ML CLOUD USER GUIDE

ML Cloud User Guide

ML Cloud Team

AI Center / ML Cluster

ML CLOUD USER GUIDE

ML Cloud User Guide

Last update: April 16, 2024 Download PDF

1. USER GUIDE

This documentation explains various aspects of using the ML Cloud systems. It contains both introductory and advanced material. The documentation will be continuously updated.

- **All users: read the Good Conduct section.** The Ml Cloud is a shared resource and your actions can impact other users. (17/02/2023)

First time users might want to start at the beginning and work through the first few chapters, skipping those sub-chapters that contain advanced material. This way, you will learn how to log in to the ML Cloud familiarize yourself with the environment, find pre-installed software, run your containers and experiments on the ML Cloud.

1.1 News and notifications

News about planned/unplanned downtime, changes in hardware, and important changes in software will be published on the Portal. For more information on the different situations see below.

1.2 System status and activity

You can get a quick overview of the system utilization and status:

- Galvani
- Ferranti

1.3 Maintenance and Downtime

The ML Cloud Team will schedule maintenance in one of the following three manners:

1. **Rolling reboots:** Whenever possible, the ML Cloud Team will apply updates and do other maintenance in a rolling fashion in such a manner as to have either no or as little impact as possible to ML Cloud services.
2. **Partial outages:** The ML Cloud Team will do these as needed but in a manner that impacts only some ML Cloud services at a time.
3. **Full outages** These are outages that will affect ML Cloud Services depending on the system, such as outages of core networking services, data storage services, data centers power of cooling system maintenance or outages.

In the case of a planned downtime, a reservation will be made in the queuing system, so new jobs, that would not finish until the downtime, won't start. A notification message will be present in the Portal System Status Page, as well as mailing list notification will be sent in advance. We apologize for any inconveniences this may cause.

1.4 AI Conference Deadlines

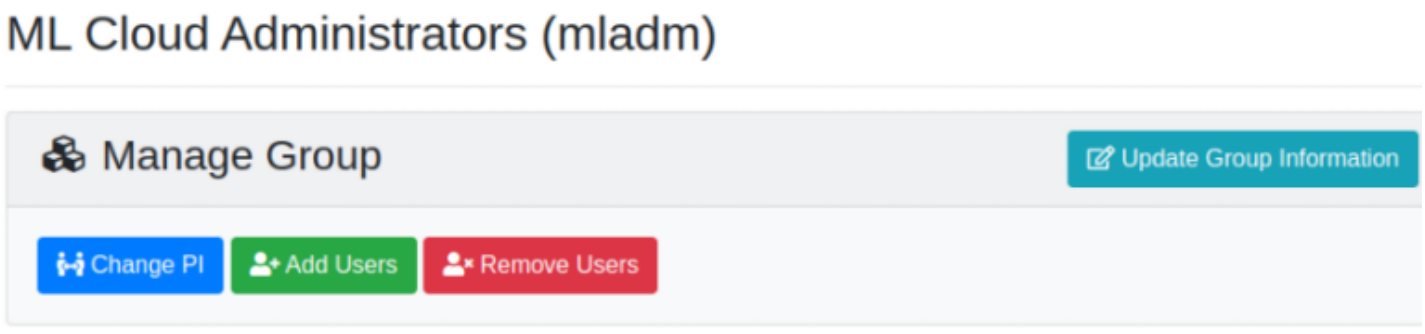
The ML Cloud Team is aware of the AI Conferences deadlines and will try to abide by them for regular maintenance schedules. If we have missed a conference from the list, please let us know from this form.

2. USER MANAGEMENT

The ML Cloud has a user management system where your Group PI or Manager can request the creation of a new account or decommisioning of an account for a member leaving the group.

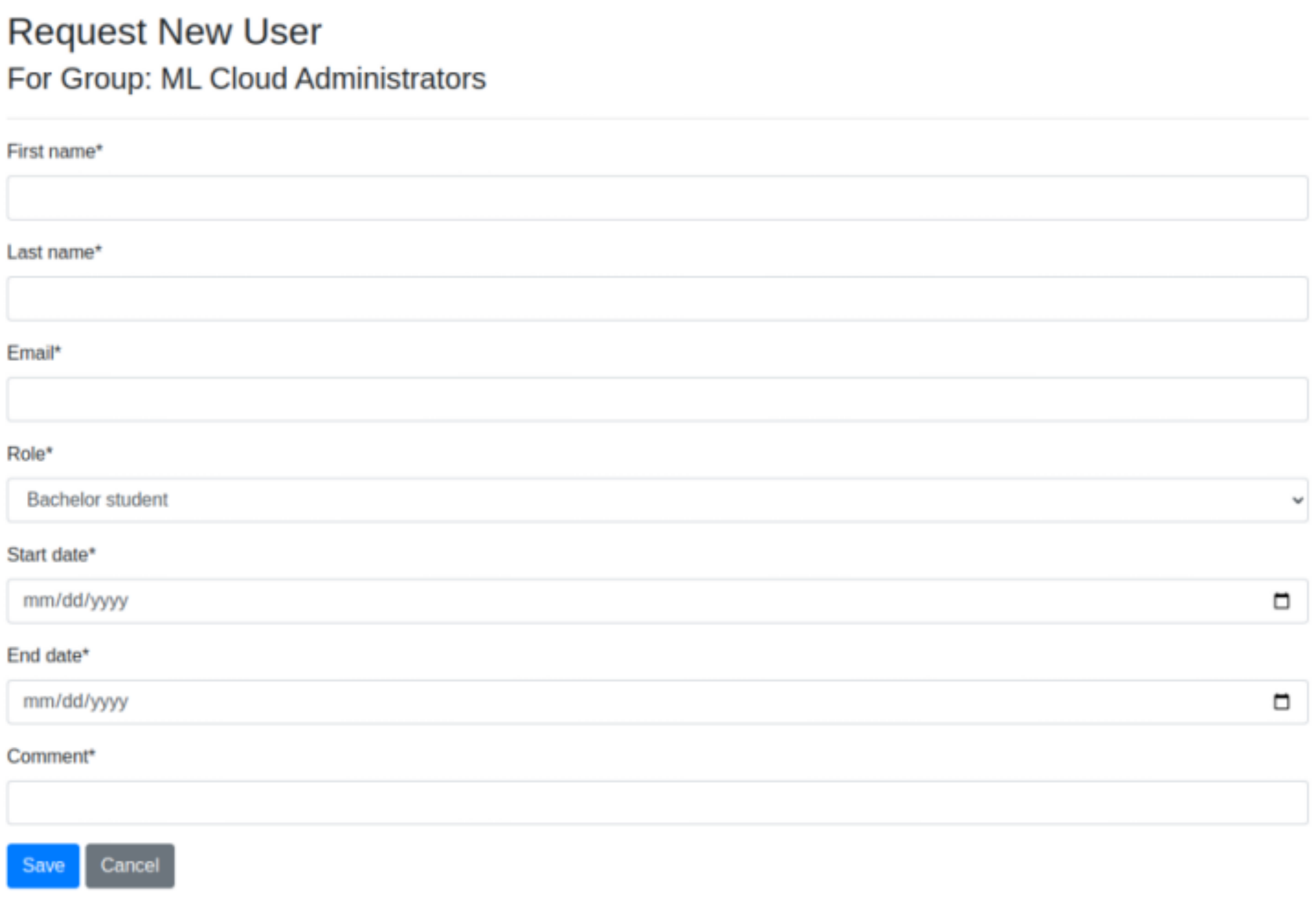
2.1 New User Accounts

The group manager or the group PI should use the User Management System to request new accounts by clicking on `Request New User`:



Then a new form will open with several fields:

1. First and Last Names,
2. Email,
3. Role - which tells us whether the account owner is student/PhD student/Postdoc/HiWi etc.
4. Start Date - or from when the account should be active. Keep in mind it might take a day or two to process your request.
5. End Date - this is to tell us when the account should be decommissioned.
6. Comment - anything else in particular we need to be made aware of. You can fill N/A if there is nothing.



Request New User
For Group: ML Cloud Administrators

First name*	
Last name*	
Email*	
Role*	Bachelor student
Start date*	mm/dd/yyyy
End date*	mm/dd/yyyy
Comment*	

2.2 User Modification

Users can upload their public ssh key to the user management system, which will then be used to access Slurm for each compute system.

To change your email, or name - please send us a ticket.

2.3 User Removal

Decommissioning is triggered in one of two ways:

1. If your end date is present in the system, we will begin the offboarding process 1 month before your departure, abiding by the principles described here.
2. A PI or Group Manager requests user removal through the user management system - which can be made either for one or multiple users.

To understand more what happens on decommissioning please refer to this section.

3. ACCESS

This page describes how to access ML Cloud resources; you will need an ML Cloud account and an ssh login key (described below).

3.1 Prerequisites

In order to gain access to the ML Cloud, you have to be given access to the system. To that, your group PI or Group Manager need to request one to be created for you through the User Management System. All ML-Cloud services share the same identity backend. The account/credentials can therefore be used to log into all all of them. These accounts are exclusive to the ML-Cloud.

Your username is generated by the ML Cloud; you should receive a Nextcloud password reset link to enable you to set your password (if you didn't, follow that link). This username and password is separate from those provided by other university services, and is specific to the ML Cloud systems.

The account will grant you access to:

- Compute Clusters
- Openstack
- ML Cloud Gitlab
- Nextcloud as flexible cloud-storage.
- ML Cloud User Management System

3.2 Login

At the ML Cloud, we do not allow logging into our systems solely with a password; rather, we require key-based authentication:

- You cannot log in to SLURM using username/password credentials. Instead, password free login based on public key cryptography is required.
- Your private key has to be stored in a file that is encrypted using a secure passphrase.

Once you have received an account on the ML Cloud, you have to deploy your ssh public key within **2 days** of account creation (the system administrators temporarily whitelist you for password based login) to the compute systems. If you miss this window, you will not be able to access the system.

3.3 Generate your key

OpenSSH is a popular and freely available SSH client (and server) for UNIX-like operating systems such as GNU/Linux and macOS. OpenSSH comes pre-installed on macOS. It is also contained in the package repository of many GNU/Linux distributions, e.g. `openssh-client` on Debian, Ubuntu, etc. or `openssh-clients` on Fedora, Centos, etc. As a last resort, the OpenSSH source code can be downloaded from the OpenSSH web site.

It is important to generate secure key pairs. The current best key type is called `Ed25519`. Generate a key of this type with the following line

```
ssh-keygen -a 100 -t ed25519 -f ~/.ssh/id_ed25519
```

The options specify the type of the key (`-t`), the number of key derivation function rounds (`-a`), and the location to place the key (`-f`), which is selected to be the default. Optionally, one can give a comment to the key with -C to help distinguish multiple keys.

If you still want (or have to) use RSA please make sure to use a bit length of `4096` by the following command:

```
 ssh-keygen -t rsa -b 4096 -o -a 100
```

Note

If the file `~/.ssh/id_ed25519` already exists, you probably don't want to override it as you might already be using it as credentials for another system. Instead, use a different file name, e.g. `~/.ssh/id_ed25519_mlcloud` and remember to use the same file name on all subsequent command lines in this document.

Afterwards, `ssh-keygen` will ask for a passphrase. This passphrase is purely private and has a priori nothing to do with your University or ML Cloud credentials. It is imperative to provide a strong passphrase at this point, i.e. one that cannot easily be guessed or found by brute force. It needs to be entered in the future to unlock your private key. You might want to use a password manager to save your key and ease the use of complicated passphrases.

A pair of keys, one public and the other private, will be generated. The public key authentication is the most secure and flexible approach to ensure a multi-purpose transparent connection to a remote server. This approach is enforced on the ML Cloud Platforms and assumes that the public key is known by the system in order to perform an authentication based on a challenge/response protocol instead of the classical password-based protocol.

The generated keys for `ed25519` are stored in the following files:

Key	Explanation
`~/.ssh/id_ed25519`	Contains private key that should be stored only on your machine. NEVER EVER TRANSMIT THIS FILE
`~/.ssh/id_ed25519.pub`	This file is the ONLY one SAFE to distribute

and for `rsa4096` are stored:

Key	Explanation
`~/.ssh/id_rsa`	contains the private key. NEVER EVER TRANSMIT THIS FILE
`~/.ssh/id_rsa.pub`	This file is the ONLY one SAFE to distribute

Keep the private part (i.e., `~/.ssh/id_ed25519`) of the key-pair safe, confidential, and on your local host only.

Only the generated public key (in the example above `~/.ssh/id_ed25519.pub`) needs to be uploaded to the ML Cloud.

3.4 Deploying your SSH Key to the System

Once you have your ssh key generated, your public key needs to be deployed on the ML Cloud. The easiest way to do that is to use the following command:

```
ssh-copy-id -i ~/.ssh/id_ed25519.pub ml-cloud-user-id@IP_OF_LOGIN_NODE
```

The command `ssh-copy-id` copies your public key into `~/.ssh/authorized_keys` on the cluster's login node.

In the event that you don't have permission to write to `~/.ssh/authorized_keys`, you have to grant yourself that permission with `chmod 600 ~/.ssh/authorized_keys`.

Once your key is on the cluster you may have to change its permission:

```
ssh USERNAME@IP_OF_LOGIN_NODE chmod 600 ~/.ssh/authorized_keys
```

After you have deployed your key you will be able to ssh into the cluster from your device without entering a password. **The authentication is performed via your key**.

3.5 Setting up Persistent Configuration

The "`ssh`" command (SSH protocol) is the standard way to connect to the ML Cloud. SSH also includes support for the file transfer utilities `scp` and `sftp`. Wikipedia is a good source of information on SSH. SSH is available within Linux and from the terminal app in the Mac OS. If you are using Windows, you will need an SSH client that supports the SSH-2 protocol: e.g. Bitvise, OpenSSH, PuTTY, or SecureCRT.

The user-side SSH configuration can be used to create shortcuts to targets/hosts and configure connections. The following entry creates a shortcut that allows you to refer to login nodes via short names by adding your short names to your ~/.ssh/config:

```
Host slurm Hostname IP_OF_LOGIN_NODE User ml-cloud-user-id ForwardAgent=yes
```

Then you can simply ssh by:

```
ssh slurm
```

3.6 Troubleshooting

If you have trouble connecting to one of our systems, please run the SSH client with verbose output:

```
ssh -vvv -i .ssh/id_ed25519 ml-cloud-user-id@IP_OF_LOGIN_NODE
```

Send the resulting output to the support team at support@mlcloud.uni-tuebingen.de with a description of your problem and we will try to fix the issue for you.

3.7 Linux Shell

The default login shell for your user account is Bash. To determine your current login shell, execute:

```
$ echo $SHELL
```

If you'd like to change your login shell to `tcsh`, `sh`, or `zsh`, submit a ticket through the support system.

When you start a shell on the ML Cloud, system-level startup files initialize your account-level environment and aliases before the system sources your own user-level startup scripts. You can use these startup scripts to customize your shell by defining your own environment variables, aliases, and functions. These scripts (e.g. `.profile` and `.bashrc`) are generally hidden files: so-called dotfiles that begin with a period, visible when you execute: `ls -a`.

4. INTRODUCTION

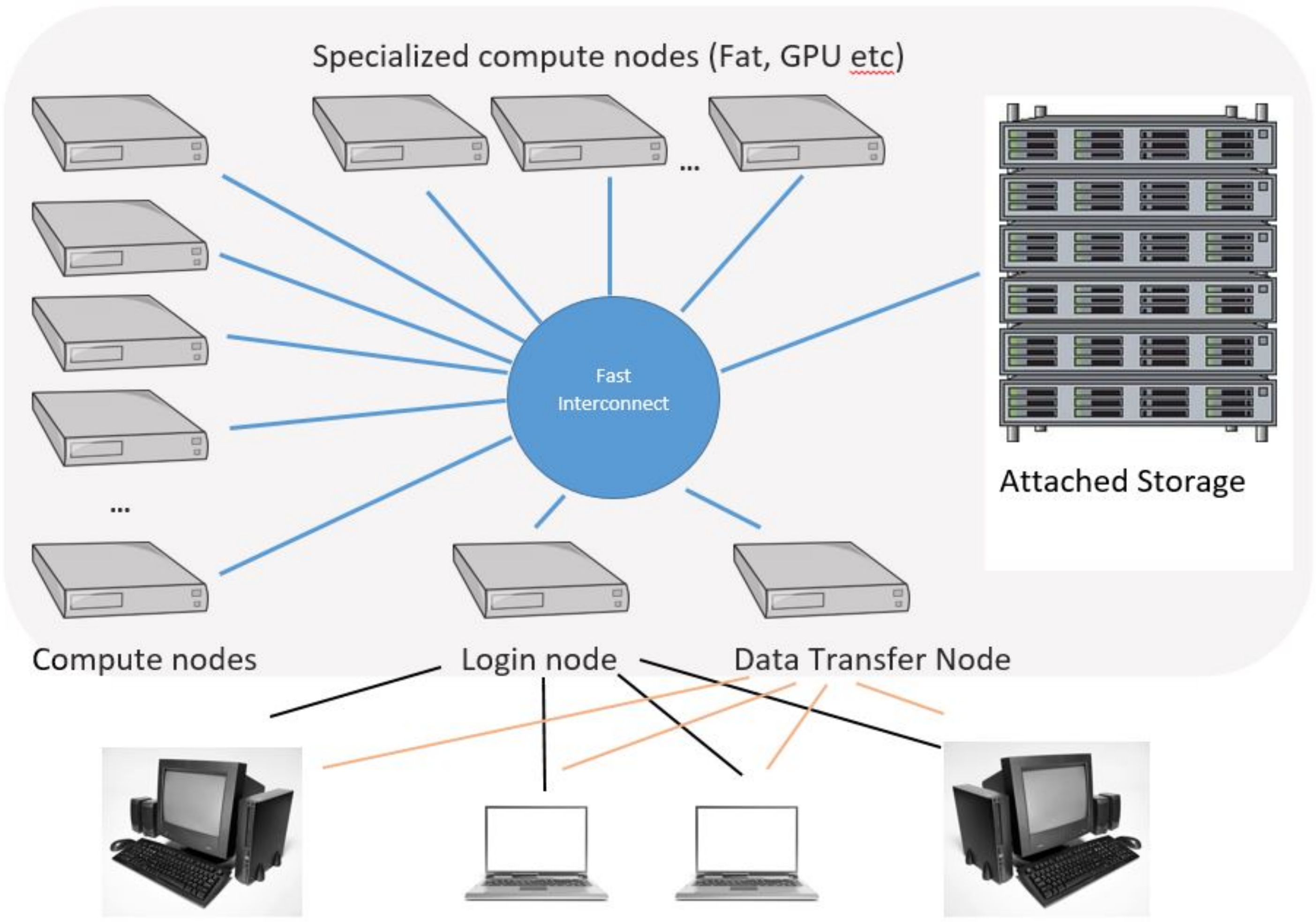
This section provides brief introductory information about what is a cluster, what is the composability of the ML Cloud.

4.1 Brief Introduction to Clusters

A cluster is a collection of computers (often referred to as "nodes"). They're networked together with some shared storage and a scheduling system that lets people run programs on them without having to enter commands "live".

There may be different types of nodes for different types of tasks. Generally, each cluster will have:

- **Login nodes:** one or more login nodes for users log in.
- **Storage Nodes:** where data is stored and transfered from for computation
- **Compute nodes:** those can be variety of different node types, some of which are:
    - **regular compute nodes:** with CPU and memory
    - **fat compute nodes:** with more memory
    - **GPU nodes:** on these nodes computations can be run both on CPU cores and on a Graphical Processing Unit)
- **Interconnect:** switches, cables and network cards that connect the nodes, storage together and provide access to the users.



4.2 The ML Cloud

The ML Cloud is composed of hardware suitable for AI based workloads. We provide variety of node types:

1. Traditional CPU compute nodes,
2. Traditional CPU compute nodes with large memory,
3. GPU nodes with Nvidia RTX 2080ti accelerator cards
4. GPU nodes with Nvidia V100 accelerator cards
5. GPU nodes with Nvidia A100 accelerator cards
6. GPU nodes with Nvidia H100 accelerator cards

Due to cooling capacity limitations our clusters are installed in georgraphically different physical locations, which are not connected to one another.

5. NVIDIA GPUS AT ML CLOUD

5.1 Physical architecture

Within ML Cloud, we have four types of physical GPU cards, all made by Nvidia:

- RTX 2800 Ti (PCIe) (Full information)
- V100 (Mezzanine) (Full information)
- A100 (PCIe) (Full information)
- H100 (Mezzanine) (Full information)

5.2 What's inside a GPU?

Each GPU is made of a full GPU chip, each having many SM (Streaming Multiprocessors). Within each SM are several components - including shared memory, registers, schedulers, etc. - but the most relevant component for most applications is the compute cores. Our GPUs have a variety of different amounts and kinds of compute cores within their SMs, each designed to be optimal for a certain kind of datatype (see below).

- INT32 cores are optimal for 32-bit integers.
- FP32 cores are optimal for 32-bit floating-point operations. (aka "single precision")
- FP64 cores are optimal for 64-bit floating-point operations. (aka "double precision")
- Tensor cores vary in capability for each GPU type, but provide very high performance for e.g. FP8, FP16, INT16, BF16, and other datatypes often relevant for AI applications.

5.3 Why is datatype important?

- Computational numbers are stored in registers of specific datatypes, and GPUs are optimized for specific datatypes.

Datatype is how a number is stored in computer memory. The number of bits determines how many different numbers are available. An 8-bit datatype has 2^8 = 256 possible different numbers, stored in 8 bits in memory. A 16-bit datatype has 2^16 = 65536 possible different numbers, stored in 16 bits in memory.

Computer memory stores numbers for calculation in a *register* of a specific datatype. A computer core performs a numerical operation on registers. While a CPU core is able to deal with a variety of different data types for general processing, a GPU core is designed to only address registers of specific sizes. Hence, a FP32 core is optimized to only perform operations on 32-bit floating point numbers.

Therefore, GPUs are most effective at performing operations on specific kinds of datatypes matching their cores. *For example, the RTX 2080 Ti is very bad for processing 64-bit floating point numbers, because it does not have cores specialized for FP64 - the V100, A100, and H100 all do.*

5.4 What is a Tensor Core?

- Nvidia GPUs include "Tensor Cores" designed to handle AI-relevant lower-precision datatypes (such as FP8) at extremely high speeds. More advanced Tensor cores also handle specific operations on other datatypes.

Traditionally, HPC applications have used 32-bit and 64-bit datatypes. Recently, lower-precision datatypes have become increasingly important for their applications in AI algorithms (among others). This is a field of rapid ongoing improvement, so GPUs vary not just in performance, but which datatypes are supported. More advanced Tensor cores are higher performance and support more datatype.

As Tensor cores have become more advanced, they have also increasingly gained specialized functionality for specific mathematical operations on different datatypes.

The exact benchmarks of the Tensor Cores within the ML Cloud's GPUs are currently being determined, and this document will be updated when they are known.

- *Generally, the RTX 2800Ti has basic Tensor cores, the V100 has intermediate Tensor cores, the A100 has advanced Tensor cores, and the H100 has cutting-edge Tensor cores - these vary in both performance and datatype support.*

5.5 Not all cores are equal

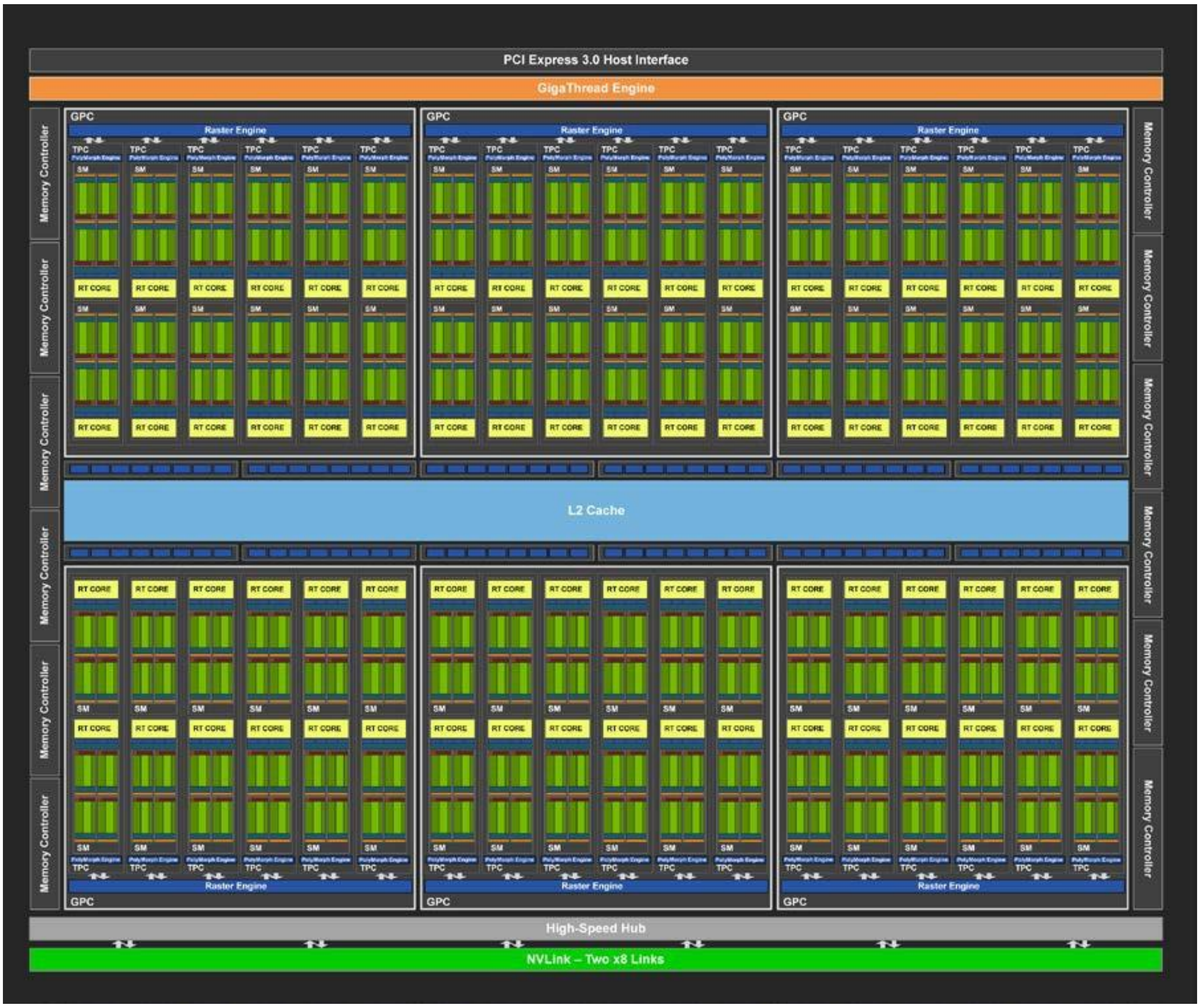
- Due to performance increases from i.a. decreasing lithographic feature sizes, newer GPUs are able to perform more calculations using the same amount of power. A smaller transistor generally requires fewer electrons to operate, so more advanced chips can perform better even if they have the same number of cores or the same power limit.

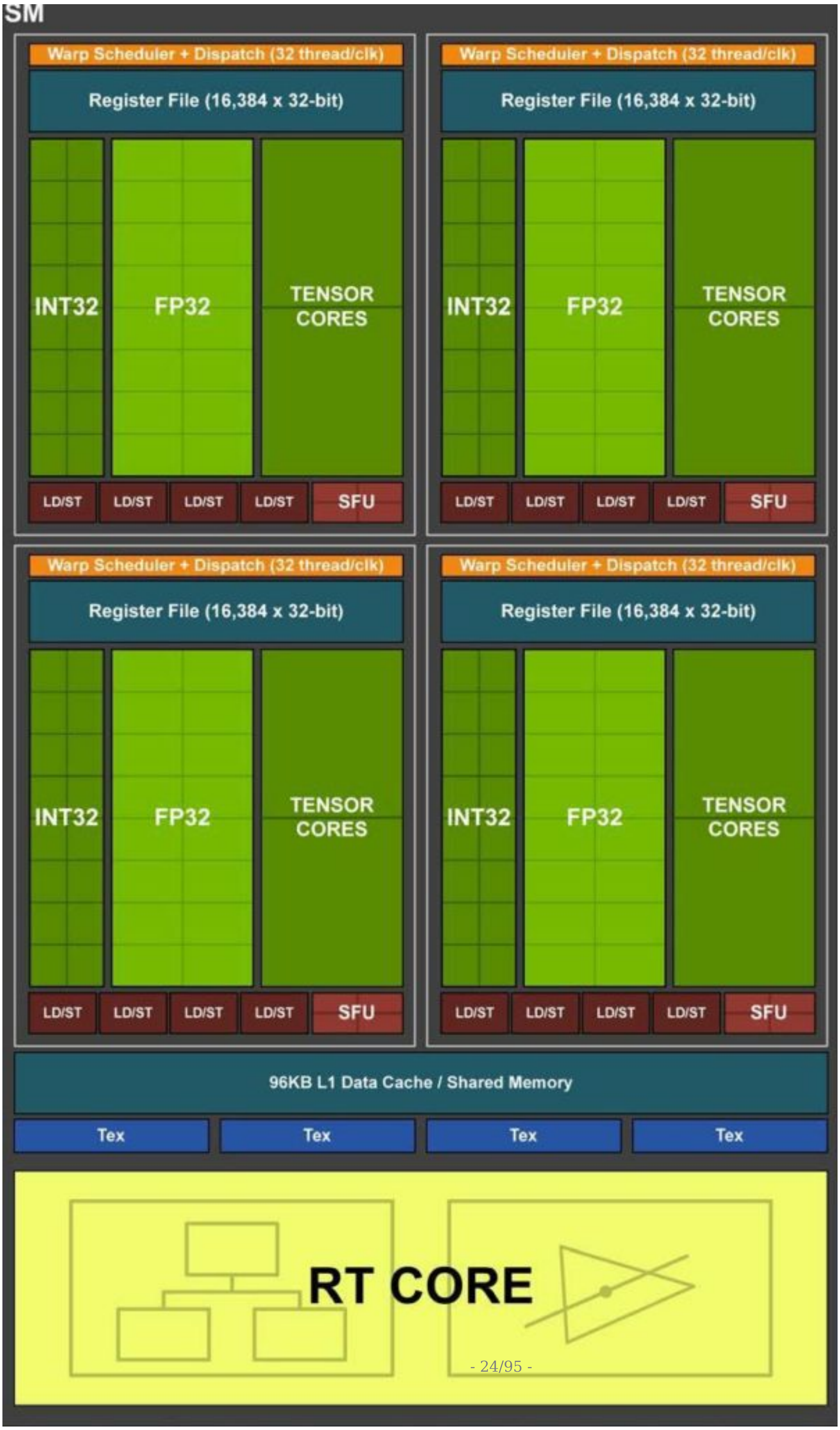
Therefore, newer GPUs will often be faster than older ones.

5.6 Comparison of physical architecture

- The following diagrams from Nvidia show visually the differences in each GPU's full-chip and SM. This shows the basic differences in complexity between the different GPU cards.
- **Important** - Not all cores are created equal. We are still determining the exact benchmarks, but, in general, the RTX 2800Ti cores will be slower than the V100 cores, which will be slower than the A100 cores, which will be slower than the H100 cores.

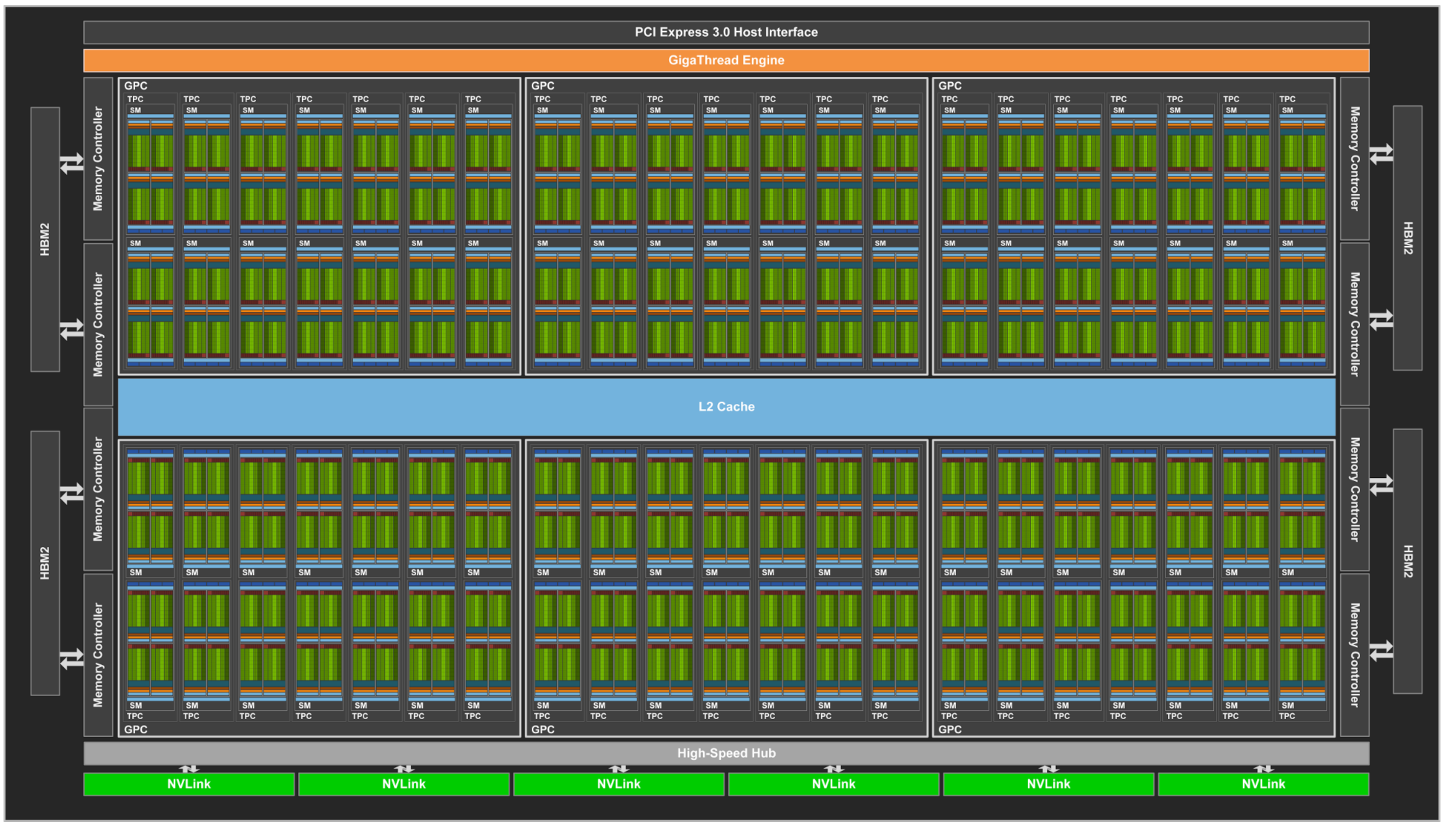
5.6.1 RTX 2800 Ti

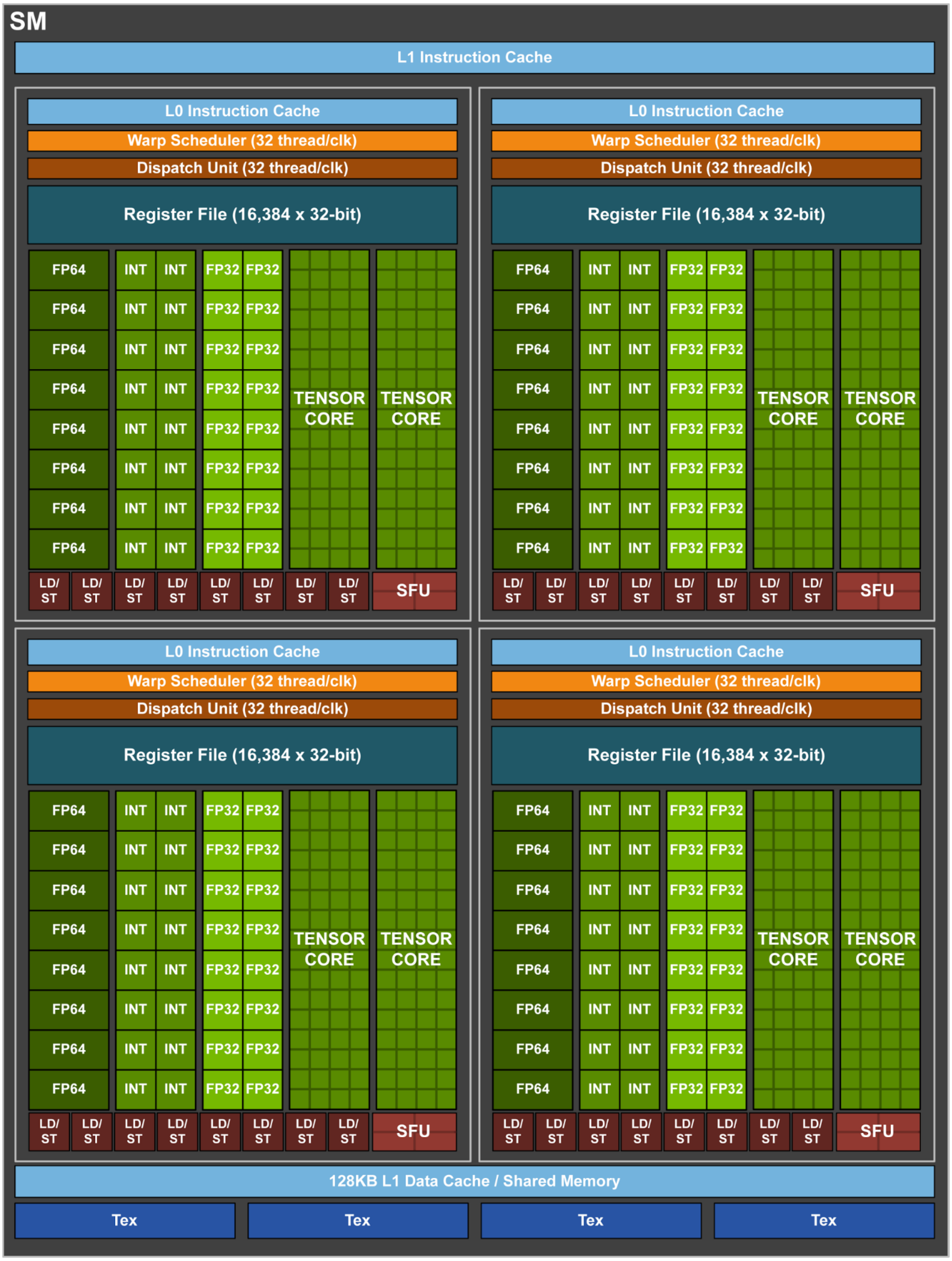

PCI Express 3.0 Host Interface
GigaThread Engine
GPC
Raster Engine
TPC
PolyMorph Engine
SM
RT CORE
Memory Controller
L2 Cache
High-Speed Hub
NVLink – Two x8 Links


SM
Warp Scheduler + Dispatch (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
TENSOR CORES
LD/ST
SFU
96KB L1 Data Cache / Shared Memory
Tex
RT CORE

The RTX 2800 Ti GPU has 68 SMs. Within each SM are 64 FP32, 64 INT32, *zero* FP64, and 8 basic Tensor cores.

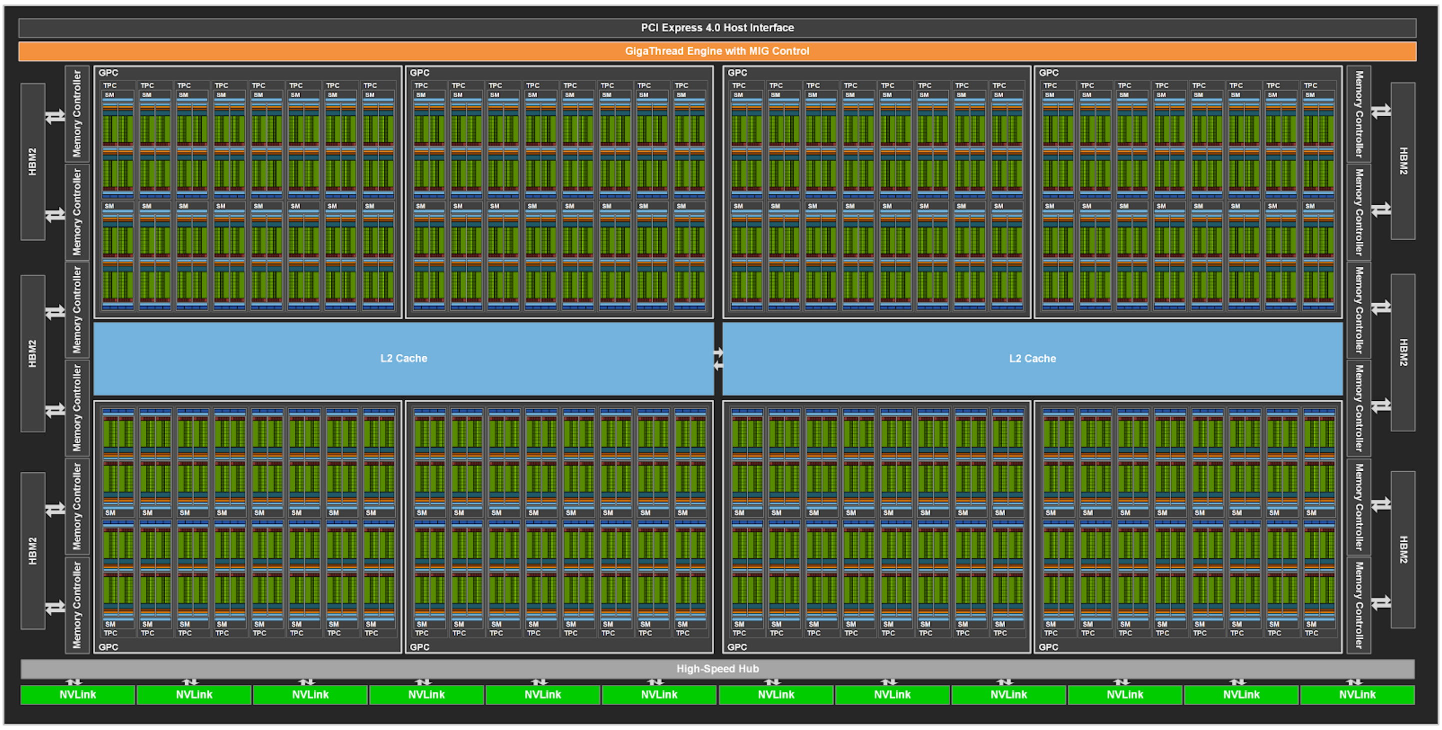
5.6.2 V100

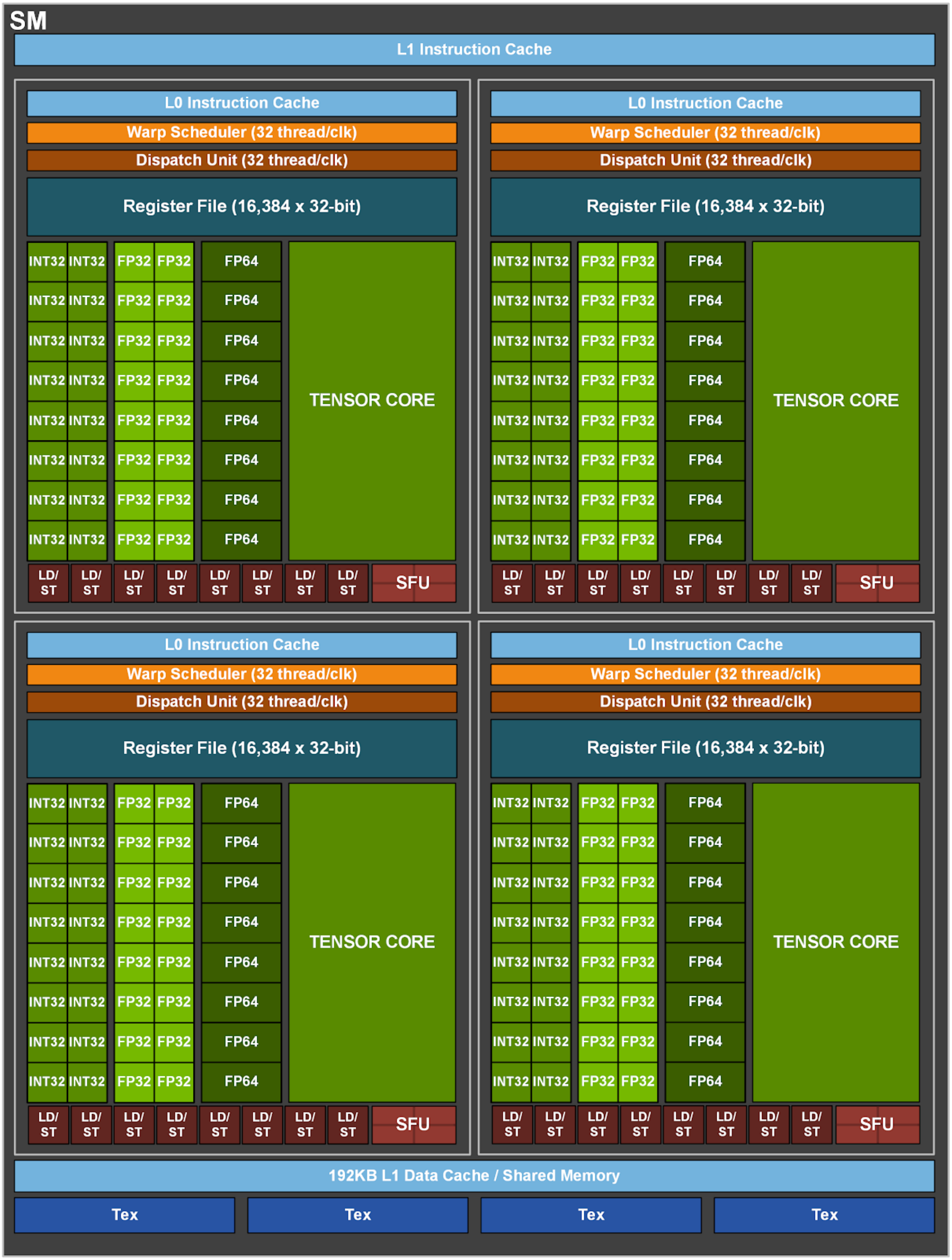

PCI Express 3.0 Host Interface
GigaThread Engine
GPC
TPC
SM
Memory Controller
HBM2
L2 Cache
High-Speed Hub
NVLink


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
FP64
INT
FP32
TENSOR CORE
LD/ST
SFU
128KB L1 Data Cache / Shared Memory
Tex

The V100 GPU has 84 SMs. Within each SM are 64 FP32, 64 INT32, 32 FP64, and 8 intermediate Tensor cores.

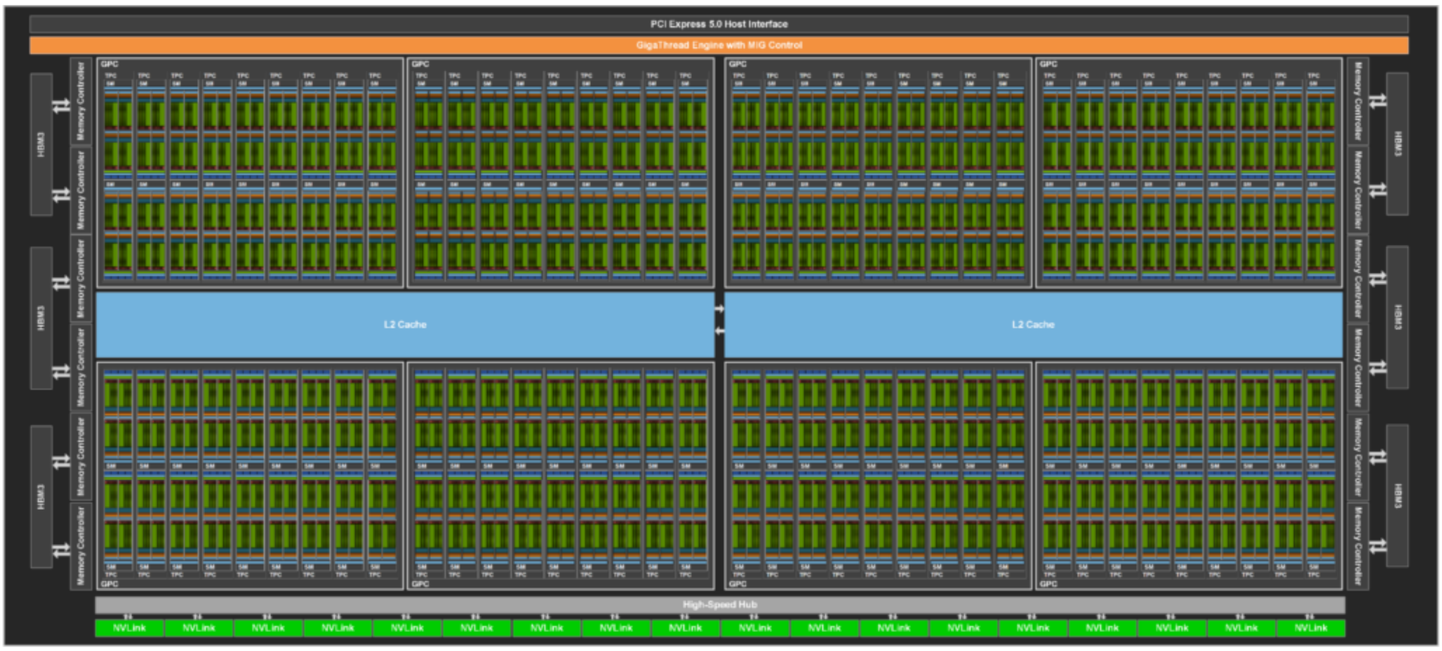
5.6.3 A100

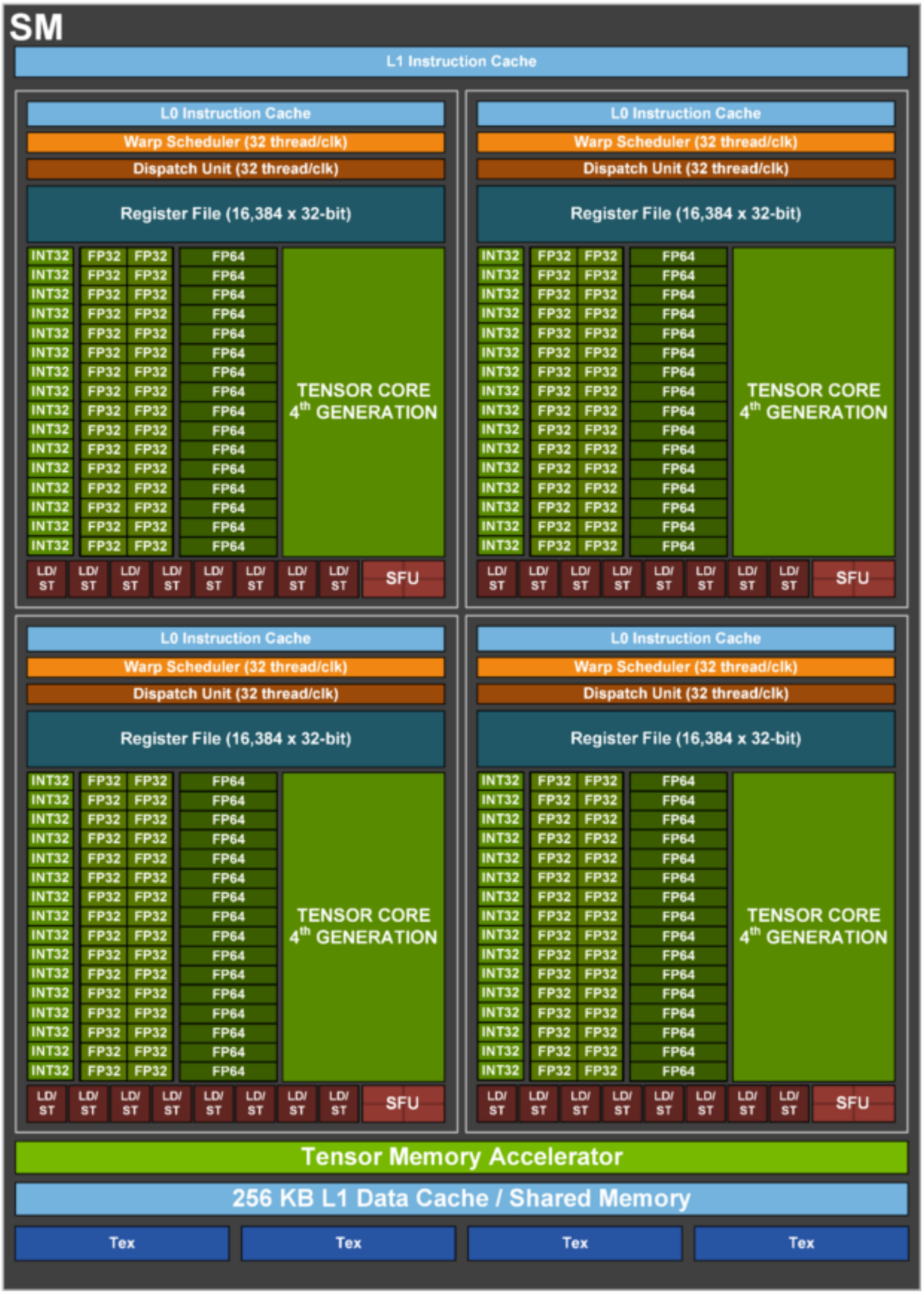

PCI Express 4.0 Host Interface
GigaThread Engine with MIG Control
GPC
TPC
SM
L2 Cache
Memory Controller
HBM2
High-Speed Hub
NVLink


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
TENSOR CORE
LD/ ST
SFU
192KB L1 Data Cache / Shared Memory
Tex

The A100 GPU has 108 SMs. Within each SM are 64 FP32, 64 INT32, 32 FP64, and 4 advanced Tensor cores.

5.6.4 H100


PCI Express 5.0 Host Interface
GigaThread Engine with MIG Control
GPC
TPC
SM
L2 Cache
Memory Controller
HBM3
High-Speed Hub
NVLink


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
TENSOR CORE 4th GENERATION
LD/ ST
SFU
Tensor Memory Accelerator
256 KB L1 Data Cache / Shared Memory
Tex

The H100 GPU has 144 SMs. Within each SM are 128 FP32, 64 INT32, 64 FP64, and 4 cutting-edge Tensor cores.

6. JOBS SCHEDULING WITH SLURM

The ML Cloud can be accessed through a dedicated set of login nodes used to write and compile applications as well as to perform pre- and post-processing of simulation data. Access to the compute nodes in the system is controlled by the workload manager.

On the ML Cloud the Slurm (Simple Linux Utility for Resource Management) Workload Manager, a free open-source resource manager and batch system, is employed. Slurm is a modern, extensible batch system that is widely deployed around the world on clusters of various sizes.

This page describes how you can run jobs and what to consider when choosing SLURM parameters. You submit a job with its resource request using SLURM, SLURM allocates resources and runs the job, and you receive the results back. There are interactive modes available. Slurm:

- It allocates exclusive or non-exclusive access to the resources (compute nodes) to users during a limited amount of time so that they can perform they work
- It provides a framework for starting, executing and monitoring work
- It arbitrates contention for resources by managing a queue of pending work.
- It permits to schedule jobs for users on the cluster resource

A Slurm installation consists of several programs and daemons. The `slurmctld` daemon is the central portion of the batch system responsible for monitoring the available resources and scheduling batch jobs. The `slurmctld` runs on an management node with a special setup to ensure availability in the case of hardware failures. Most user programs such as `srun`, `sbatch`, `salloc` and `scontrol` interact with the `slurmctld`. For the purpose of job accounting `slurmctld` communicates with the `slurmdbd` database daemon. Information from the accounting database can be queries using the `sacct` command. Slurm combines the functionality of the batch system and resource management. For this purpose Slurm provides the `slurmd` daemon which runs on the compute nodes and interacts with `slurmctld`.

6.1 Slurm Partitions

In Slurm multiple nodes can be grouped into partitions which are sets of nodes aggregated by shared characteristics or objectives, with associated limits for wall-clock time, job size, etc. These limits are hard limits for the jobs and can not be overruled. In practice, these partitions can be used by the user to signal a need for resources that have certain hardware characteristics (cpu-only, large memory, GPU type) or that are dedicated to specific workloads (large production jobs, small debugging jobs, interactive, etc.). ML Cloud has implemented several production queues.

See Galvani's Partitions See Ferranti's 1's Partitions

6.2 Preemptable partitions

These are cheap partitions where jobs will only cost 25% compared to their non-preemptable counterparts. But in these partitions your job may be canceled or requeued if a job in a non-preemptable partition requires resources. Preempted jobs are requeued by default. When a job is requeued, the batch script is initiated from its beginning. If you do not desire this please set the `--no-requeue` sbatch option in your job submission scripts. If you set this option your job will just be cancelled.

6.3 Job cost

The cost of your job is calculated as the maximum of the cost of the resources you consume. For example if you submit job to the **gpu-2080ti** partition that used 10 CPUs, 50G RAM, 1 GPU then:

cost=MAX(10 * 0.278,50 * 0.0522,1 * 2.5)=2.78

These costs will be used to calculate your fairshare.

Accounting and fairshare will be based on the amount of resources you are blocking and not on what you reserve: E.g. requesting `#v100=1` and `#cpu=64` will still charge you the equivalent of a whole `V100` node.

6.4 Quality of Service (QoS)

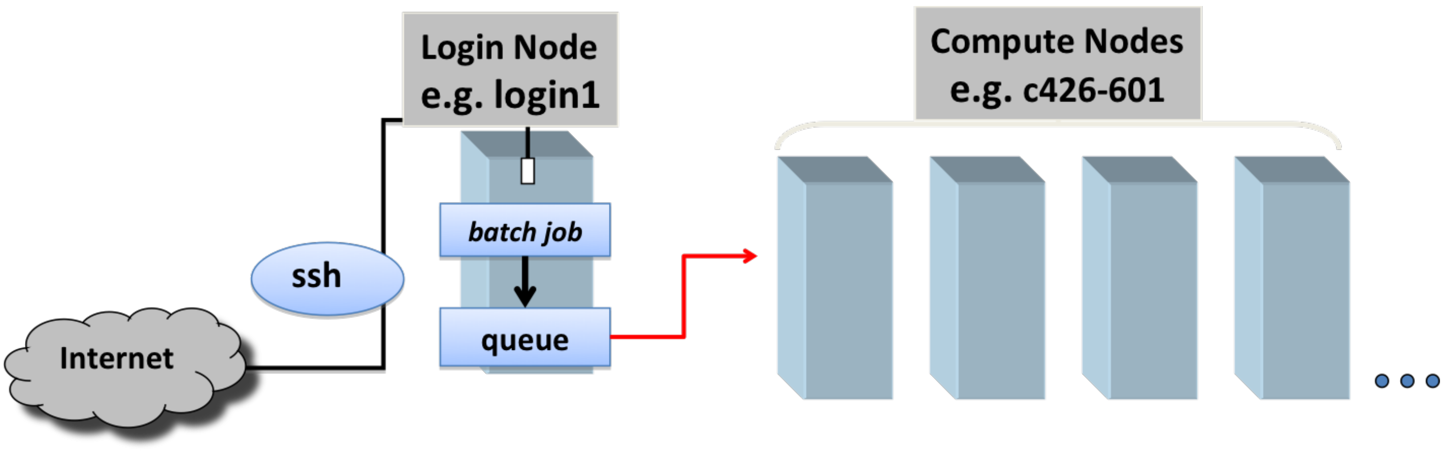
Quality of Services are the most versatile way of setting specific privileges, but also limits, to users and jobs, dictating the limit in the resources and partitions that a job is entitled to request.

Each partition of the clusters can be associated with a QoS, from which it will inherit all limits.

6.5 Login Nodes

When you login to the ML Cloud system you land on a login node. The login nodes are shared resources: at any given time, there are many users logged into each of these login nodes, each preparing to access the "back-end" compute nodes (Figure 2. Login and Compute Nodes). What you do on the login nodes affects other users directly because you are competing for the same resources: memory and processing power. This is the reason you should not run your applications on the login nodes or otherwise abuse them. Think of the login nodes as a prep area where you can manage files and compile code before accessing the compute nodes to perform research computations. See Good Conduct for more information.

Figure 2. Login and Compute Nodes



Login and Compute Nodes

To discern whether you are on a login node or a compute node use the command-line prompt, or the `hostname` command. The hostname for a ML Cloud login node has a string `login` (e.g. `galvani-login`), while compute node hostnames will start with the name of the cluster and a number (e.g. `galvani-cn`).

6.6 Allocations, Jobs and Job Steps

In Slurm a job is an allocation of selected resources for a specific amount of time. A job allocation can be requested using `sbatch` and `salloc`. Within a job multiple job steps can be executed using `srun` that use all or a subset of the allocated compute nodes. Job steps may execute at the same time if the resource allocation permits it. A `user job` is characterized by: * number of computing resources: nodes (including all their CPUs and cores) or CPUs (including all their cores) or cores * amount of memory: either per node or per CPU * (wall)time needed for the users tasks to complete their work * the launcher script, which will initiate your tasks

There are everal ways of submitting jobs with slurm, using either `sbatch`, `srun` or `salloc`:

1. Submit a **batch job** using the `sbatch` command. This directs the scheduler to run the job unattended when there are resources available. Until your batch job begins it will wait in a queue. You do not need to remain connected while the job is waiting or executing. Note that the scheduler does not start jobs on a first come, first served basis; it juggles many variables to keep the machine busy while balancing the competing needs of all users. The best way to minimize wait time is to request only the resources you really need: the scheduler will have an easier time finding a slot for the two hours you need than for the 24 hours you unnecessarily request.
2. Begin an interactive session using `ssh` to connect to a compute node on which you are already running a job. This is a good way to open a second window into a node so that you can monitor a job while it runs.
3. Begin an **interactive session** using `srun`. This will log you into a compute node and give you a command prompt there, where you can issue commands and run code as if you were doing so on your personal machine. An interactive session is a great way to develop, test, and debug code. The `srun` command submits a new batch job on your behalf, providing interactive access once the job starts. You will need to remain logged in until the interactive session begins.
4. `salloc` is used to allocate resources for a job in real time. Typically this is used to allocate resources (nodes, tasks, partition, etc.) and spawn a shell. The shell is then used to execute srun commands to launch parallel tasks.

6.7 Job submission options

There are several useful environment variables set be Slurm within an allocated job. The most important ones are detailed in the below table which summarizes the main job submission options offered with {`sbatch` | `srun` | `salloc`}.

You can pass options using either the command line or job script; most users find that the job script is the easier approach. Slurm directives begin with `#SBATCH`; most have a short form (e.g. `-N`) and a long form (e.g. `--nodes`).

Option	Argument	Comments
`--partition`	*queue_name*	Submits to queue (partition) designated by *queue_name*
`-nodes`	*nodes*	Define, how many nodes you need.
`--ntasks`	*tasks*	Number of tasks
`--gres`	*gpu:N*	request GPUs type and number of resouces
`--job-name`	*job_name*	Give your job a name, so you can recognize it in the queue overview
`--output`	*output_file*	Direct job standard output to *output_file* (without `-e` option error goes to this file. Make sure this is not on `$HOME`
`--error`	*error_file*	Direct job error output to *error_file*. Make sure this is not on `$HOME`
`--time`	*D-HH:MM*	Wall clock time for job.
`--mem`	*#G*	Memory pool for all cores (see also `--mem-per-cpu`)
`--mail-type`	`BEGIN`, `END`, `FAIL`, `ALL`	Specify when user notifications are to be sent (one option per line).
`--mail-user`	*user@uni-tuebingen.de*	Email to which notifications will be sent
`--constraint`	`ImageNetC`, `ImageNet2012`, `ImageNetR`, `ffcvImageNet`, `nodata`	deployed some commonly used datasets locally on comput nodes on select partitions.

By default, Slurm writes all console output to a file named `slurm-%j.out`, where `%j` is the numerical job ID. To specify a different filename use the `-o` option. To save `stdout` (standard out) and `stderr` (standard error) to separate files, specify both `-o` and `-e`.

6.8 Writing a Batch Script

Users submit batch applications (usually bash scripts) using the `sbatch` command. The first line of your job script must specify the interpreter that will parse non-Slurm commands; in most cases `#!/bin/bash` or `#!/bin/csh` is the right choice. All `#SBATCH` directives must precede all shell commands. Note also that certain `#SBATCH` options or combinations of options are mandatory (see table above for the list of common environmental variables):

```
#!/bin/bash #SBATCH --ntasks=1 # Number of tasks (see below) #SBATCH --cpus-per-task=1 # Number of CPU cores per task #SBATCH --
```

6.9 Job Script Examples (MORE ADDED HERE)

Once you have created your script, you can use Slurm's `sbatch` command to submit a batch job to one of the ML Cloud queues:

```
sbatch -p 2080-galvani test.sh
```

Here `test.sh` is the name of a text file containing `#SBATCH` directives and shell commands that describe the particulars of the job you are submitting. The details of your job script's contents depend on the type of job you intend to run. `-p 2080-galvani` specifices the partition (queue) under which the script will be run.

In each job script:

1. use `#SBATCH` directives to request computing resources (e.g. 10 nodes for 2 hrs);
2. then, list shell commands to specify what work you're going to do once your job begins.

There are many possibilities: you might elect to launch a single application, or you might want to accomplish several steps in a workflow. You may even choose to launch more than one application at the same time. The details will vary, and there are many possibilities. But your own job script will probably include at least one launch line that is a variation of one of the examples described here.

See the customizable job script examples.

Your job will run in the environment it inherits at submission time; this environment includes the specific execution directives and the current working directory. You can of course use your job submission script to modify this environment by defining new environment variables; changing the values of existing environment variables; changing directory; or specifying relative or absolute paths to files.

Consult the Common `sbatch` Options table below describes some of the most common `sbatch` command options. Slurm directives begin with `#SBATCH`; most have a short form (e.g. `-N`) and a long form (e.g. `--nodes`). You can pass options to `sbatch` using either the command line or job script; most users find that the job script is the easier approach.

6.10 Common SLURM commands

Command	Function
`sbatch`	Submit a batch job script. The command exits immediately when the script is transferred to the Slurm controller daemon and assigned a Slurm job ID.
`sacct`	Used to query past jobs.
`squeue`	Print table of submitted jobs and their state.
`sinfo`	Provide overview of cluster status.
`scancel`	Cancel a job prior to its completion.
`seff jobid`	Reports the computational efficiency of your calculations.
`sacctmgr`	with `show associations user=username` find out what account(s) your usesrname is associated with.
`scontrol show partitions`	detailed information about all available partitions and their definition/limits.
`sprio -w`	The weights for the prioritization can be found by running the `sprio -w` command.
`sshare`	The command shows how many shares your group has as well as your fairshare value

6.11 Cluster status

The command sinfo provides status information of nodes in various partitions (also the associated time limit for each partition). The default partition is marked with an "`*`". This information can be useful in deciding where to submit your job. Status codes (abbreviated form) are explained below:

Status Code	Description
`alloc`	The node has been allocated to one or more jobs.
`mix`	The node has some of its CPUs ALLOCATED while others are IDLE.
`idle`	The node is not allocated to any jobs and is available for use.
`down`	The node is down and unavailable for use.
`drain`	The node is unavailable for use per system administrator request. (for maintenance etc.)
`drng`	The node is being drained but is still running a user job. The node will be marked as drained right after the user job is finished. Do not worry if you have a job running on a node with this state.

6.12 Monitor Your Job

Once submitted, the job will be queued for some time, depending on how many jobs are presently submitted. Eventually, more or less after previously submitted jobs have completed, the job will be started on one or more of the nodes determined by its resource requirements. The status of the job can be queried with the `squeue` command.

Option	Description
`-a`	Display information for all jobs.
`-j jobid`	Display information for the specified job ID.
`-j jobid -o %all`	Display all information fields (with a vertical bar separating each field) for the specified job ID.
`-l`	Display information in long format.
`-n job_name`	Display information for the specified job name.
`-t state_list`	Display jobs that have the specified state(s). Valid jobs states include `PENDING`, `RUNNING`, `SUSPENDED`, `COM` `DEADLINE`, `OUT_OF_MEMORY`, `COMPLETING`, `CONFIGURING`, `RESIZING`, `REVOKED`, and `SPECIAL_EXIT`.
`-u username`	Display jobs owned by the specified user.

For example, to see pending jobs for a particular user:

```
squeue -u mfa608 -t PENDING
```

You can use `sacct` to get details of a previously run job:

```
sacct -j 15370
```

or

```
sacct -j 15370 --format JobID,JobName,Partition,Account,AllocCPUS,State,ExitCode,NodeList
```

Be sure to distinguish between internal Slurm replacement symbols (e.g. `%j` described above) and Linux environment variables defined by Slurm (e.g. `SLURM_JOBID`). Execute `env | grep SLURM` from within your job script to see the full list of Slurm environment variables and their values. You can use Slurm replacement symbols like `%j` only to construct a Slurm filename pattern; they are not meaningful to your Linux shell. Conversely, you can use Slurm environment variables in the shell portion of your job script but not in an `#SBATCH` directive. For example, the following directive will not work the way you might think:

```
~~#SBATCH -o myProgram.o${SLURM_JOB_ID}~~   # incorrect
```

Instead, use the following directive:

```
#SBATCH -o myProbgram.o%j # "%j" expands to your job's numerical job ID
```

For more information on this and other matters related to Slurm job submission, see the Slurm online documentation; the man pages for both Slurm itself (`man slurm`) and its individual commands (e.g. `man sbatch`); as well as numerous other online resources.

6.13 Using srun --pty bash

`srun` uses most of the options available to sbatch. When the interactive allocation starts, a new bash session will start up on one of the granted nodes:

```
[mfa608@galvani-login ~]$ srun --job-name "InteractiveJob" --ntasks=1 --nodes=1 --time 1:00:00 --pty bash
```

6.14 Using Salloc

`salloc` functions similar to `srun --pty bash` in that it will add your resource request to the queue. However once the allocation starts, a new bash session will start up on the login node. To run commands on the allocated node you need to use `srun`.

If you connection is lost for some reason you can use `salloc --no-shell` to resume shell/ jobs sessions.

7. CONTACT

7.1 Goal

Our Goal is to assist researchers from the AI Center and ML Cluster of Excellence effectively using the ML Cloud Infrastructure by creating support environment for any user.

7.2 Our Team

- Kristina Kapanova: kristina.kapanova(at)uni-tuebingen.de
- Robert Pennington: robert.pennington(at)uni-tuebingen.de

7.3 ZDV ML Cloud Coordinator (Interface Position between ZDV and ML Cloud)

- Benjamin Gläßle: benjamin.glaessle(at)uni-tuebingen.de
- Rasool Almasikoupaei: rasool.almasikoupaei(at)uni-tuebingen.de

7.4 Ticketing

Submit a ticket for a problem with the system you have at support@mlcloud.uni-tuebingen.de

7.5 Office Hours

See our office hours here and come for a consultation.

8. KNOWN ISSUES

This page collects known issues affecting the ML Cloud systems and application software.

Note

The following list of known issue is intended to provide a quick reference for users experiencing problems on the ML Cloud systems. We strongly encourage all users to report the occurrence of problems, whether listed below or not, to ML Cloud Support.

8.1 GPU nodes may crash but still be available for scheduling jobs

Added: 2023-01-28

Affects: Galvani

Description: the GPU nodes may not find the GPUs on the nodes. This results in an immediate `CUDA not found` error and unfortunately, many jobs may get send to this node since it appears free.

Status: Open

Workaround/Suggested Action: Open a ticket. The node needs to be restarted and only the ML Cloud admins can do this.

8.2 Slurm Configuration File

Added: 2023-01-28

Affects: Galvani

Description: This can happen during slurm package updates. You will see an error like:

```
sbatch: error: resolve_ctls_from_dns_srv: res_nsearch error: Unknown host sbatch: error: fetch_config: DNS SRV lookup failed sba
```

Workaround/Suggested Action: Just restart the current shell (log out and log in again) or use export `SLURM_CONF=/etc/slurm/slurm.conf` to set the path manually.

9. FREQUENTLY ASKED QUESTIONS

Here you can find a list of frequently askes questions about the ML Cloud. Such a list comes from user problems, submitted through the ticketing system. We will periodically update the FAQ section with such information.

9.1 I am experiencing login issues

One possible problem for you experiencing login issues can be due to maxed out quota on `$HOME`. Please do **NOT** store input data, results or models on `$HOME`. Please do NOT let your jobs write to `$HOME`. If this happens, send us a ticket to the ML Cloud Team - we will temporarily increase your quota, so you can log in and clean space.

9.2 My SLURM job should be showing live output, but it isn't.

This is due to output buffering, so you need to make sure your program flushes the buffer. This problem can happen with Python inside or outside a container, and potentially with other programs as well. To fix this, where you've written `python script.py`, change it to `python -u script.py`. For other programs, search for how to flush the stdout buffer.

9.3 I can't activate `conda` in a SLURM job

Either write out the explicit path to the conda_env executable (`/mnt/lustre/.../conda_envs/my_env/bin/python`) or include a `source ~/.bashrc` line in your `sbatch` script before you try `conda activate ...`. You can also check how to do this from the Pytorch and conda tutorial. Same applies for Tensorflow.

9.4 An individual node is having problems that other nodes are not experiencing

To fix your SLURM job to a specific compute node, use `--nodelist=galvani-cn101,galvani-cn104` etc. To exclude specific nodes, use `--exclude=galvani-cn108,galvani-cn109`. Once you've verified that your job crashes on a specific node or nodes but works on others, please report it as a bug, including with the node-specific information. The admins will need the exact commands you used; please include those in your bug report, as well as log locations and if the admins can use your account to diagnose the problem. Possible admin fixes may include `--gpu-reset`.

9.5 csh availability

`csh` is not available. Recently we have provided `tcsh` installation on all slurm nodes.

9.6 I don't see a specific package, why?

The current strategy of the ML Cloud Team is that unless a package is required, we do not install by default. Thus, if you have a particular need, please open a request for package through the ticketing system.

9.7 How to change my default shell to zsh

Please contact the support team with your request and list the username and shell you want, so we can change your default shell.

9.8 Problem accessing the login nodes and receiving unsuccessful login attempt with partially the following information

```
..... debug1: No credentials were supplied, or the credentials were unavailable or inaccessible No Kerberos credentials availabl
```

This probably pertains to a client problem on login node. Inform the ML Cloud Team so we can fix the client problem.

9.9 Problem with your conda

If you see an error such as this one below:

```
Collecting package metadata (repodata.json): failed >>>>>>>>>>>>>>>>>>>>>> ERROR REPORT <<<<<<<<<<<<<<<<<<<<<< Traceback (most r
```

you should remove `~/.condarc` from your directory and initiate again.

9.10 Possible QB-client mount

If you receive error similar to the ones below:

```
[user@galvani-login ~]$ srun --job-name "InteractiveJob" --ntasks=1 --nodes=1 --time 1-00:00:00 --gres=gpu:1 --pty bash slurmste
```

or

```
/var/spool/slurmd/job1234/slurm_script: line 20: /home/group/user/.bashrc: Transport endpoint is not connected
```

this is probably related to qb-client mounting error on the node where your job has been submitted to. Please open a ticket and we will fix the problem.

9.11 Strange wget error

If you encounter strange `wget` errors simply delete your `~/.wget-hsts` which will solve the issue.

10. GALVANI SYSTEM ARCHITECTURE

This page describes the infrastructure that makes up the Galvani Cluster.

10.1 Login Nodes

Currently there is one login node for galvani:

`134.2.168.43`

with the following configuration:

Feature	Specifications
CPUs:	2 x Intel Xeon Gold 16 cores, 2.9GHz
RAM:	1536GB (2933 MT/s) DDR4
Local Storage:	960GB SSD

10.2 Galvani: Compute Infrastructure

The **36** GPU compute nodes with the following specifications:

Feature	Specifications	Specifications
Total Nodes:	28	8
Accelerators:	8 Nvidia 2080ti / node	8 Nvidia A100 / node
CUDA Parallel Processing Cores:	4352 / card	3,456 (FP64), 6,912 (FP32) / card
NVIDIA Tensor Cores:	544 / card	432 / card
GPU Memory:	11GB GDDR6 (memory bus: 352 bit) / card	40GB HBM2 / card
CPUs:	2 x Intel Xeon Gold 6240, 18 cores, 2.6GHz	2 x AMD EPYC 7302, 16 cores, 3.0GHz
RAM:	384GB (3200 MT/s) DDR4	1TB (3200 MT/s) DDR4
Local Storage:	1,92TB	3,84TB
Theoretical Peak Performance:	266.79 TFLOPs/node	196.54 TFLOPs/node

There are additionally **3** CPU only nodes with the following composition:

Feature	Specifications
Total Nodes:	3
CPUs:	2 x Intel Xeon Gold 16 cores, 2.9GHz
RAM:	1536GB (2933 MT/s) DDR4
Local Storage:	960GB SSD
Theoretical Peak Performance:	

10.3 Galvani: Interconnect

The Galvani Cluster uses fat-tree non-blocking topology with ethernet. Most of the nodes have a 40Gb/s HCA. Storage is connected on a 100Gb/s.

11. AVAILABLE STORAGE IN GALVANI

This page describes how Galvani deals with data storage. Right now the storage solution for Galvani and Galvani is the same.

Note

We are decommissioning QB in 2024 with the storage being all NVMe Lustre. Beegfs will also be decommissioned.

11.1 On Storage Availability and Backups

Please note that storage space and services on Galvani is provided as is. We cannot guarantee high availability and back-up of any of our storage space, which refers to all storage on the ML Cloud.

11.2 Storage Quotas

Storage is a shared and limited resource and in a number of places we need to enforce quota to avoid that some script accidentally fills up the disk and the system becomes unusable for everybody.

Storage quota is specified in:

- **Space limit:** affects the aggregated size of all your files or files of a group. When this limit is reached you or the group cannot store more data (new data or increasing file sizes) on the system.

11.3 Quota applies to specific folders

Often it is intended that storage quota applies to a specific folder on the file system. For example, the so-called `HOME` quota shall apply to your home folder `$HOME`.

11.4 Quota applies to snapshots

It is important to remember that the file system snapshots that are created impact your storage quota. This means that if you truncate and then delete a file, the reduce in quota will not be visible immediately, but it will have a delayed effect of about 14 days.

11.5 QB Storage

The main storage backend for your data in Galvani is a Quobyte storage cluster. The only things that are not stored on the backend are the root disks of VMs and bare-metal machines as wells as ephemeral disks of VMs.

The following table summarizes the different storage options:

Table 2.1 ML Cloud File Systems For Galvani

File System	Quota	Disc Type	Filelocks	Key Features
`$HOME`	**20GB**, ### files	SSD	No implicit filelocks	**Not intended for parallel or high-intensity file operations**. **NOT** intended to store input data, results or models. **NOT** intended for usage to write jobs on `HOME` since such process might max out your quota and lead to inability to login.
`$WORK`	Initial Quota: **8TB**	SSD	No implicit filelocks	Volume `WORK` can hold `~50 mln` files. The environment variable is set to `/mnt/qb/work/YOUR_GROUP/YOUR_UID` for each user. This can be used to store input data and results with the quota limits in mind.
`$WORK2`	part of `WORK` quota	SSD	No implicit filelocks	`WORK2` is a different volume, but still part of your `WORK` quota
`$WORK3`	part of `WORK` quota	SSD	No implicit filelocks	`WORK3` is a different volume, but still part of your `WORK` quota
`$SCRATCH`	-	SSD	No implicit filelocks	should be used to access local, temporary storage on compute nodes. It is set to the folder `/scratch_local/$SLURM_JOB_USER-$SLURM_JOBID`. Please note that this folder and everything in it will be deleted after the job finishes.
`/mnt/qb/YOUR_GROUP`	**25TB**	HDD	No implicit filelocks	shared HDD volume for each group
`/mnt/qb/datasets`	-	SSD	No implicit filelocks	Accessible to all users. Read only on GPU nodes and writable for staging on login and cpu-only nodes.

Note

The separation of `WORK`, `WORK2`, `WORK3` in different volumes (while all three volumes comprise your group quota) was to spread the volumes for rebalancing purposes for the metadata.

11.6 QB Quotas and Freeing Space

The shared network volumes which provide `$HOME`, `$WORK` and `/mnt/qb/datasets` have `user` or `group` quotas in Galvani. You can query them with `qinfo quota` if your current working directory is part of those volumes.

There are two ways to delete/remove files and free quota:

- `rm file`: although this command will romove the file, the quota size decrease will be reflected with a delay of app. 2 weeks if files were present at the latest QB metadata snapshot.
- To free quota instantaneously use:

```
: > file rm file
```

for a single file. For deleting a directory recursively use:

```
find DIRECTORY -type f ! -size 0c | parallel -X --progress truncate -s0 rm -rf DIRECTORY
```

- using mv file `/ouside/of/volume/` might not free quota instantaneously. Instead use:

```
cp file /ouside/of/volume/ : > file rm file
```

Note

Truncating any files within your conda environments, dirs and packages most likely corrupts your environments and might required you to rebuild them from scratch.

12. GALVANI CEPH S3

12.1 Why use Ceph?

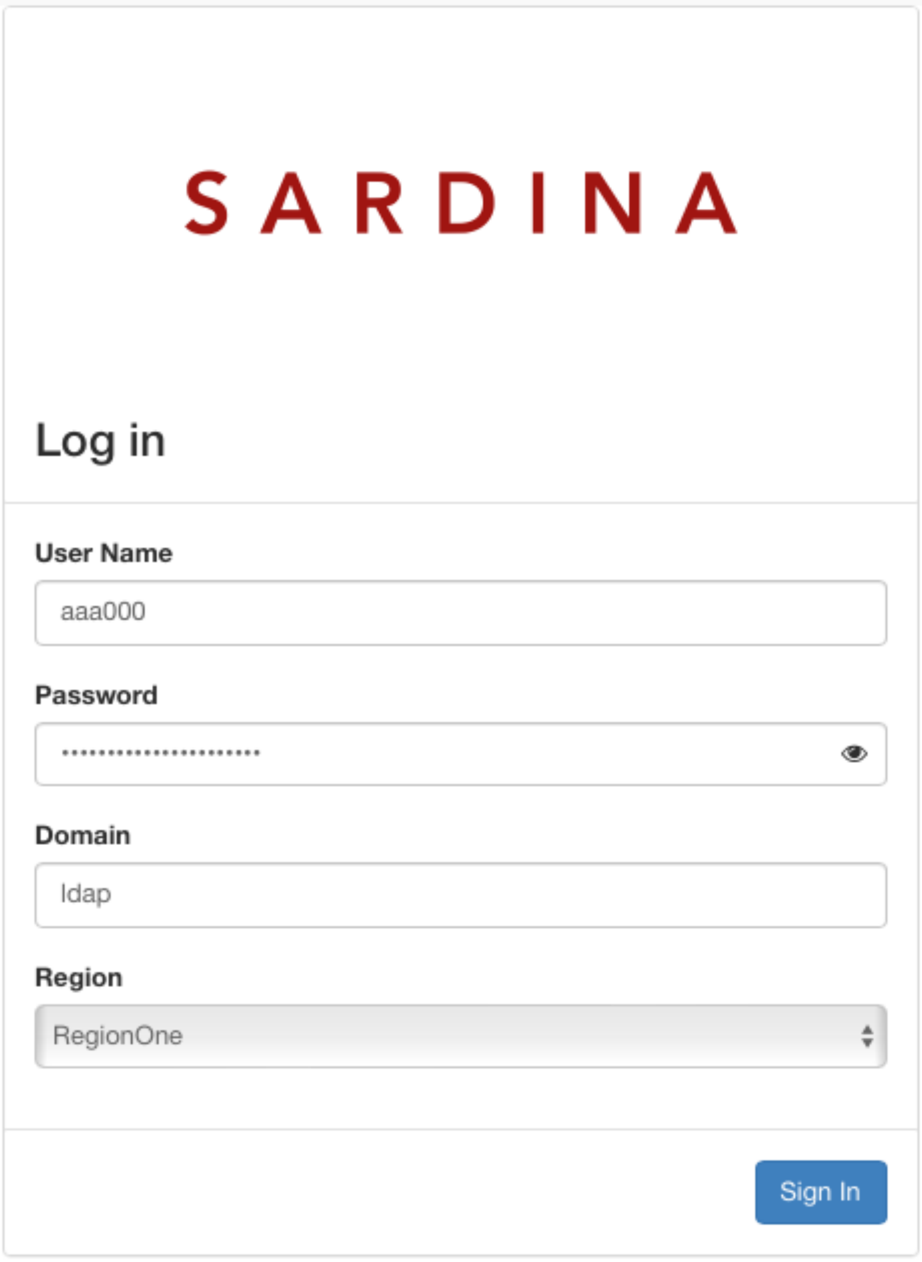
Ceph provides large amounts of slow storage for archival purposes, also known as *cold storage*. *Cold storage* is designed to store large amounts of data as a slow archive, with larger quotas, as opposed to *hot* or *warm* storage, which are for storing data to be used for immediate computation. For offloading data you want to keep, but do not need for present computations, Ceph is available.

12.2 Accessing Ceph

Ceph access uses the *S3 protocol*, which means you must have both *S3 credentials* and an *S3 client*.

12.2.1 S3 credentials: Web interface

To use the S3 protocol, you must first generate credentials through OpenStack.


SARDINA

Log in

User Name: aaa000

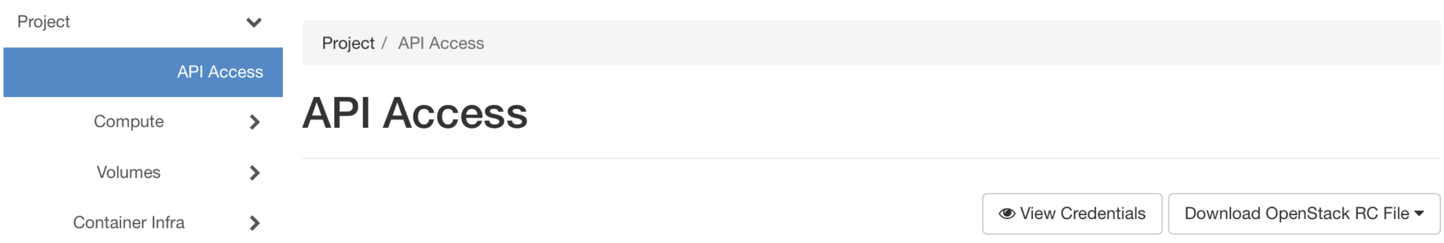
Password: ••••••••••••••••••••••

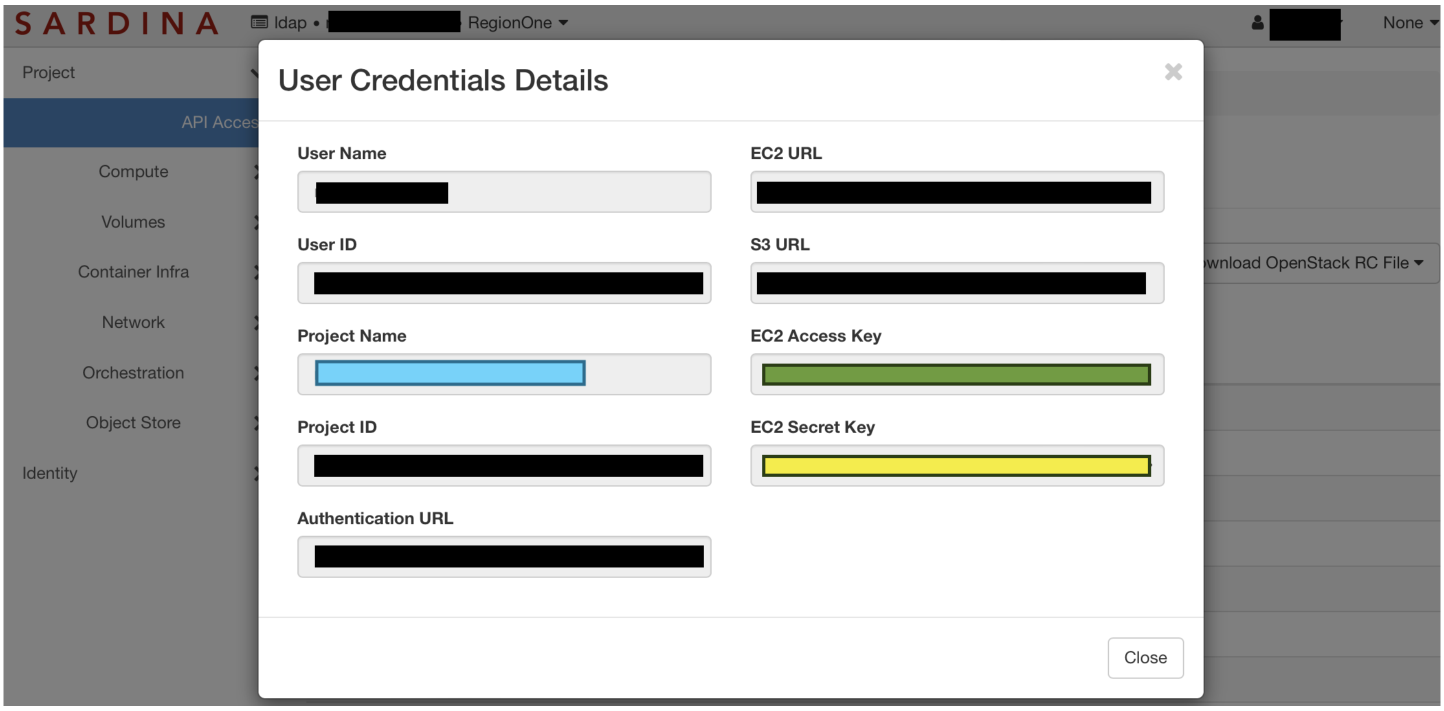
Domain: ldap

Region: RegionOne

Sign into OpenStack with your ML Cloud credentials.

Upon signing in, go to Project / API Access, as shown, and click the "View Credentials" button on the right hand side:





User Credentials Details

User Name	**EC2 URL**
[redacted]	[redacted]
User ID	**S3 URL**
[redacted]	[redacted]
Project Name	**EC2 Access Key**
🟦	🟩
Project ID	**EC2 Secret Key**
[redacted]	🟨
Authentication URL	
[redacted]	

Note down the colored fields: in this screenshot, the **Project Name** is in blue, the **EC2 Access Key** is in green, and the **EC2 Secret Key** is in yellow. These together are your S3 credentials for your personal Ceph access. **Do not share these credentials with anyone!**

12.2.2 S3 credentials: Command line

You may also create EC2 credentials with the `openstack` command-line interface as follows:

```
source ./openrc openstack ec2 credentials create +----------------------------------+----------------------------------+--------
```

Or, if you are member of multiple projects: `openstack ec2 credentials create --project Project_Name`

12.2.3 S3 client: Inside the ML Cloud

Inside the ML Cloud, you can use the `aws` or `s5cmd` clients. (`s5cmd` is potentially faster but has fewer features.)

To use the `aws` client, you must first create a credential file. On the login nodes, run the following set of commands, substituting your **EC2 access key** and **EC2 secret key** from the OpenStack dashboard you accessed before:

```
mkdir $HOME/.aws/ echo "[mlcloud] aws_access_key_id=EC2_ACCESS_KEY aws_secret_access_key=EC2_SECRET_KEY" >> $HOME/.aws/credentia
```

Within the cluster, the Ceph S3 addresses are: * *Galvani*: http://192.168.213.10:7480

These are known as *endpoints* in Ceph documentation.

12.3 S3 client: On your own computer

On your own computer, if you are running Linux, you can access Ceph using the command-line AWS-S3 client `aws` or `awscli`, depending on your operating system.

The credential setup is the same, but the Ceph S3 addresses are instead: * *Galvani*: https://mlcloud.uni-tuebingen.de:7443

For Windows users, if you've already installed WinSCP, its S3 connection functionality is described here, and, for both Windows and Mac users, a variety of other S3 clients are available (for example, Cyberduck).

12.4 Storing data on Ceph

Data on Ceph is stored in `buckets` - the S3 term for remote folders. The buckets for ML Cloud users correspond to projects - remember the **Project Name** mentioned above? The project name has two parts separated by a dash, e.g. *mladm0-mfa555* The first part - e.g. *mladm0* - is your *bucket name*, corresponding to the research project in question. Everyone with access to the same bucket can access the data in that bucket. The second part is your *user name* for that bucket.

You can read data from the bucket at `s3://BUCKET_NAME` and you can write data to the bucket at `s3://BUCKET_NAME/USER_NAME` - you can make subdirectories, as well - for example, `s3://BUCKET_NAME/USER_NAME/directory_name/`

Remember, all users in the same project can access files in the same bucket. Never ever ever store credentials (including SSH private keys) in an S3 bucket!

The first time ever that you access your group's bucket, you must initialize it with the following command from inside the cluster (remember to set up your credential on the login node first) - this command also shows the contents of your upload folder:

- `aws --endpoint-url http://192.168.213.10:7480 --profile mlcloud s3 ls s3://BUCKET_NAME/USER_NAME/`

To move local data (e.g. `~/localdata.file`) to your S3 Ceph bucket in Galvani, you'd run the following within the cluster:

- `aws --endpoint-url http://192.168.213.10:7480 --profile mlcloud s3 cp ~/localdata.file s3://BUCKET_NAME/USER_NAME/my_directo`

And this from your own computer (using `aws`):

- `aws --endpoint-url https://mlcloud.uni-tuebingen.de:7443 --profile mlcloud s3 cp ~/localdata.file s3://BUCKET_NAME/USER_NAME`

To move data (e.g. `ceph_data.file`) from your Ceph S3 bucket in to a compute node you're logged into:

- `aws --endpoint-url http://192.168.213.10:7480 --profile mlcloud s3 cp s3://BUCKET_NAME/USER_NAME/my_directory/ceph_data.file`

And this from your own computer (using `aws`):

- `aws --endpoint-url https://mlcloud.uni-tuebingen.de:7443 --profile mlcloud s3 cp s3://BUCKET_NAME/USER_NAME/my_directory/cep`

It is technically possible to use `s3fs` to mount an S3 bucket directly, but we do not recommend this, as it's 5x slower on typical data and has problems with large files and POSIX operations generally.

12.5 Publishing data on Ceph for the world to see

It is possible to publish an S3 bucket for the whole world to access and see. To do this, please contact ML Cloud support.

13. GALVANI SLURM

Galvani can be accessed through a dedicated set of login nodes:

`134.2.168.43`

used to to write and compile applications as well as to perform pre- and post-processing of simulation data.

13.1 Available Partitions on Galvani

Queues and limits are subject to change.

Queue Name	**Limits**	**Resources per node**	**Cost**	**Description**

`cpu-galvani`	3d	32 CPUs
		1228 GB RAM
			CPU=0.218,
			Mem=0.0126G	

`a100-galvani`	3d	32 cores
		1024 GB RAM
		8 A100
			CPU=1.406,
			Mem=0.1034G,
			gres/gpu=11.25
				GPU nodes with 8x A100

`2080-galvani`	3d	72 CPUs
		384 GB RAM
		8 2080Ti
			CPU=0.278,
			Mem=0.0522G,
			gres/gpu=2.5
				GPU nodes with 8x 2080Ti

`2080-preemptable-galvani`	3d	-	same as gpu-2080ti	CPU=0.0695,
				Mem=0.01305G,
				gres/gpu=0.625

Terms
* **MaxJobsPU:** Maximum number of jobs each user is allowed to run at one time. * **MaxJobsPA:** Maximum number of jobs each account/group is allowed to run at one time. * **MaxSubmitPU:** Maximum number of jobs pending or running state at any time per user. * **MaxSubmitPA:** Maximum number of jobs pending or running state at any time per account/group.

13.2 Cuda Kernels compilation

There are several development tools installed on Galvani, to see them:

`scl --list`

which should output:

`gcc-toolset-10 gcc-toolset-11 gcc-toolset-9`

To source and enable them:

```
source scl_source enable <devtoolset-x>
```

13.3 CUDA on Compute Nodes

You can find different cuda versions on compute node under:

```
ls -1 /usr/local/cuda* -d
```

```
/usr/local/cuda-11 /usr/local/cuda-11.7 /usr/local/cuda-11.8 /usr/local/cuda-12 /usr/local/cuda-12.1
```

13.4 Submitting Jobs to Galvani Slurm

To understand how to submit jobs, please refer to the Slurm explanation.

14. FERRANTI SYSTEM ARCHITECTURE

This page describes the infrastructure that makes up the Ferranti Cluster.

14.1 Login Nodes

There will be two login nodes for Ferranti:

`TBD`

with the following configuration:

Feature	**Specifications**
CPUs:	2x Intel Xeon Gold 6430 CPUs (32 Cores, 2.1 GHz)
RAM:	1024GB DDR5-4800
HCA:	2x NVidia Mellanox ConnectX-7 NDR200 Adapter

14.2 Ferranti: Compute Infrastructure

The **5** GPU compute nodes with the following specifications:

Feature	**Specifications**
Total Nodes:	5
Accelerators:	8 Nvidia H100 in SXM5 / node
FP32 Cores:	16896 / card
FP64 Cores:	8448 / card
NVIDIA Tensor Cores:	528 / card
GPU Memory:	80GB HBM3 / card (bandwidth: 3.35TB/s)
CPUs:	2x Intel Xeon Platinum 8468 CPUs, 48 cores, 2.1 GHz
RAM:	2048GB DDR5-4800
HCA:	6x NVidia Mellanox ConnectX-7 NDR400 Adapter
Local Storage:	96TB NVMe in RAID0
Theoretical Peak Performance:	--

14.3 Ferranti: Interconnect

Ferranti has Fat Tree interconnect topology with the following composition:

Feature	**Specifications**
Number of core switches:	4
Number of edge switches:	6
Interconnect topology and type:	NDR InfiniBand Fat Tree, non-blocking
Blocking factor:	1:1
Switch type:	Nvidia NDR switch with bandwidth of 400 Gb/s per port.

15. FERRANTI STORAGE

To be published on production time.

16. FERRANTI SLURM

To be published on production time.

17. ACCOUNTING

This page describes what a fairshare is, how it is used within the ML Cloud to calculate priority and what some of the terms actually entail.

17.1 Fairsharing and Job Accounting

Fairshare allows past resource utilization information to be taken into account into job feasibility and priority decisions to ensure a fair allocation of the computational resources between the all ML Cloud users. Fairshare allows those users who have not fully used their resource grant to get higher priority for their jobs on the cluster, while making sure that those groups that have used more than their resource grant do not overuse the cluster.

There are several existing fairshare algorithms, but at the ML Cloud we use Fair Tree, which is done through a a rooted plane tree. Some observations include: 1. All users from a higher priority account receive a higher fair share factor than all users from a lower priority account. 2. New jobs are immediately assigned a priority.

17.2 Definitions

- **Shares:** This is an integer value that symbolizes the portion of the computing resource that has been promised to the account or user.
- **Usage:** an integer value that symbolizes how much the account or user has consumed from the computing resources.
- **Usage Unit:** Integer value that is the result of: `TRES-sec` - `TRES` - resources that the job requested, and seconds is the time measured in seconds that the job used from start to finish, not considering time waiting for execution, we can call it real usage.
- **Priority:** : Integer value that ranges from 0 to 4294967295. The larger the value, the sooner the job will be scheduled for execution.
- **Fairshare:** Floating point number between 0.0 and 1.0 that reflects the shares of a computing resource that a user has been allocated and the amount of computing resources the user's jobs have consumed. For Fairshare calculation, the terms `RawShares` and `Shares` are equivalent, `RawUsage` and `Usage` are also equivalent.

To understand and determine how much a job will cost your fairshare account, please see Job cost

17.3 Your Fairshare Score Meaning

- **1.0:** Unused. The account has not run any jobs recently.
- **1.0 > f > 0.5:** Underutilization. The Group/User is underutilizing their granted Share. If for instance `f=0.75` the underutilization of share of resources is `1:2`.
- **0.5:** Average utilization. On average a user/group is using exactly as much as their granted Share.
- **0.5 > f > 0:** Over-utilization. The Group has overused their granted Share. For instance, `f=0.25` shows that the group has overutilized their Share of the resources `2:1`.
- **0:** No share left. The Group has vastly overused their granted Share. If there is no contention for resources, the jobs will still start.

Of course, the usage of the ML Cloud varies and the scheduler does not prevent groups/users from using more than their granted Share. The schedule will want to fill idle cycles, so it will run whatever jobs it has submitted. In this case, we can consider the group/user essentially "borrows" compute resource time in the future to be used now. This naturally drives down the user/group fairshare score, but still allows jobs to start. At some point another group with a higher fairshare score will start submitting jobs and those group jobs will have a higher priority because they have not used their granted Share.

Note

If there are two members of a given group, and if one of those users has run many jobs under that group, the job priority of a job submitted by the user who has not run any jobs will be negatively affected. This ensures that the combined usage charged to a group matches the portion of the cluster that is allocated to that group.

17.4 Shares

On the ML Cloud each user is associated by default to a group reflecting its direct supervisor within the institution. You may have other account associations, but only through the main one a user is granted shares. These Shares determine how much of the cluster that group has been granted. Users when running a job are "charged" for their runs against the group they belong to.

17.5 Priority computation

The way the priority is computed for a job depends on another parameter which is called PriorityType. In the ML Cluster the priority type is `multifactor`. The priority then depends on five elements: * Job age: how long the job has been waiting in the queue; * User fairshare: a measure of past usage of the cluster by the user; * Job size: the number of CPUs a job requests; * Partition: the partition to which a job is submitted, specified with the `--partition` submission parameter; * QOS: a quality of service associated with the job, specified with the `--qos` submission parameter.

Note that the job age parameter is bounded so that priority stops increasing when the bound is attained. The job size parameter can be configured to favor small or large jobs.

18. AVAILABLE DATASETS

Note

If you would like an additional dataset installed for general use please use the following form or contact us though the ticketing system.

Some commonly used datasets have been deployed already:

18.1 Available Datasets in *Galvani*

Cluster	Dataset Name	location
Galvani	`ImageNet-C`	`/scratch_local/datasets/ImageNet-C`
Galvani	`Imagenet2012`	`/scratch_local/datasets/ImageNet2012`
Galvani	`Imagenet-r`	`/scratch_local/datasets/imagenet-r`
Galvani	`ImageNet2012_val.tar`	`/mnt/qb/datasets/ImageNet2012_val.tar`
Galvani	`ImageNet-ffcv`	`/mnt/qb/datasets/ImageNet-ffcv`
Galvani	`CLEVR_v1.0`	`/mnt/qb/datasets/CLEVR_v1.0`
Galvani	`cl_ssl_ica`	`/mnt/qb/datasets/cl_ssl_ica`
Galvani	`coco`	`/mnt/qb/datasets/coco`
Galvani	`Falcor3D_down128`	`/mnt/qb/datasets/Falcor3D_down128`
Galvani	`ffcv_imagenet_data`	`/mnt/qb/datasets/ffcv_imagenet_data`
Galvani	`imagenet-styletransfer`	`/mnt/qb/datasets/imagenet-styletransfer`
Galvani	`kitti`	`/mnt/qb/datasets/kitti`
Galvani	`laion400m`	`/mnt/qb/datasets/laion400m`
Galvani	`ModelNet40`	`/mnt/qb/datasets/ModelNet40`
Galvani	`NMR_Dataset`	`/mnt/qb/datasets/NMR_Dataset`
Galvani	`stl10_binary`	`/mnt/qb/datasets/stl10_binary`
Galvani	`WeatherBench`	`/mnt/qb/datasets/WeatherBench`
Galvani	`yfcc100m`	`/mnt/qb/datasets/yfcc100m`
Galvani	`yfcc15m`	`/mnt/qb/datasets/yfcc15m`

18.2 Datasets on *Galvani* Compute Nodes

We have also deployed some commonly used datasets locally on compute nodes on select partitions for faster I/O in your jobs. Here is a list of currently available datasets:

Dataset	location	partition
Imagenet-c	`/scratch_local/datasets/ImageNet-C`	`2080-galvani` and `a100-galvani`
Imagenet	`/scratch_local/datasets/ImageNet2012`	`2080-galvani` and `a100-galvani`
Imagenet-r	`/scratch_local/datasets/imagenet-r`	`2080-galvani` and `a100-galvani`

Note

On request we can manually deploy datasets on a subset of nodes, which are then selectable with SLURM features/constraints. To request this, please contact us.

19. CONTAINERS

To ensure your results are reproducible and verifiable, use containers to package your research projects. Containers include an entire operating system, GPU libraries, etc., so this page discusses what a container is and how to work with our Singularity container system.

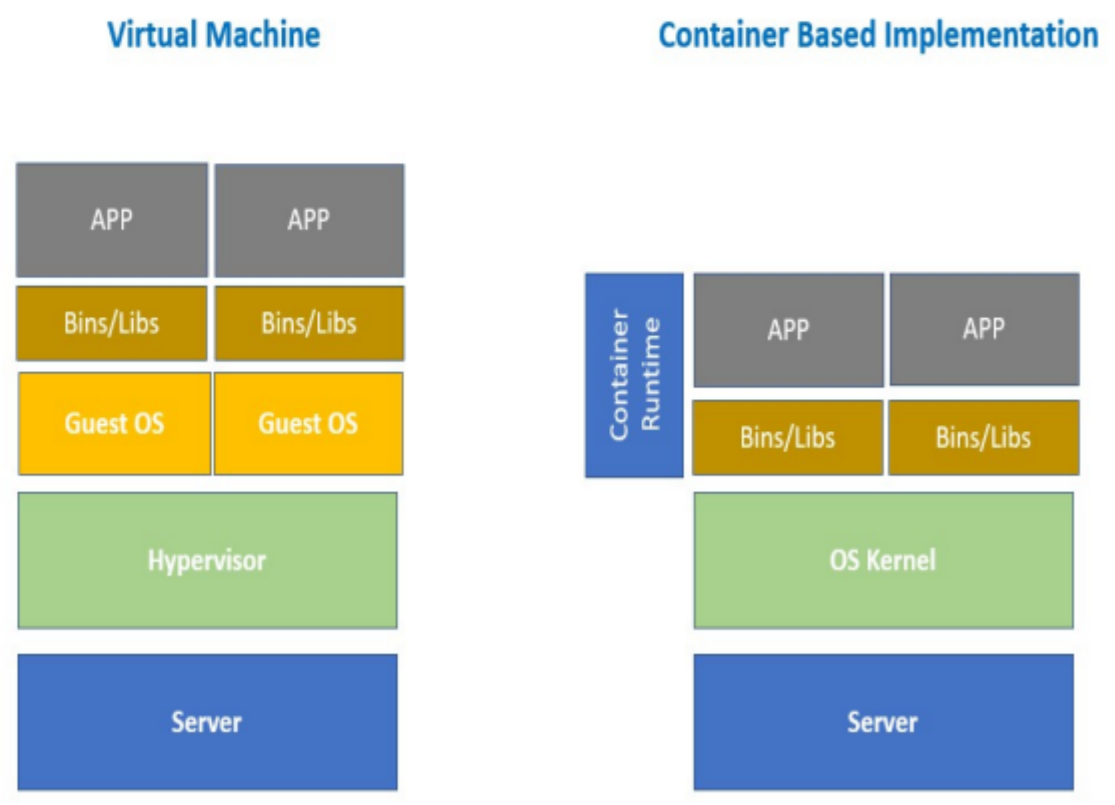
19.1 Reasons to Use Containers

Software has grown in complexity over the years making it difficult at times to install and run the software. Containers address this problem by storing the software and all of its dependencies (including a minimal operating system) in a single, large image so that there is nothing to install and when it comes time to run the software everything "just works". This makes the software both shareable and portable while the output becomes reproducible.

- A Singularity image bundles an application together with its software dependencies, data, scripts, documentation, license and a minimal operating system. Software in this form ensures **reproducible** results.
- Singularity images are stored as a single file which makes them easily shareable.
- A Singularity image can run on any system that has the same architecture (e.g., x86-64) and binary file format for which the image was made. This provides portability.
- Bring Your Own Software (BYOS). That is, you don't have to ask the system adminstrators if they are willing to install something for you. You can install whatever you want inside the image and then run it. This is because there is no way to escalate priviledges. That is, the user outside the container is the same user inside so there are no additional security concerns with containers.

19.2 Differences between Virtual Machines and Containers

Container	**VM**
OS process level isolation	OS level isolation with virtualized hardware
Can run 1,000s on a single machine	Can run “dozens” on a single machine.
Leverages kernel features (requirements on kernel version)	Leverages hypervisors (requirements on hardware)
Start up time ~100s of ms	Start up time ~minutes



19.3 Singularity

Docker images are not secure because they provide a means to gain root access to the system they are running on. This is not a problem because you can use Singularity which is an alternative to Docker that is both secure and designed for high-performance computing. Singularity is compatible with all Docker images and it can be used with GPUs and MPI applications, as well as Infiniband networks.

Singularity is already installed globally on all our systems, and should be immediately available on your command line (no `module load` necessary):

```
singularity --version
```

If you want to use Docker and other format containers they need to be converted to singularity format for use on the cluster.

19.4 GPU support for containers

Commands that run, or otherwise execute (singularity) containers (`shell`, `exec`) can take an `--nv` option, which will setup the container's environment to use an NVIDIA GPU and the basic CUDA libraries to run a CUDA enabled application. The `--nv` flag will:

- Ensure that the `/dev/nvidiaX` device entries are available inside the container, so that the GPU cards in the host are accessible.
- Locate and bind the basic CUDA libraries from the host into the container, so that they are available to the container, and match the kernel GPU driver on the host.
- Set the `LD_LIBRARY_PATH` inside the container so that the bound-in version of the CUDA libraries are used by applications run inside the container.

19.5 Converting docker to singularity images

There are (at least) 2 options for converting docker images to singularity dockers.

19.6 Use Singularity

If you want to convert a docker image called `ubuntu` with the tag `latest` to a singularity file called `IMAGE_NAME.sif`, just run this command:

```
singularity build IMAGE_NAME.sif docker://ubuntu:latest
```

You can execute this command on the login nodes, as well as on any of the compute nodes of the SLURM cluster. Also, you can find more information here.

19.7 Use docker2singularity

If you have access to a computer with docker installed (and running) you can also use the `singularity2docker` tool for converting your images. To convert the `ubuntu` image to singularity (as above) use:

```
docker run -v /var/run/docker.sock:/var/run/docker.sock \ -v /tmp/test:/output \ --privileged -t --rm \ quay.io/singularity/dock
```

This will create a `*.simg` file you can then transfer to a ML-Cloud server. By itself this is more complicated then the first method, but it can become necessary if you want to make adjustements to a container that are difficult to reproduce in a singularity recipe.

You can find more about `singularity2docker` usage on github.

19.8 Singularity on ML Cloud Example

Set the cache and tmp directories for singularity

This is required to build the container, as otherwise storage can be insufficient and result in a build failure.

```
export SINGULARITY_CACHEDIR=/scratch_local/group-name/your-username export SINGULARITY_TMPDIR=/scratch_local/group-name/your-us
```

Mounting behavior

Singularity by default mounts these 5 folders from the host filesystem into the container filesystem:

```
/home/$USER /tmp /dev /sys /proc
```

so if your script or data is in one of these folders then you don't need manual mounting and can simply specify the path to the files to singularity commands.

Build the Image

To build an image without root privileges, use `--fakeroot`, e.g. as in

```
singularity build --fakeroot IMAGE_NAME.sif DEFINITION_NAME.def ``` Where `DEFINITION_NAME.def` is the definition file and `IMAG
```

so if your script or data is in one of these folders then you don't need manual mounting and can simply specify the path to the files to singularity commands.

19.9 An example singularity definition file with dropbear SSH

The following example has been provided by the Bethge/Brendel on-boarding notion documentation.

```
Bootstrap: docker From: nvidia/cuda:11.3.0-cudnn8-runtime-ubuntu20.04 %environment # overwrite Singularity prompt with something
```

19.10 Install matlab into docker/singularity images

```
 # --- MATLAB --- ENV MATLAB_RELEASE=r2022a # required packages depend a bit on which matlab toolboxes are required RUN apt-get
```

This example was kindly provided by ***Matthias Kümmerer***.

19.11 References:

- Singularity Library
- Nvidia GPU-Accelerated Containers (NGC)
- singularity2docker

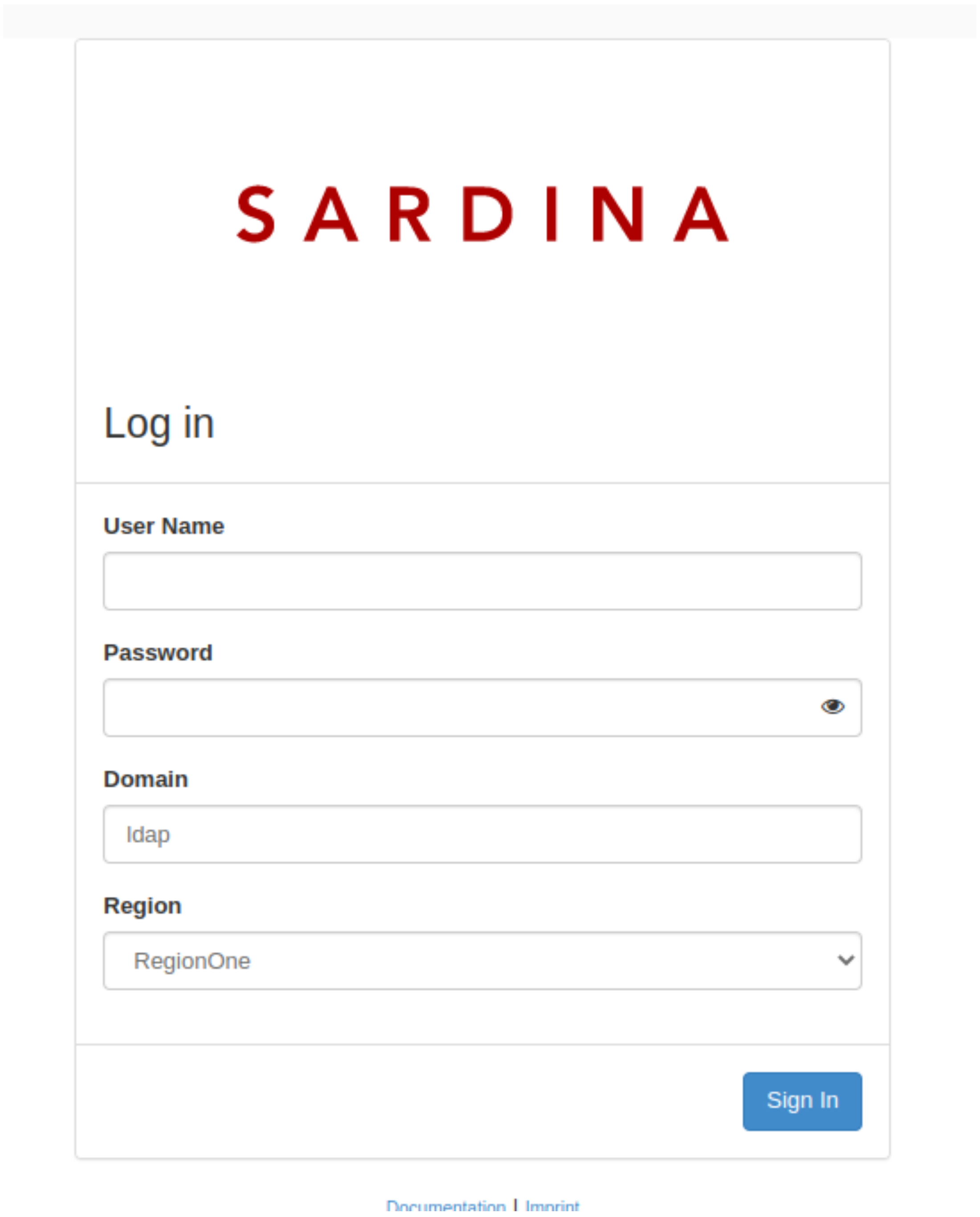
20. OPENSTACK

The ML Cloud has an Openstack implementation, with the **yoga** release. OpenStack is an open source software that provides cloud infrastructure for virtual machines, bare metal, and containers. This page describes how to access OpenStack, allocate resources, and available interactions with a running VM.

OpenStack is an open source cloud computing platform that is used by organizations to manage and control large scale deployments of virtual machines, such as in a cloud computing or virtual private server environment. OpenStack is a popular choice for organizations because it is scalable, reliable, and provides a high degree of control over the underlying infrastructure. OpenStack can also be used to manage storage and networking resources in a cloud environment.

20.1 Login

Go to https://mlcloud.uni-tuebingen.de where in the login screen you have to enter your username and password from your ML Cloud account, then choose **Domain:** ldap, and the Region which you would like to work on:


SARDINA

Log in

User Name

Password

Domain

ldap

Region

RegionOne

OpenStack Client can be installed from PyPI using pip:

```
pip install python-openstackclient
```

There are a few variants on getting help. A list of global options and supported commands is shown with `--help`:

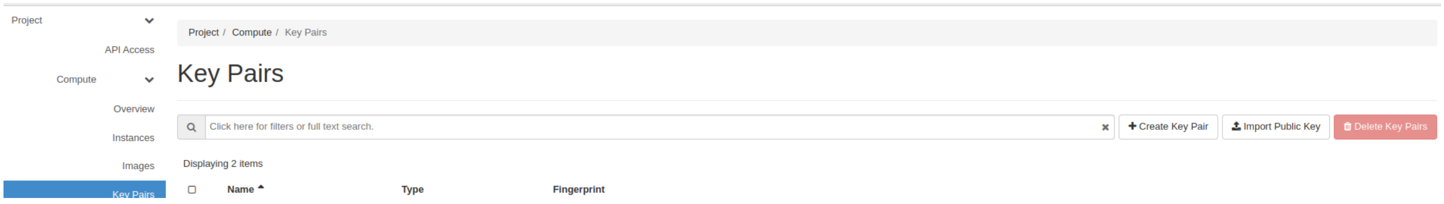
```
openstack --help
```

20.2 Change to English

Once you are logged in go to your personal settings (top-right corner: `user01 > Settings`) and switch the language to English. This will make it easier to follow the majority of the avaible documentation and to communicate issues to us.

20.3 Set up SSH keys

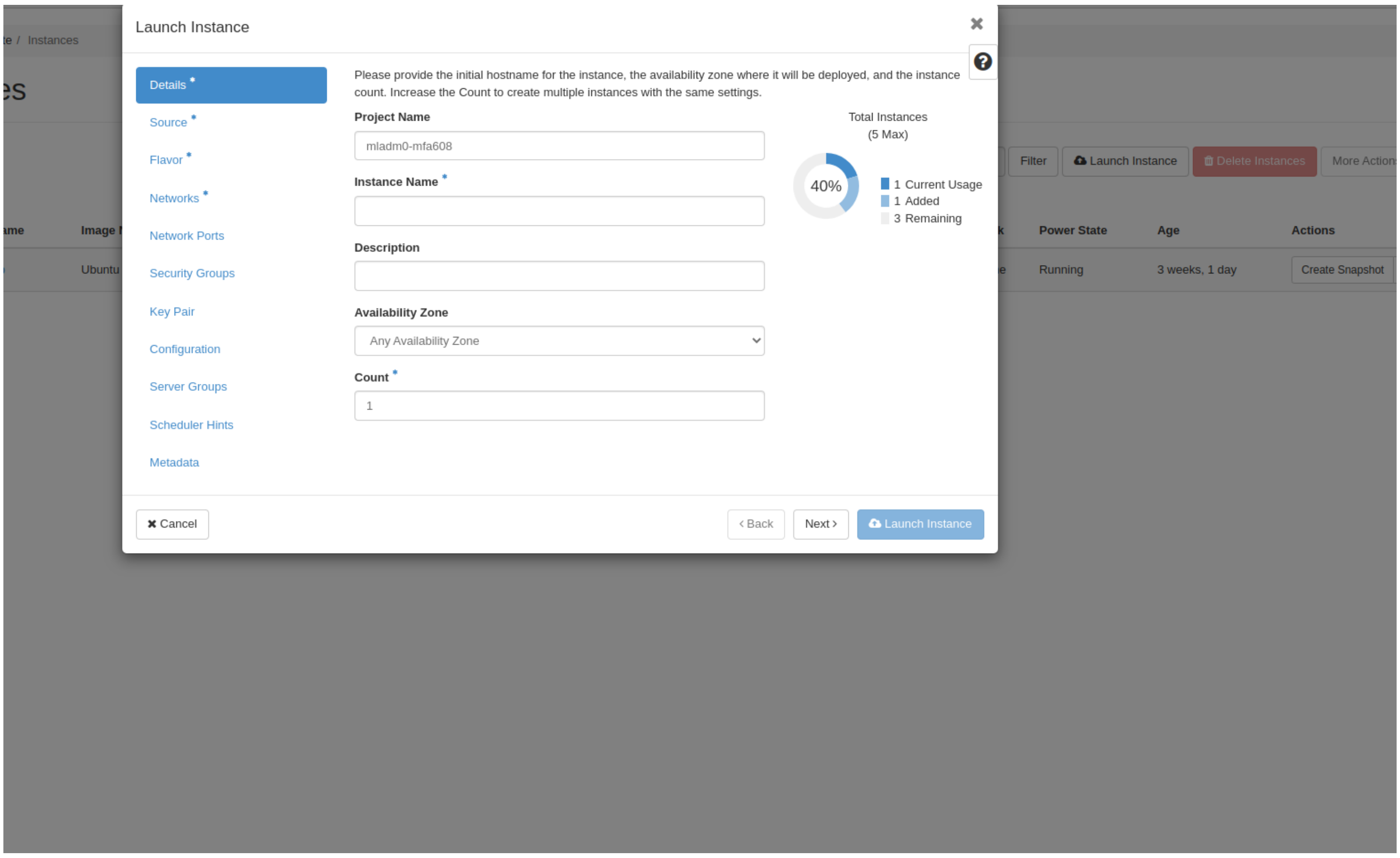
Once you are logged in, go to `Compute-> Key Pairs` and either create a key pair or upload your public key. The key uploaded must be in the OpenSSH format.



20.4 Start a VM/Instance

Openstack refers to VMs as Instances. Be sure to be in the correct project and Region (top-left corner)

- Go to `Project > Compute > Instances` and use `Launch Instance`



Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Project Name: mladm0-mfa608

Instance Name *:

Description:

Availability Zone: Any Availability Zone

Count *: 1

Total Instances (5 Max)

40%

1	Current Usage
1	Added
3	Remaining

You have to fill out several details such:

- Details:
  - Choose instance name
  - Availability Zone: `nova`
- Source:
  - Image or Instance Snapshot: the first allows you to select from a pre-configured cloud-images, whereas `Instance Snapshot` allows you to launch a clone of a previous VM.
- Flavor
  - specify your resource requirements. Total Disk refers only to the size of the root partition. Additional storage for data can and should be added later.
- Networks
  - Select `qbstorage.4005` in Region 1 (and `storage.3905` in Region 2) if you later want to use Quobyte volumes (recommended for compute VMs) and `local_network` otherwise.
  - We advice against adding more than one network.
- Security Groups:
  - Choose one or more security groups which do not have overlapping rules.
  - This setting can be adjusted afterwards and multiple times.
- Key Pair
  - You can choose one of your uploaded public ssh keys.
  - This setting can not be modified later: If you choose the wrong key or none the VM cannot be accessed.
- Network Ports/Configuration/Server Groups/Scheduler Hints/Metadata are optional and potentially destructive: Leave at their default values.

Confirm with the Launch Instance button.

20.5 Floating IPs

- Allocate a floating IP by going to `Project > Network > Floating IPs` and using the button `Allocate IP` to `Project`.
  - Use the Pool osprovider to get a public IP. The `DNS-*` fields currently have no use and should be left empty.
- Associate a floating IP to a VM in the dropdown of this VM in `Project > Compute > Instances`. The VM itself does not know anything about the floating IP itself makeing it independent of the IPs value

20.6 Access a running VM

To access your VMs you will need: * a floating IP or access via a jump-host you might have previously configured * appropriate security groups (the default security group is sufficient for ssh access), * the username of standard cloud-images is usually the lower-case distribution name: e.g. `ubuntu` or `centos`.

Assuming you meet all requirements, used an Ubuntu image and a ssh key named `id-user01_rsa`, and associated the floating IP `134.2.168.10`, you then should be able to log in via:

```
ssh -i /path/to/id-user01_rsa ubuntu@134.2.169.10
```

20.7 Security Groups

Security Groups are firewalls simplified to white-listing of protocols and ports. They can be edited under `Project > Network > Security Groups`. VMs can use multiple security groups.

While each project has its own `default` security group, which is managed by the admin team and opens the ssh port, you can create and manage other security groups within your poject. Be aware that if rules of different security groups overlap: e.g. if ALLOW IPv4 to `0.0.0.0/0` (which allows all outgoing connections) is part of more than one used security groups it can cause problems. Therefore the ML Cloud team recommends you: * use one base security group like default per VM * use additional extending security groups that add functionality: e.g. `+http`, `https` which would only whitelist tcp port `80` and `443`. * when you create a new extending security groups Openstack inserts `ALLOW IPv4 to 0.0.0.0/0` and `ALLOW IPv6 to ::/0`, which you should **delete**.

20.8 Release ressources

Unused and running, or stopped VMs allocate ressources that other people could use. Please shelve VMs if you do not need them for more than 3 days or delete them if you do not need them at all.

- **Stopping a VM means:** Force a shutdown of the VM and stop all `qemu-kvm` processes of that VM. CPU Cores, GPUs, RAM and storage devices are still reserved for the VM and allow to restart it immidiately, but also prevents other people from using these resources. A stopped VM uses less energy and frees some compute time for other VMs on the hypervisor (if there are any).
- **Shelving a VM means:** Stop the VM, snapshot it (transfer its `'/'` filesystem to the storage backend) and delete it on the hypervisor. The VM remains in the Openstack database. This frees all ressources your VM uses on the hypervisor, which can then be used by other users/VMs. Can be reversed by unshelving the VM if the required amount ressources is available on any of the hypervisors. Unshelving takes a little longer than restarting a stopped VM, because the snapshots needs to be transferred back to a hypervisor. The `'/'` filesystem you used is still the same and in the state you left it. Note that all data on ephemeral diskspace (typically /dev/vdb) is lost by this procedure.
- **Deleting a VM means:** It will be totally removed from the system (hypervisor + compute database). If you want to have a copy from your VMs `'/'` filesystem please make a snapshot of the VM before deleting it.

20.9 Resizing Instance

TBD

20.10 Moving VMs to a Different Network

TBD

21. PROFILING AND DEBUGGING

This page discusses profiling tools (to ensure your code is running optimally) and debugging tools (to find errors in your code).

21.1 Python Profiling with line profiler

The starting point for profiling a Python code that uses a GPU (this includes `PyTorch` and `TensorFlow`) is to use `line_profiler`.

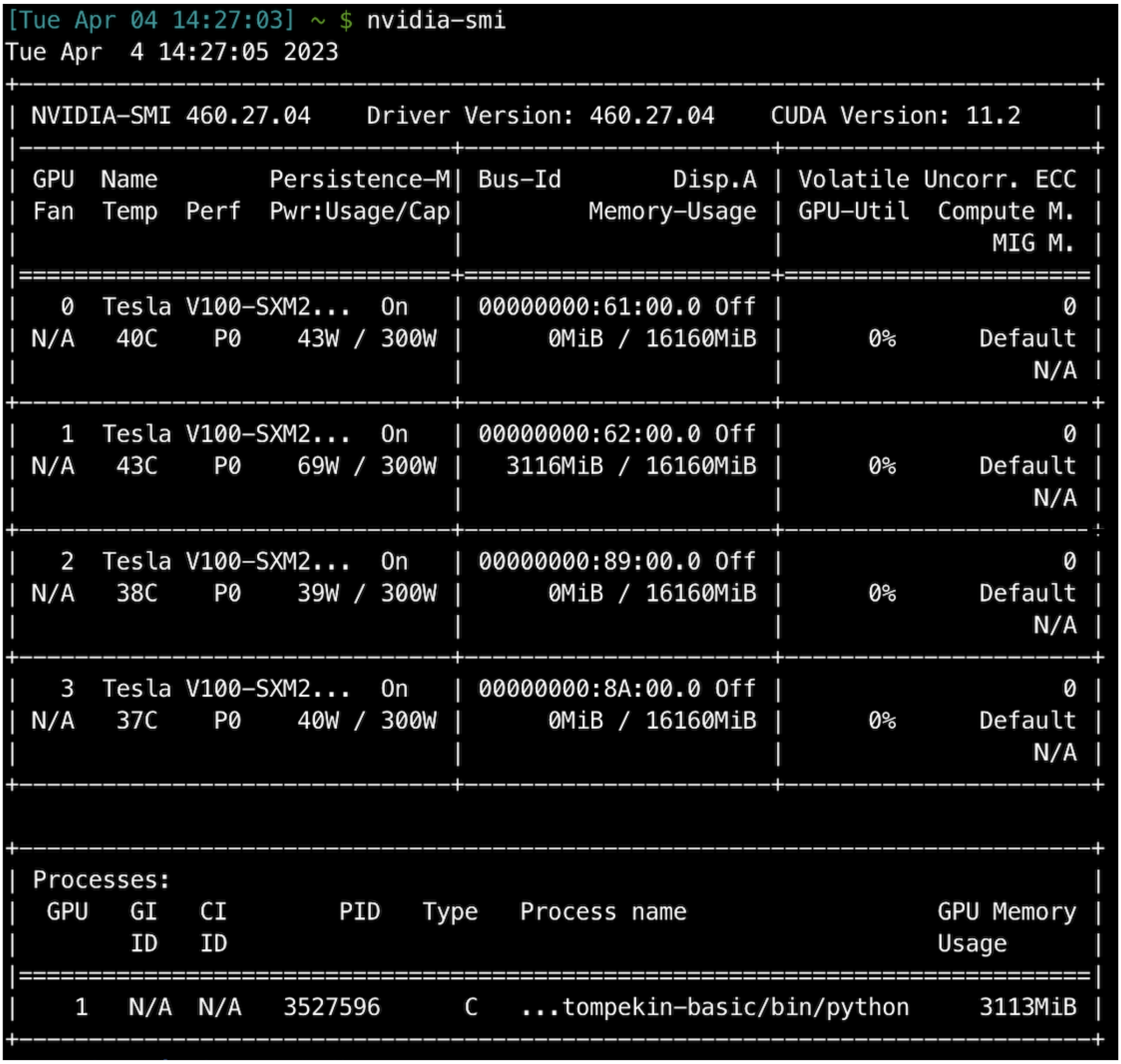
Install line profiler

```
pip install line-profiler
```

21.2 Nvidia diagnostic commands

The most used command to check Nvidia GPU status is `nvidia-smi`, which provides basic information about GPU usage on a node. This shows what GPU resources the job is currently using, helping a user to optimize what resources to request.

Running `nvidia-smi` with no arguments will provide a quick overview of current GPU status on the node the command is executed on, and only showing the GPUs allocated via SLURM. The response may look like this:



```
[Tue Apr 04 14:27:03] ~ $ nvidia-smi
Tue Apr  4 14:27:05 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.27.04    Driver Version: 460.27.04    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:61:00.0 Off |                    0 |
| N/A   40C    P0    43W / 300W |      0MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000000:62:00.0 Off |                    0 |
| N/A   43C    P0    69W / 300W |   3116MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-SXM2...  On   | 00000000:89:00.0 Off |                    0 |
| N/A   38C    P0    39W / 300W |      0MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100-SXM2...  On   | 00000000:8A:00.0 Off |                    0 |
| N/A   37C    P0    40W / 300W |      0MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+


+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    1   N/A  N/A   3527596      C   ...tompekin-basic/bin/python     3113MiB |
+-----------------------------------------------------------------------------+
```

From this output, we know that this process on this machine has access to four Nvidia V100 GPUs, numbered 0 to 3, and there is a process running on GPU 1. Running `nvidia-smi -i 0` would show the same view, but only for GPU 0.

- In this case, the job is only currently using 1 of the 4 requested GPUs, although it may have used more earlier. It is wise to only request resources you need, and `nvidia-smi` can help understand what resources your job is currently using.

More advanced information is returned using `nvidia-smi -q`, which returns a list of properties line-by-line, suitable for `grep`, etc., and `nvidia-smi -q -i 0` would return the same list of properties for GPU 0 only.

The use of the real-time monitoring options for `nvidia-smi` is possible, but discouraged. Should you need to use them, remember to use relatively long polling times (e.g. 1-3 minutes) to avoid log spam.

`nvidia-smi -h` will yield a complete list of `nvidia-smi` options.

Remember that `nvidia-smi` only knows about GPUs connected to the CPU evaluating the command and allocated using SLURM.

21.3 Dlprof by Nvidia

If you are using PyTorch or TensorFlow on A100 GPUs then consider profiling your code with dlprof by NVIDIA. dlprof provides suggestions on how to improve performance.

21.4 Nsight by Nvidia

NVIDIA provides Nsight Systems for profiling GPU codes. It produces a timeline and can handle MPI but produces a different set of profiling data for each MPI process.

To look closely at the behavior of specific GPU kernels, NVIDIA provides Nsight Compute.

22. TUTORIALS

22.1 Launch collaborative Jupyter notebooks on the ML Cloud

If two people want to collaborate using a single Jupyter notebook - for example, if a PhD student and their advisor want to work on the same document - this quick tutorial describes how to use `Jupyterlab` to collaborate on a workbook running on a compute node.

First, the host's setup actions are described, then the collaborator's setup actions. Finally, the remote connection is described for both users. This is set up to describe *Galvani*.

Both host and collaborator need to have ML Cloud access.

22.1.1 Host Tutorial

Host prerequisites:

For this to work, you first need `jupyterlab` and a specific version of `jupyter-server` installed on your server Python environment. This tutorial assumes you are using `conda` and already have a `conda` environment.

- If you do not have a conda environment, run the following commands starting on the login node:
    - `srun --pty bash` # will open a command prompt on a compute node.
    - `conda create -p $WORK/conda_envs/jupyter-collab` # creates a new conda environment
    - `conda activate $WORK/conda_envs/jupyter-collab` # activate the conda environment
    - `conda install -c conda_forge jupyter` # Install Jupyter generally
    - `python3 -m pip install "jupyter_server>=2.0.0"` # install the required version of the `jupyter-server`.

You will likely need to install other packages as well for your specific application.

- If you do have a conda environment, then, starting on the login node,
    - `srun --pty bash` # will open a command prompt on a compute node.
    - Activate the conda environment and run `python3 -m pip install "jupyter_server>=2.0.0"` which will install the necessary version of the Jupyterlab server that supports collaboration.

Host actions:

- When logged in on a login node, run `srun --pty bash`
- Now, run `hostname`. It will return something like `galvani-cn105.sdn` which you should note down.
- Now, run `jupyter lab --collaborative`. The `--collaborative` string enables collaborating on the same document. You will see the standard startup sequence ending with some lines like

Or copy and paste one of these URLs:

`http://localhost:8888/lab?token=76ebcaaf860c9426a1a34d373ad11ecf165371074dc69030`

`http://127.0.0.1:8888/lab?token=76ebcaaf860c9426a1a34d373ad11ecf165371074dc69030`

- The host now needs to note down three things:
    - The *compute node* currently being used for computation (the result of the `hostname` command, but only the string before the `.`)
    - The *compute node port*, which is the number after "`localhost`"
    - The *Jupyter server access URL*, which is the entire line including `localhost`
- In the example above,
    - The *compute node* is `galvani-cn105`
    - The *compute node port* is `8888`
    - The *Jupyter service access URL* is `http://localhost:8888/lab?token=76ebcaaf860c9426a1a34d373ad11ecf165371074dc69030`

Keep this terminal window open as long as you're working on the Jupyter notebook.

Give these three things to your collaborator. They can then start on the **Collaborator Tutorial** below.

Now, open a new terminal window and proceed to the **Remote Jupyter Connection** tutorial below.

Collaborator Tutorial

Once the collaborator has been given the *compute node*, *compute node port*, and *Jupyter service access URL*, they need to do the following:

- Log into the same cluster as the host.
- On the login node, run `srun --nodelist=$NODE --pty bash` where `$NODE` is the *compute node* given above.

Keep this terminal window open as long as you're working on the Jupyter notebook.

Remote Jupyter Connection

For both the host and the collaborator, this section describes the protocol for connecting to a Jupyter notebook running on a compute node.

- Obtain this session's *compute node*, *compute node port*, and *Jupyter service access URL*. They will change each time you go through the above steps
- Open a new terminal window.
- On your computer, run the following command, and note down the number as the *bounce port*, which we will call the `$B_PORT`.
    - `echo $(( 8800 + RANDOM%100 ))`
- On your computer, run the following command, replacing `$YOURLOGIN`, `$NODE`, `$B_PORT`, and `$COMPUTE_PORT` with *your login name*, the *compute node*, the *bounce port* and the *compute node port*, respectively:
    - `ssh -AtL $B_PORT:localhost:$B_PORT $YOURLOGIN@LOGIN_NODE_IP "ssh -AtL $B_PORT:localhost:$COMPUTE_PORT $YOURLOGIN@$NODE b`

(Yes, we know that's a long command with many things to replace.) This long command will enable you to open a browser window to the running Jupyterlab server.

With that terminal window open (plus the one from the Host/Collaborator Tutorial steps above; you should have two terminal windows open now), you should be able to open the *Jupyter service access URL* in your browser and be able to collaborate! Have fun!

22.1.2 References

- Jupyter Documentation
- Jupyter Lab

22.2 Installing PyTorch with GPU support using Conda.

The easiest way to install PyTorch with GPU support is using Conda. This will yield a Python environment, located in your `$WORK` directory, with a GPU-enabled version of the PyTorch package.

First, you need to open a terminal connection to a node with a GPU. The following command will do this:

```
srun --partition=2080-galvani --gres=gpu:1 --pty bash
```

Now you are in a terminal window on a compute node. On the new terminal on the compute node, run the following commands. First, you are creating a new Conda environment with Python version 3.11. Second, you are activating that environment so that you can run commands within it. Third, you are installing the PyTorch package with CUDA/GPU support.

```
conda create -p $WORK/.conda/py-311-pytorch python=3.11 conda activate $WORK/.conda/py-311-pytorch conda install pytorch torchvi
```

If you need other Python packages in your PyTorch environment, you can install them with `python3 -m pip install $PACKAGE_NAME` or `conda install $PACKAGE_NAME.` To find which packages are available, you must search the PyPi website.

If you want to check that this worked, just after running the three commands above, run these two commands at the terminal.

```
python3 -c "import torch; print(torch.cuda.is_available())" # Should return True python3 -c "import torch; print(torch.rand(5, 3
```

22.2.1 Using PyTorch with GPU support

To use PyTorch with GPU support, run the following in your `sbatch` script:

```
conda activate $WORK/.conda/py-311-pytorch
```

This will activate the Conda environment for PyTorch (plus whatever other packages you have in that environment).

22.3 Installing TensorFlow with GPU support using Conda.

The easiest way to install TensorFlow with GPU support is using Conda. This will yield a Python environment, located in your `$WORK` directory, with a GPU-enabled version of the TensorFlow package.

First, you need to open a terminal connection to a node with a GPU. The following command will do this:

```
srun --partition=2080-galvani --gres=gpu:1 --pty bash
```

Now you are in a terminal window on a compute node. On the new terminal on the compute node, run the following commands. First, you are creating a new Conda environment with Python version 3.11 (the most recent one that supports TensorFlow with GPU) Second, you are activating that environment so that you can run commands within it. Third, you are installing the TensorFlow package with CUDA/GPU support.

```
conda create -p $WORK/.conda/py-311-tf python=3.11 conda activate $WORK/.conda/py-311-tf python3 -m pip install tensorflow[and-c
```

If you need other Python packages in your TensorFlow environment, you can install them with `python3 -m pip install $PACKAGE_NAME`. To find which packages are available, you must search the PyPi website.

If you want to check that this worked, just after running the three commands above, run these two commands at the terminal.

```
python3 -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))" # Should return some error messages about fa
```

22.3.1 Using TensorFlow with GPU support

To use TensorFlow with GPU support, run the following in your `sbatch` script:

```
conda activate $WORK/.conda/py-311-tf
```

This will activate the Conda environment for TensorFlow (plus whatever other packages you have in that environment).

23. GOOD CONDUCT ON THE ML CLOUD

Follow the guidelines and rules on this page when interacting with the ML Cloud in order to be respectful of your fellow users.

You share the ML Cloud with many of other users, and what you do on the system affects others. All users must follow a set of good practices which entail limiting activities that may impact the system for other users. Exercise good conduct to ensure that your activity does not adversely impact the system and the research community with whom you share it.

The ML Cloud staff is developing the following guidelines to good conduct on usage of the ML Cloud. Please familiarize yourself especially with the first two mandates. The next sections discuss best practices, and we provide job submission tips when constructing job scripts to help minimize wait times in the queues.

- Do Not Run Jobs on the Login Nodes
- Do Not Stress the File Systems
- Limit Input/Output (I/O) Activity
- File Transfer Guideliones
- Job Submission Tips

23.1 Do Not Run Jobs on the Login Nodes

ML Cloud's login nodes are shared among all users. Dozens of users may be logged on at one time accessing the file systems. Think of the login nodes as a prep area, where users may edit and manage files, compile code, perform file management, issue transfers, submit new and track existing batch jobs etc. The login nodes provide an interface to the "back-end" compute nodes.

The compute nodes are where actual computations occur and where research is done. Tens of jobs may be running on all compute nodes, with many more queued up to run, especially before deadline submissions. All batch jobs and executables, as well as development and debugging sessions, must be run on the compute nodes.

Running jobs on the login nodes is a sure way to impact performance for other users. Instead, run such jobs if necessary on the compute nodes via an interactive session or by submitting a batch job.

23.2 Dos & Don'ts on the Login Nodes

- **Do not run research applications on the login nodes;**. If you need interactive access, use the interactive session utility or Slurm's `srun` to schedule one or more compute nodes.

  DO THIS: Start an interactive session on a compute node.
- **That script you wrote to poll job status should probably do so once every few minutes rather than several times a second.**

23.3 Do Not Stress the Shared File Systems

23.4 More File System Tips

23.5 File System Usage Recommendations

23.6 Optimize Input/Output (I/O) Activity

23.7 File Transfer Guidelines

23.8 Job Submission Tips

- **Request Only the Resources You Need** Make sure your job scripts request only the resources that are needed for that job. Don't ask for more time or more nodes than you really need. The scheduler will have an easier time finding a slot for a job requesting 2 nodes for 2 hours, than for a job requesting 4 nodes for 24 hours. This means shorter queue waits times for you and everybody else.
- **Test your submission scripts.** Start small: make sure everything works on 2 nodes before you try 20. Work out submission bugs and kinks with 5 minute jobs that won't wait long in the queue and involve short, simple substitutes for your real workload: simple test problems.
- **Respect memory limits and other system constraints.** If your application needs more memory than is available, your job will fail, and may leave nodes in unusable states. Use ML Cloud Guide sections on the clusters hardware composition to understand the systems limits. Additional dashboard will be available to users to track resource availability.

24. REFERENCES

Additional references and information that may be interesting for ML Cloud users can be found here.

24.1 ML Cloud Documentation

- ML Cloud Usage Policy