have yet to be re-installed on Hydra-6.

Details: request for SSD storage is similar, but you only need to specify `-l ssdres=XXX`, and no longer "`-q uTSSD.tq -l ssd`".

As for access to GPUs, we will soon migrate the GPU servers to Hydra-6 and install the required software. We also plan to use the Grid Engine native GPU support, stay tuned and/or contact us.

Modules Changes

What: the module `tools/local` has been split into `tools/local-users` and `tool/local-admin`,

and the module `tools/local-users` is always loaded, like the `uge` module.

Impact: users have access to some Hydra-specific tool without having to load an extra module.

Details: we decided to split what was available with `module load tools/local` into `tools/local-users` and `tool/local-admin` and load the `tools/local-users` module for all users. You can still load `tools/local` that will load `tools/local-user` and `tools/local-admin,` and `tools/local+` remains unchanged.

Changes Affecting Bioinformatics Modules

What: you can now load any module previously know as `bioinformatics/XXX` using the `bio/XXX` shortcut.

Impact: bio is now a shortcut to bioinformatics.

Details: a symbolic link `bio`, pointing to `bioinformatics,`has been created as a shortcut. We plan to migrate all the bio-informatics tools to `bio/` in the future, hence we recommend that you start using `bio/` in lieu of `bioinformatics/`

Changes Affecting IDL

What: IDL versions prior to version 8.6 are no longer available.

Impact: IDL users must use version 8.6 or higher, 8.8 is the most recent version (default and recommended version).

Details: IDL changed the license manager as of 8.6 and we decided not to migrate the old, unsupported and obsolete licence manager,

hence only version 8.6 or higher are available. The default IDL version is 8.8 (8.8.0), version 8.8.1 will be installed soon and will become the default value.

Note that most likely version 8.9 will come with a new license manager.

Changes Affecting R

What: R versions prior to version 3.5.2 are no longer available.

Impact: R user will need to switch to the supported version or use conda to install a different version.

Details: We removed old versions to streamline availability of R, under bio/R or tools/R, the default version is 3.6.1.

Changes Affecting MPI Users of OpenMPI

What: a workaround is needed for users whose login shell is `/bin/csh` and use OpenMPI using `-pe orte` for parallel jobs

Impact: users whose login shell is `/bin/csh` need to add a line to their `~/.cshrc`

Details: A "*feature"* of UGE version 8.6.18 is causing a mysterious `Unmatch ".`error message for users whole login shell is `/bin/csh`

when submitting jobs with `-pe orte` (OpenMPI jobs). There is a simple fix to avoid this problem, simply add the line

```
if ($?JOB_ID) set backslash_quote
```

to your `~/.cshrc` file. Explanations can be found under the `execd_params` section of `man sge_conf,` see E`NABLE_BACKSLASH_ESCAPE`,

and the Lexical structure section of `man csh`.

Changes Affecting the qacct+ Local Tool

What: `qacct+` now use the GE's native DB ARCo and has been modified accordingly.

Impact: Almost no delay between `qacct` and `qacct+,` and the arguments to `qacct+` have changed.

Details: `qacct+` is now using the ARCo DB, a GE native product that is updated nearly in real time by the GE.

`qacct+` has been modified to query ARCo, hence some types of queries are no longer available, and the arguments to `qacct+` have been modified accordingly.

In the process most of the argument syntax has been changed, see `man qacct+` or `qacct+ -help` .

Changes Affecting the Compilers

What: we are no longer supporting old versions of some of the compilers and plan to install the most recent versions of these compilers soon.

Impact: Users need to use the more recent versions of the GCC, PGI/NVIDIA, Intel compilers

Details: Rather than delaying the reopening, we have not yet, but will soon install the most recent versions of the NVIDIA and Intel compilers and will consider installing the most recent GCC compilers. We no longer support some of the old version of the PGI and Intel compilers, and have removed some of the associated modules . Once we have installed and validated the most recent versions of the compilers we will consolidate which versions of each compiler will be supported, notify the users and update the documentation.

Changes Affecting Blast2GO

What: Blast2GO has been upgraded from 1.4.4 to 1.5.1. The reference GO database has been upgraded from 2019_10 to the latest, 2021_06.

Impact: Newer database will give you the most up to date annotations. You will need to re-run `hydracliprop` after loading the module to generate an up to date cli.prop file. In Blast2GO, the argument `-saveb2g <path>` has been changed to `-savebox <path>` (OmicsBox format).

Details: The new module is `bio/blast2go/1.5.1` . The previous version's module `bio/blast2go/1.4.4` and its database are currently still available, but will be removed soon.

Coming Up

What: more compute servers and more GPFS disk space and rewritten documentation

Impact: more CPUs, more disk space under `/scratch` and better documentation

Details: We have ordered eight new 64-core servers and an extra ~500TB of GPFS disk space.

Because of a shortage of computer parts and the current supply chain disruption, these will likely be delivered by the end of 2021.

We will rewrite the documentation hosted on confluence.si.edu/display/HPC; it will be reorganized and updated to reflect the most recent upgrade.

Last updated 13 Sep 2021 SGK

2019 Cluster Upgrade to Hydra-5

⚠ ⚠ ⚠ Read the Hydra-5 Migration Notes and Reset Your Password ⚠ ⚠ ⚠

The Hydra cluster has been upgraded as follows:

- Software
    - Upgrade OS to CentOS 7.6,
    - Change our management tool (from Rocks 6.x to Bright Cluster, aka BCM 8.2),
    - Switch from SGE to UGE (Univa) version 8.6.6, a commercially supported version of SGE (backward compatible).

- Hardware
    - Added 16 new nodes (40c/380GB) for a total of 640 cores,
    - Decommissioned oldest nodes (old compute-6* and compute-4-*),
    - Add ed1.5PB of high performance storage (parallel file system GPFS, aka Spectrum Scale),
    - This is in addition to the 500TB of NetApp and the 950TB of near-line storage (NAS),
    - The `/scratch` disk has been moved to the GPFS and increase in size from 100TB to 900TB (450TB+450TB for `/scratch/genomics` and `/scratch/sao`)
    - Moved all the nodes to a 10Gb/s Ethernet connection (was 1Gb/s).

> ⚠ **Please Note**
>
> - While we made every effort to set up the new configuration as backward compatible as possible, there are changes.
> - Read the migration notes!
> - ~~The upgrade will take place from August 26th through September 2nd, 2019.~~
> - ~~During this time, Hydra will be inaccessible to users and as of 9am on Monday August 26th any remaining running jobs will be killed.~~
> - ~~We hope to have Hydra back up before September 2nd, but that decision won't be made until the upgrade work is completed.~~
> - ~~During the down time, access to files stored on Hydra will be limited, and at times unavailable. Note that none of your files will be deleted.~~
>
> ~~Plan your use of Hydra accordingly.~~

💡 Please read this page carefully, and after the upgrade, do not hesitate to contact us (at SI-HPC-Admin@si.edu) if something is not working as it should any longer.

- For biology and genomics software issues and/or incompatibilities that arise, please contact SI-HPC@si.edu,
- SAO users please contact Sylvain at hpc@cfa.harvard.edu.

List of Software Changes

- Implementation of user account management using LDAP
    - The procedure to change your password has changed:
        - You can use `passwd` on either the login node and that's it, or
        - Use the Self Service Password (SSP) page, listed at https://hydra-5.si.edu
    - Email SI-HPC-Admin@si.edu ***only if this fails***.

- Switching from Rocks to BCM
    - BCM locates things differently from Rocks;
        - for example `/opt` is no longer used, instead BCM uses `/cm`
    - If you use modules, everything should work the same;
        - but if you hardwired locations, things may no longer work the same;
    - The compute nodes will be renamed as follows:
        - `compute-NN-MM,`
        - *i.e.* the "`N`", or logical rack number, and
        - M the node *index* are both two-digit numbers;
    - You should use modules as much as possible.

- Job Scheduler
    - We switched to UGE, or Univa Grid Engine; that
        - is backward compatible with SGE;
        - offers some additional features;
        - although the output of some commands will look different, and
        - some have different options .

- The list of queues and their limits will not changed,
    - a few *complex values* will change (to use local SSD and GPU usage)
    - ⚠ One important change ⚠ is that

the memory reservation value will no longer be specified per slots, but per jobs:

```
the specification  "-pe mthread 10 -l mres=20G,h_data=20G,h_vmem=20G,himem"
must be changed to "-pe mthread 10 -l mres=200G,h_data=20G,h_vmem=20G,himem"
to reserve 200GB of memory for this job, asking the scheduler to start on a node that has 20G per
thread of free memory
```

- Compilers
  - The default compiler versions has changes, and
  - they had to be reloaded.
  - For MPI jobs the compiler/flavor/version combos has also changed.

- Local tools
  - The tools accessible with the `tools/local` module have been split into two groups and
  - some of the names have been changed or simplified
  - Use `module help tools/local` and `module help tools/local+` to see what the split is and what the names are.
  - Use `module help tools/local-bc` to see the correspondence
  - Loading `tools/local-bc` will
    - load `tools/local` and `tools/local`+ and
    - create aliases to be backward compatible (-bc)
  - 💡use the new names whenever possible

Last updated 06 Sep 2019 SGK.