Tools and services offered by HPAC Platform and Fenix with hands-on

2nd HPAC Platform Training, 26-28 Nov 2019

Colin McMurtrie (CSCS)       Alex Upton (CSCS)
Agenda

- 09:30 - 10:00 Overview of Tools and Services
- 10:00 - 10:30 Pollux OpenStack VM Tutorial
- 10:30 - 11:00 Coffee break
- 11:00 - 12:00 Piz Daint Scalable Compute Tutorial including Sarus

- Tutorial sheet and slides can be downloaded from: bit.ly/HPAC_tutorial
- Slides can be downloaded from: bit.ly/HPAC_slides
Before we begin...

- Who is in the room?
  - Masters students/PhD students/Postdocs/Professors/Other?

- How many people have a HBP collab/wiki account?
  - If not, why?

- How many people have an ICEI account?
  - If not, why?

- How many people in the room find their research is limited by lack of computational resources?
  - e.g. not enough computing power/unable to share workflows/unsure on how to share data etc.
HPAC/Fenix/ICEI - what are they?

- The High Performance Analytics and Computing (HPAC) Platform develops and provides supercomputing, storage, visualisation and simulation technology that can run on supercomputers. This allows scientists to:
  - Run large-scale, data intensive, interactive multi-scale brain simulations up to the size of a full human brain
  - Manage the large amounts of data used and produced by simulations and in experiments
  - Manage complex workflows comprising concurrent simulation, data analysis and visualisation workloads

- The ICEI project delivers a set of e-infrastructure services that will be federated across five European supercomputing centres that include BSC (Spain), CEA (France), CINECA (Italy), CSCS (Switzerland) and JSC (Germany), to form the Fenix Infrastructure.
What Services does Fenix/ICEI/HPAC provide?

- **End-user Services**
  - Scalable Compute Services (both hybrid CPU+GPU nodes and multicore CPU-only nodes)
  - Interactive Compute Services (including hybrid nodes)
  - SWIFT Object Storage
  - Data Storage Services
  - (Data Transfer Service) \(\leftarrow\) HPAC
  - (Continuous Integration Services) \(\leftarrow\) HPAC
  - (Software Packaging and Deployment Services) \(\leftarrow\) HPAC
  - (Visualisation Services) \(\leftarrow\) HPAC

- **Platform Services**
  - Infrastructure Services (middleware access to HPC resources via Rest APIs)
  - Infrastructure as a Service (e.g. OpenStack) for Virtual Machine Services
  - Data Management Services
  - User and Resource Management Services
  - Service Accounts (currently not available at all sites)
Fenix/ICEI provides the Base Infrastructure for HPAC

- Support of user workflows
- Job Submission
- Data access
- Site-to-site Data Transfer
- Single sign-on to HPC Allocations
- Launch simulations and data analysis tasks from Jupyter notebooks
- Software deployment via Docker Containers

HPAC Infrastructure

1. Create Docker image
2. Push to Docker Hub
3. Pull into storage at HPC center
4. Run at scale on HPC system

Platform Services (PaaS)
- Data Services Portal
- HPC Portal
- Supported Scientific Libraries
- Externally supported portals

Infrastructure Services (IaaS)
- Virtualization
- Containers
- Web interfaces
- Custom middleware

IT infrastructure
- Computing
- Storage
- Networking
- Security

HBP Platforms

Collaboratory

UNICORE

Infrastructure Services

REST APIs
ICEI Resources for HBP

- ICEI resources have already been made available to HBP and PRACE by CSCS

- There are currently 21 HBP projects with compute allocations at CSCS
  - More are in the approval stages

- More resources are available than are being consumed so HBP users are encouraged to apply for a compute allocation
  - More on this in the next session and tomorrow

<table>
<thead>
<tr>
<th>Component</th>
<th>ICEI Service Type</th>
<th>ICEI Total Allocation (Raw Resource)</th>
<th>Allocatable Unit</th>
<th>ICEI (100%)</th>
<th>HBP (25%)</th>
<th>PRACE (15%)</th>
<th>National (60%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack Cluster</td>
<td>VM</td>
<td>35 servers</td>
<td>Servers</td>
<td>35</td>
<td>8.75</td>
<td>5.25</td>
<td>21.00</td>
</tr>
<tr>
<td>Piz Daint Multicore</td>
<td>SCC</td>
<td>250 nodes</td>
<td>Node-Hours</td>
<td>465375</td>
<td>116,344</td>
<td>69,806</td>
<td>279,225</td>
</tr>
<tr>
<td>Piz Daint Hybrid</td>
<td>SCC + IAC</td>
<td>400 nodes</td>
<td>Node-Hours</td>
<td>744600</td>
<td>186,150</td>
<td>111,690</td>
<td>446,760</td>
</tr>
<tr>
<td>Store POSIX and Object</td>
<td>ARD</td>
<td>1000 TB</td>
<td>TB</td>
<td>1000</td>
<td>250</td>
<td>150</td>
<td>600</td>
</tr>
<tr>
<td>Tape library</td>
<td>ARD</td>
<td>3000 TB</td>
<td>TB</td>
<td>3000</td>
<td>750</td>
<td>450</td>
<td>1,800</td>
</tr>
<tr>
<td>Low latency storage tier*</td>
<td>NVM</td>
<td>80 TB</td>
<td>TB</td>
<td>80</td>
<td>20</td>
<td>12</td>
<td>48</td>
</tr>
</tbody>
</table>

*Early access technology. User workflows need to be adapted/augmented.
## Resources currently available

<table>
<thead>
<tr>
<th>Component</th>
<th>Site</th>
<th>Total ICEI</th>
<th>Minimum Request</th>
<th>Technical Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scalable Computing Services</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Piz Daint Multicore</td>
<td>CSCS (CH)</td>
<td>250 nodes</td>
<td>1 node</td>
<td>• Memory per node 64/128 GB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Compute nodes/processors: 1813 Cray XC40 nodes with Two</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Interconnect configuration: Cray Aries</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Intel® Xeon® E5-2695 v4 @ 2.10GHz (2 x 18 cores) CPUs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Interconnect configuration: Cray Aries</td>
</tr>
<tr>
<td><strong>Interactive Computing Services</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Piz Daint Hybrid</td>
<td>CSCS (CH)</td>
<td>400 nodes</td>
<td>1 node</td>
<td>• Memory per node: 64 GB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• GPU memory: 16 GB CoWoS HBM2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Compute nodes/processors: 5704 Cray XC50 nodes with</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Interconnect configuration: Cray Aries</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Intel® Xeon® E5-2690 v3 @ 2.60GHz (12 cores) CPUs and NVIDIA®</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tesla® P100 GPUs</td>
</tr>
<tr>
<td>VM Services</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pollux OpenStack Cluster</td>
<td>CSCS (CH)</td>
<td>35 servers</td>
<td>1 VM</td>
<td>2 types of compute node:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Type 1 - CPU: 2x Intel E5-2660 v4 14C/RAM: 512 GB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Type 2 - CPU: 2x Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz 8C/RAM: 768 GB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• VMs can be of various flavours and use up to 16 cores</td>
</tr>
<tr>
<td>Archival Data Repositories</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Store POSIX and Object, including backup on</td>
<td>CSCS (CH)</td>
<td>4000 TB</td>
<td>1 TB</td>
<td></td>
</tr>
<tr>
<td>Tape library (2x)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Active Data Repositories</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low latency storage tier (DataWarp)</td>
<td>CSCS (CH)</td>
<td>80 TB</td>
<td>1 TB</td>
<td>• Non-volatile Memory</td>
</tr>
</tbody>
</table>
How do I use ICEI Resources? (1)

• Firstly, you will need to have obtained an account via an ICEI request application
  • More info with request form can be found here: https://collab.humanbrainproject.eu/#/collab/28520/nav/203167
  • Application form shown on next slide, lightweight document with only 5 short sections
  • Detailed overview about applying for resources in session tomorrow afternoon, including hands-on walkthrough of the application process
Resource Application Form

1. Relation to HBP DoA and relevance to HBP call

Please provide information on the related work packages, tools, CME, etc., and explain how the project relates to the past and objectives of the project. How does the project relate to the published HBP call for resources in [YYYY]?

2. Preliminary Work (in case of a project extension)

Please provide a brief summary of your project results obtained up to your resource application.

3. IT resources requested

3.1 Resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Units</th>
<th>Quantity (requested)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Core</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other hardware</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2 Technical implementation plans

Please outline how the requested resources are intended to be allocated to the project.

3.3 Does this project involve processing of personal data as defined by GDPR?

Please select "Yes" or "No". If you select "Yes", please specify what kind of data is processed.

4. Scientific methodology, goals and impact

4.1 Scientific implementation plans

4.2 Resource management and work plan

Please describe your plan to manage the requested resources.

4.3 Dissemination

Please describe the planned activities and strategies for dissemination and knowledge exchange. If the requested resources are used to provide CME/HEC services, please describe the relevance for these services.

5. References

Please provide relevant literature or other information that relates to the project.

*Note: The project may be rejected if the Information Management Committee (IMC) determines it does not meet the requirements.*
How do I use ICEI Resources? (2)

Scalable Compute Resources:
The *Piz Daint* system is available as a state-of-the-art scalable compute resource for use by HBP users

- Accessible globally via Command-line Interface (more about this later)
- Via the UNICORE GUI
- Via the RESTful API offered via UNICORE for platforms
  - Use of Service Accounts for Platforms is also acceptable at some sites (e.g. CSCS)
  - See next slide for some more details
Interactive Compute Resources:
The *Piz Daint* system supports the use of Jupyter Notebooks for interactive supercomputing, powered by JupyterHub

- This is a multi-user Hub that spawns, manages and proxies multiple instances of the single-user Jupyter notebook server
  - More details below
- Subsequent sessions will demonstrate the use of this environment

*Piz Daint* and other HPAC HPC systems are also accessible from the Jupyter Notebooks service of the *Collaboratory*

- This relies on the RESTful API offered via UNICORE for platforms
- The sessions tomorrow will go into the details of how to do this
Archival Data Storage:
SWIFT OS can be accessed directly from your personal computer
- GUI clients e.g. CyberDuck
- SP5 Python Library
  - Better for mgmt. of access control lists (ACLs) and Object Buckets
  - More on this later
Reachable from inside the Collaboratory
- Get/Put from Jupyter Notebooks
- More capabilities coming soon
How do I use ICEI Resources? (5)

**Active Data Repositories:**
- Come as part of the compute allocation (= $SCRATCH)
- Low-latency storage tier (Cray DataWarp with SSDs) in *Piz Daint* can also be requested

**Archival Data Repositories:**
- Are available either as part of a computing request (your proposal should state how much you need)
- Or separately in a data-storage only use case (in which case a separate proposal is needed)
How do I use ICEI Resources? (6)

Virtual Machine Resources:
The Pollux OpenStack IaaS is available as a VM resource to host your platform VMs:
- Accessible globally via the Horizon GUI interface
- RESTful API can be used for automation

Example of a Platform service (NRP) using VMs AND HPC resources.
Service Detail: Software Packaging and Deployment
Containers

- Isolated environments to run applications/services
- Images include all software dependencies
- Prescriptive, portable, easy to build, quick to deploy
Container implementations

- OpenVZ (Linux Containers)
- docker
- LXC
- Singularity
- SARUS
- rkt
- ORACLE Solaris
- Charliecloud
- Shifter

HPC focused
Docker

- Extremely popular container implementation

- Easy to use authoring tools
  - Container images are created from recipe-like files
  - Images can be named, tagged and built on top of other images

- Cloud-based image distribution strategy
  - Several remote registries available (e.g. Docker Hub)
  - Client includes facilities to authenticate, push and pull images
Docker workflow

1. An image is created locally from a Dockerfile
2. Push (i.e. upload) the image to a remote registry
   • DockerHub is the public registry maintained from the Docker company
3. Pull (i.e. download) the image on a target machine and run the container
Key terms

- **Image**: standalone, executable package that includes everything needed to run a piece of software
  - code, runtime libraries, environment variables, configuration files

- **Container**: runtime *instance* of an image
  - What the image becomes in memory when actually executed
  - Runs completely isolated from the host environment by default
    - only accessing host resources if configured to do so
So... how are containers useful?

Containers give the possibility to create (scientific) applications that are:

1. Portable
2. Reproducible
3. Easy to deploy
4. Easy to test

Unfortunately Docker containers are not a panacea for HPC environments because of:

- Security concerns
  - root in the container means root on shared parallel file systems
- Performance Portability
  - Performance is important in HPC (it’s in the name...)
Sarus container engine

- OCI-compliant container engine engineered by CSCS
- Designed for the requirements of HPC
- Consistent UX with widely-used Docker: small learning curve
- Transparent native performance through hooks e.g. MPI, GPU
- Enables use of standard, open, upstream components on HPC systems
- Extensible architecture encourages vendor engagement and improves maintainability
User workflow using scalable compute

1. Create Docker image
   - FROM debian
   - RUN apt-get ...
   - RUN make ...

2. Push to Docker Hub

3. Pull into storage at HPC center

4. Run at scale on HPC system
Sarus from the user perspective

- Consistent experience
  - With Docker: closely resembles CLI (see next slide)
  - With host environment: env variables, uid/gid, file permissions, working directory

- Pull images from Docker registries (e.g. Docker Hub, NVIDIA NGC)

- Import images from local tar archives (no cloud upload required in case someone does not want to put image on Docker Hub)

- Integration with the workload manager (Slurm)

- Native performance from GPUs and Cray high-speed interconnect

- Access to parallel filesystems inside containers
Sarus CLI

- Sarus

  $ sarus pull [options] <image>[<:tag>]

  $ sarus load [options] <file> <image>

  $ sarus images

  $ sarus rmi <image>[<:tag>]

  $ sarus run [options] <image>[<:tag>] <command> <args>

- Docker

  $ docker pull [options] <image>[<:tag>]

  $ docker load [options] -i <file>

  $ docker images [options] [repo[<:tag>]]

  $ docker rmi [options] <image> [image...]

  $ docker run [options] <image>[<:tag>] <command> <args>
Service Detail:
JupyterHub Service at CSCS
Using JupyterHub at CSCS

- This service enables the interactive execution of Jupyter Notebook on Piz Daint over both single and multiple nodes.
  - The supported python version is python3.

- The service is accessed through the address
  - [https://jupyter.cscs.ch](https://jupyter.cscs.ch)
  - Users should provide their HPAC credentials in order to login

- Once logged in, the user is redirected to a job setup page
  - Allows typical job configuration options to be selected in order to allocate the resources that are going to be used to run Jupyter
    - Account
    - Type of Piz Daint node type (GPU or MC)
    - Number of nodes
    - Wall-clock time limit

- More information at: [https://user.cscs.ch/tools/interactive/](https://user.cscs.ch/tools/interactive/)
JupyterHub Service Architecture (1)

- The **current** architecture protects the notebook in each compute node (CN) by launching a JupyterHub Service along with it.
JupyterHub Service Architecture (2)

Notebooks v4.3 and newer are protected with a per-session tokens
• Avoids the creation of several custom spawners
  • Ideally we want one CSCS spawner only
• Will be integrated with an Infrastructure Services API (UNICORE or similar)
• The frontend will be kept outside of the HPC system
1. Login via Horizon Web GUI
2. Create small network
3. Create router
4. Create key pair
5. Launch instance
6. Assign floating IP
7. Ping instance to verify connection and login
8. Sharing data between workshop participants using ACLs (if time allows)
“OpenStack is a free and open-source software platform for cloud computing, mostly deployed as infrastructure-as-a-service (IaaS), whereby virtual servers and other resources are made available to customers.”
Pollux is CSCS’s general purpose OpenStack system
  • since summer 2017

~433 VMs

~338 users

VM uptime In 2018:
  • 99.93% unplanned
  • 99.76% unplanned and planned

Currently at version Queens
  • RedHat OpenStack Platform 13
Horizon OpenStack GUI

- You can **create VMs** and execute other actions using the OpenStack **graphical interface Horizon**, can be reached at the URL: [https://pollux.cscs.ch](https://pollux.cscs.ch)
View Instances in the Project

- Can see all current instances in the project

---

**Instances**

<table>
<thead>
<tr>
<th>Instance Name</th>
<th>Image Name</th>
<th>IP Address</th>
<th>Flavor</th>
<th>Key Pair</th>
<th>Status</th>
<th>Availability Zone</th>
<th>Task</th>
<th>Power State</th>
<th>Time since created</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>studox</td>
<td>-</td>
<td>10.0.0.9</td>
<td>m1.tiny</td>
<td>studox</td>
<td>Active</td>
<td>&quot;1&quot; nova</td>
<td>None</td>
<td>Running</td>
<td>52 minutes</td>
<td>Create Snapshot</td>
</tr>
<tr>
<td>HPAC_test</td>
<td>-</td>
<td>10.0.0.24</td>
<td>m1.tiny</td>
<td>AU</td>
<td>Active</td>
<td>&quot;1&quot; nova</td>
<td>None</td>
<td>Running</td>
<td>2 hours, 38 minutes</td>
<td>Create Snapshot</td>
</tr>
</tbody>
</table>
First we want to create a network to connect to.
Next create subnet
Finally define the subnet details

Subnet details

Create Network

- Enable DHCP
- Specify additional attributes for the subnet.
- DNS Name Servers: 148.187.3.88
- Host Routes
Previously we created a private isolated network so now we want to connect it to the internet via a router.
Create router
Create router

Add Interface

Subnet *

HPAC_tutorial: 10.0.0.0/8 (HPAC_subnet)

Description:

You can connect a specified subnet to the router. If you don’t specify an IP address here, the gateway’s IP address of the selected subnet will be used as the IP address of the newly created interface of the router. If the gateway’s IP address is in use, you must use a different address which belongs to the selected subnet.
Generate Key Pair

Key Pairs

Displaying 3 items

- **Name**: AU

- **Name**: AU_HPAC_test

- **Name**: HPAC

Click here for filters.
Generate Key Pair

Key Pairs are how you log in to your instance after it is launched. Choose a key pair name you will recognize. Names may only include alphanumeric characters, spaces, or dashes.

Key Pair Name

- studox

Create Key Pair
Generate Key Pair

Opening studxx.pem

You have chosen to open:

- studxx.pem

  which is: Text Document (1.6 KB)
  from: blob:

What should Firefox do with this file?

- Open with Notepad (default)
- Save File
- Do this automatically for files like this from now on.

[OK] [Cancel]
Launch Instance

- Launch a new instance called studxx

![Launch Instance Example](image-url)
Launch Instance

- Next choose instance source and volume size
Specify hardware required

- Choose m1.tiny ‘flavour’ hardware config
Connect to Network

- Connect to the HPAC_network
Select Security Groups

- Make sure to also select HPAC_tutorial security group
Specify key pair

- Choose the `studxx` key pair you created previously

<table>
<thead>
<tr>
<th>Key Pair</th>
<th>Fingerprint</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Available Key Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>stud72_test</td>
</tr>
<tr>
<td>stud51_test</td>
</tr>
<tr>
<td>HPAC_test</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key Pairs</th>
<th>Fingerprint</th>
</tr>
</thead>
</table>
Associate floating IP
Associate floating IP

- Click the plus symbol

![Manage Floating IP Associations dialog box](image-url)
Associate floating IP

- Click Allocate IP
Associate floating IP

- An IP address has been allocated, click associate
Ping instance and login

```
C:\Users\uptonna\ssh> ping 148.187.98.32
Pinging 148.187.98.32 with 32 bytes of data:
Reply from 148.187.98.32: bytes=32 time=21ms TTL=51
Reply from 148.187.98.32: bytes=32 time=19ms TTL=51
Reply from 148.187.98.32: bytes=32 time=16ms TTL=51
Reply from 148.187.98.32: bytes=32 time=15ms TTL=51

Ping statistics for 148.187.98.32:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
  Minimum = 15ms, Maximum = 21ms, Average = 17ms

C:\Users\uptonna\ssh> ssh -i studyxx.pem ubuntu@148.187.98.32
The authenticity of host '148.187.98.32 (148.187.98.32)' can't be established.
ECDSA key fingerprint is SHA256:1g4VmiSLX7P31z0185mOdp5L45YEQTZNV83Hnw9GA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '148.187.98.32' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-22-generic x86_64)

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage

System information as of Thu Nov 21 17:52:15 UTC 2019

  System load:  0.0  Processes:     82
  Usage of / :  2.4% of 38.60GB  Users logged in:  0
  Memory usage: 6%  IP address for ens3: 10.0.0.9
  Swap usage:  0%

* Overheard at KubeCon: "microk8s.status just blew my mind".
  https://microk8s.io/docs/commands#microk8s.status

Get cloud support with Ubuntu Advantage Cloud Guest:
```
Set console password

```
To run a command as administrator (user "root"), use "sudo <command>". See "man sudo_root" for details.

ubuntu@studxx:~$ sudo passwd ubuntu
Enter new UNIX password:
Retype new UNIX password:
pwd: password updated successfully
ubuntu@studxx:~$
```
Login from console

- Login using password just created for user ubuntu
"The OpenStack Object Store project, known as Swift, offers cloud storage software so that you can store and retrieve lots of data with a simple API. It's built for scale and optimized for durability, availability, and concurrency across the entire data set. Swift is ideal for storing unstructured data that can grow without bound."
Object Store

- Two entities:
  - Objects (~files)
  - Containers (~directories)

- Access through REST APIs, but there are many clients available (explained in the next slides)

- Role based access control (RBAC)
  - Users can have different roles on different projects (admin, normal user, data operator etc.)

- Access control lists (ACL)
  - R and/or W access can be potentially granted to specific users on specific projects
  - Public access is available
  - Temporary URLs for anonymous access

- Metadata can be associated to objects and containers

- Documentation for CSCS’ Swift: [https://user.cscs.ch/storage/object_storage/](https://user.cscs.ch/storage/object_storage/) (perhaps remove this/change to make HBP specific)
Object Store Web Interface

Containers

PID

Displaying 7 items

- Name: 532c1756-6a5e-11e9-a872-800000208fe0
  - Size: 133 bytes
  - Download

- Name: 55df9d19e-6a5e-11e9-a872-800000208fe0
  - Size: 199 bytes
  - Download

- Name: 610807ba-6c9f-11e9-a0e0-800000208fe0
  - Size: 133 bytes
  - Download

- Name: 74ee34f4-6a5e-11e9-8caf-800000208fe0
  - Size: 133 bytes
  - Download

- Name: 836c3006-6b25-11e9-a8d4-800000208fe0
  - Size: 133 bytes
  - Download

- Name: a9f6be0a-72f4-11e9-acd8-800000208fe0
  - Size: 133 bytes
  - Download

- Name: d41322b6-7304-11e9-96a8-800000208fe0
  - Size: 133 bytes
  - Download

Displaying 7 items
CyberDuck access

- Documented at [https://user.cscs.ch/storage/object_storage/cyberduck/](https://user.cscs.ch/storage/object_storage/cyberduck/)
- Let’s give this a go…
Access control lists

- Access through **REST APIs**, but there are many clients available (explained in the next slides)

- **Role based access control (RBAC)**
  - Users can have different roles on different projects (admin, normal user, data operator etc.)

- **Access control lists (ACL)**
  - R and/or W access can be potentially granted to specific users on specific projects
  - Public access is available
  - Temporary URLs for anonymous access

- **Metadata** can be associated to objects and containers

- **Documentation** for CSCS’ Swift: [https://user.cscs.ch/storage/object_storage/](https://user.cscs.ch/storage/object_storage/)
Other Access options

- It’s also possible to access Swift from:
  - Command line interface (see tutorial sheet)
  - S3 interface e.g. with s3curl
  - Python OpenStack libraries
  - REST APIs
Piz Daint - Scalable Compute

- **6th** most powerful supercomputer in the world (TOP500 list November 2019)

**Model**
Cray XC50/XC40

**XC50 Compute Nodes**
(Intel Haswell processor)
- Intel® Xeon® E5-2690 v3 @ 2.60GHz
  (12 cores, 64GB RAM) and NVIDIA® Tesla® P100 16GB

**XC40 Compute Nodes**
(Intel Broadwell processor)
- Intel® Xeon® E5-2695 v4 @ 2.10GHz
  (18 cores, 64/128 GB RAM)

**Login Nodes**
- Intel® Xeon® E5-2650 v3 @ 2.30GHz
  (10 cores, 256 GB RAM)

**Interconnect Configuration**
- Aries routing and communications ASIC
  Dragonfly network topology

**Scratch capacity**
Piz Daint scratch filesystem: 8.8 PB
You should have already obtained an account 😊

The front end Ela is accessible via `ssh` as `ela.cscs.ch`:
- It provides a minimal Linux environment
- ssh already available via MacOS/Linux, also now in Windows 10
- You can `ssh` the computing systems from Ela
- Can use either command line or optionally GUI tools such as WinSCP to transfer data from laptop to/from filesystems on Piz Daint and Ela

Please note the following:
- **No programming environments** on the front end system
- User scratch space is **not accessible** from Ela

$ ssh ela.cscs.ch

$ ssh daint.cscs.ch
1. Login via ssh to Ela
2. Login from Ela to Daint
3. See modules that are loaded
4. See available modules and search for specific modules
5. Load/unload module and select either hybrid or multicore nodes (DONE)
6. Transfer data to Ela/Piz Daint via command line
7. Toy example - submit batch job (https://user.cscs.ch/access/running/jobscript_generator/)
8. Graphical tools for transferring data e.g. WinSCP
ssh ela.cscs.ch - login node

```
C:\Users\uptona>ssh uptona@ela.cscs.ch
Warning: Permanently added the ECDSA host key for IP address '2001:620:808:4001::21' to the list of known hosts.
uptona@ela.cscs.ch's password:
Last login: Wed Nov 13 09:12:08 2019 from dhcp-133-237.cscs.ch

=================================================================
IMPORTANT REMINDER FOR USERS of CSCS facilities

help@cscs.ch - http://user.cscs.ch

=================================================================

=================================================================
IMPORTANT REMINDER FOR USERS of CSCS facilities

help@cscs.ch - http://user.cscs.ch

=================================================================

[uptona@ela6 ~]$  
```
ssh daint - connecting to Piz Daint

```
[uptona@ela6 ~]$ ssh daint
Password:

IMPORTANT REMINDER FOR USERS of CSCS facilities

help@cscs.ch - http://user.cscs.ch

Please load 'daint-gpu' module for using the GPU/Haswell nodes
or load 'daint-mc' module for the Multicore/Broadwell nodes

For more info, please refer to the User Portal:
https://user.cscs.ch/access/running/piz_daint

uptona@daint102:~>
```
module list - modules currently loaded
module avail - see modules available
module avail - search for Python

• Rather than go through all the modules, can return a list with all the modules that match a specific argument, e.g. in this case we look for Cray Python modules

```
uptona@daint102:~> module avail cray-python

---------------------------------  /opt/moduleFiles
cray-python/2.7.15.6            cray-python/2.7.15.7(default) cray-python/3.6.5.6    cray-python/3.6.5.7
uptona@daint102:~>
```
In addition to listing modules that match our interest, we can also find out more about them.

In this case, we see that it is a module for Python on the Cray XC system.
module daint-gpu - use Hybrid nodes

- Hybrid CPU/GPU nodes - load if you want to use GPUs

```bash
uptona@daint105:--> module load daint-gpu
uptona@daint105:--> module list
Currently Loaded Modulefiles:
1) modules/3.2.11.3
2) cce/9.0.2
3) craype-network-aries
4) craype/2.6.1
5) cray-libsci/19.06.1
6) udev/2.32-7.0.1.1_3.9__g8175d3d.ari
7) ugni/6.0.14.0-7.0.1.1_7.10__ge78e5b0.ari
8) pmi/5.0.14
9) dmapp/7.1.1-7.0.1.1_4.8__g38cf134.ari
10) gni-headers/5.0.12.0-7.0.1.1_6.7__g3b1768f.ari
11) xpmem/2.2.19-7.0.1.1_3.7__gcd43af6.ari
12) job/2.2.4-7.0.1.1_3.8__g36b56f4.ari
13) dvs/2.12_2.2.151-7.0.1.1_5.6__g7eb5e703
14) alps/6.6.56-7.0.1.1_4.10__g2e60a7e4.ari
15) rca/2.2.20-7.0.1.1_4.9__g8e3fb5b.ari
16) atp/2.1.3
17) perftools-base/7.1.1
18) PrgEnv-cr/6.0.5
19) cray-mpich/7.7.10
20) slurm/19.05.3-2
21) craype-haswell
22) xalt/2.7.10
23) daint-gpu
```

See in the list that the GPU module is loaded
module daint-mc - use multicore nodes

- Multicore nodes with 2x18 cores - ideal for OpenMP

See in the list that the multicore module is loaded
## Filesystems

<table>
<thead>
<tr>
<th></th>
<th>/scratch (Piz Daint)</th>
<th>/scratch (Clusters)</th>
<th>/users</th>
<th>/project</th>
<th>/store</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Lustre</td>
<td>GPFS</td>
<td>GPFS</td>
<td>GPFS</td>
<td>GPFS</td>
</tr>
<tr>
<td><strong>Quota</strong></td>
<td>Soft quota 1 M files</td>
<td>None</td>
<td>10 GB/user 100K files</td>
<td>Maximum 50K files/TB</td>
<td>Maximum 50K files/TB</td>
</tr>
<tr>
<td><strong>Expiration</strong></td>
<td>30 days</td>
<td>30 days</td>
<td>Account closure</td>
<td>End of the project</td>
<td>End of the contract</td>
</tr>
<tr>
<td><strong>Data Backup</strong></td>
<td>None</td>
<td>None</td>
<td>90 days</td>
<td>90 days</td>
<td>90 days</td>
</tr>
<tr>
<td><strong>Access Speed</strong></td>
<td>Fast</td>
<td>Fast</td>
<td>Slow</td>
<td>Medium</td>
<td>Slow</td>
</tr>
<tr>
<td><strong>Capacity</strong></td>
<td>8.8 PB</td>
<td>1.4 PB</td>
<td>86 TB</td>
<td>4.7 PB</td>
<td>3.6 PB</td>
</tr>
</tbody>
</table>

**Soft quota:**
- Soft quota on **scratch** to prevent excessive loads on the Lustre filesystem
- Quota reached: **warning at submit time, no job submission** allowed
Shared parallel filesystems based on the IBM GPFS software:
- Accessible from the login nodes using native GPFS client
- Storage space for datasets, shared code or configuration scripts
- Better performance with larger files (archive small files with `tar`)

Users are NOT supposed to run jobs here:
- The emphasis is on reliability over performance
- All directories are backed up with GPFS snapshots
- No cleaning policy until 3-months after the end of the project

Environment variables pointing to personal folders:
- `$HOME` points to `/users/$USER`
- `$PROJECT` points to `/project/<group_id>/$USER`
Fast workspace for running jobs:

- Designed for performance rather than reliability
- Cleaning policy: files older than 30 days deleted daily
- No backup: transfer data after job completion

Performance of Piz Daint scratch (Lustre filesystem):

- **Soft quota on inodes** (files and folders) **to avoid** large numbers of small files
- Occupancy impacts performance:
  - > 60%: we will ask you to **remove unnecessary data immediately**
  - > 80%: we will free up disk space **manually removing data**

All CSCS systems provide a scratch personal folder:

- the variable `$SCRATCH` points to the user space
Filesystems

What this means in practice:
1) Store important input/output data in $HOME in ela
2) Move data needed for running job to $SCRATCH in Piz Daint
3) Run job on data in $SCRATCH using batch script
4) Move output data such as results from job to $HOME or optionally local file filesystem

Let’s look at how we do this with a toy example...
Copy data to $HOME using SCP

• First we copy data from the local filesystem to the $HOME directory
Copy input data to $SCRATCH

• Next we copy input data from $HOME to $SCRATCH

```bash
[uptona@elal ~]$ ls -a
.
... bash_history .cache .emacs .gnu-emacs
[uptona@elal ~]$ ls
bin hbp_logo.jpg HPAC_tutorial_1.txt HPAC_tutorial.txt
[uptona@elal ~]$ 
[uptona@elal ~]$ ssh daint
Last login: Mon Nov 18 11:18:03 2019 from 148.107.1.6
IMPORTANT REMINDER FOR USERS of CSCS facilities
help@cscs.ch - +41 91 610 82 10 - http://user.cscs.ch

Please load 'daint-gpu' module for using the GPU/Haswell nodes or
load 'daint-mc' module for the Multicore/Broadwell nodes

For more info, please refer to the User Portal:
https://user.cscs.ch/access/running/piz_daint

uptona@daint105:~$ scp $HOME/HPAC_tutorial_1.txt $SCRATCH
uptona@daint105:~$ 
```
Batch script

- Instead of entering the information about allocations and jobs through the command line, a *batch script* is used.
- All the details can be written and submitted through the `sbatch` command.
- This is the preferred way of running complex jobs or making a large amount of submissions.
- It may seem daunting, but to help you there is a jobscript generator that you can use found here: https://user.cscs.ch/access/running/jobscript_generator/
Run job on $SCRATCH

• For our toy example we run a simple Python program to print ‘Hello from Python’ on different tasks
Inspect output file

- Output file named slurm-18178790.out is created
- Can open it using e.g. vim to see results
In the final step, we copy the output file to the home directory, and can optionally open again with e.g. Vim to see that it is the same.
Monitoring Jobs

- We can monitor submitted jobs in two ways
- 1st is via the `squeue -u <username>` command
- More convenient however is via the email notification setup in the batch script, submit the job and we are then sent an email when the job begins and another when it has finished
Graphical tools for moving data

- Also possible to use graphical tools such as WinSCP to move data from local system to Ela/Piz Daint.
Graphical tools for moving data

- After entering host name and password, we can now see data in the $HOME directory in the login node
Graphical tools for moving data

- Can also setup ssh tunnel to also connect to $SCRATCH directory on Piz Daint
Graphical tools for moving data

- Now able to see files in Piz Daint $SCRATCH$ directory and drag/drop to and from local file system
Sarus hands-on

- Load Sarus and pull image
Run container

- Run latency test in container with native MPI support (lower is better)
Run container

- Run latency test in container with MPI from the image (lower is better)
Run container

- Run bandwidth test in container with native MPI support (higher is better)
Run container

- Run bandwidth test in container with MPI from the image (higher is better)
How to get Help or More Information

General Contact for HPAC Platform:
  • HPAC Platform:  
    https://collab.humanbrainproject.eu/#/collab/264/nav/2378

How to apply for resources:
  • Send your proposals to: icei-coord@fz-juelich.de

Getting help:
  • Send emails to: hpac-support@humanbrainproject.eu
Thank You

colin@cscs.ch       alex.upton@cscs.ch

www.humanbrainproject.eu  @HumanBrainProj  Human Brain Project

2nd HPAC Platform Training | 26-28 Nov 2019 | Heidelberg