Introduction to the MaRC3a Cluster

Hessisches Kompetenzzentrum für Hochleistungsrechnen (HKHLR)

René Sitt

November 10, 2022

HKHLR is funded by the Hessian Ministry of Sciences and Arts

What is an "HPC Cluster"?

- Historically (1960s-1970s): No "personal computers"
  - University computing center is running "The Computer"
- If you had to calculate something:
  - Apply for compute time, drop off the punch cards with your program
  - Wait a few days
  - Come back to collect your results (as a printout)
- ***The same general concept* - with lots of added automation and convenience features - *is still used on HPC clusters even today!*...**
- ...but the needs of both software and users have changed.

Figure: IBM punchcard sorting machine (MZG A4, own picture)

What is an "HPC Cluster"?

- Today: (Almost) everyone has a personal computer...
- **...but the needs of scientific computing have been growing faster than the abilities of personal hardware!**
  - Simulations in physics, chemistry, or meteorology would take days, weeks or even *years* to complete on a current personal computer
  - Data sets are getting too large to store on local diskspace (several Terabytes or even Petabytes)
  - Memory needs are outscaling personal hardware (up to Terabytes, even more for large calculations)
  - Research might need 1000's or 10,000's of calculations (search over parameter space, trainig DNNs, iterating over large data sets)
    - ⋆ These might be individually small, but *only a few can be run concurrently* on a personal computer

What is an "HPC Cluster"?

- Can't research groups just buy and run hardware by themselves? Possible! But:
  - Quite expensive
    - ★ Single 64-core server with 256 GB RAM: ~5,000-15,000 EUR
    - ★ Single 64-core server with 256 GB RAM + 4 server-grade GPUs: ~30,000-40,000 EUR
  - Needs maintenance
    - ★ Additional personnel for administration, updating, support, repair
  - Idle times still cost money
    - ★ Electrical power, cooling
  - Needs appropriate housing
    - ★ Probably can't be run from "under the desk" (fire regulations, noise, power draw)
- **Conclusion: Not every research group - or even department - can afford to buy, house, run, and maintain their own hardware infrastructure for scientific computations!**

What is an "HPC Cluster"?

- Result: University computing centers have moved from having "The Computer" to having "The Large Bunch Of Interconnected Computers"
  - i.e. a **Cluster**
- Using the cluster (mostly) follows the same paradigm as outlined above...
  - Submit your calculation: Define resource needs and what program(s) to run
  - Wait until your calculation completes
  - Collect the results
- ...but you can submit calculations from anywhere (as long as you can login to the cluster)
- ...and lots of your and other's calculations can run concurrently
- ...and complex hierarchies can be in place to automatically (and -ideally- fairly) decide priority of calculations (i.e. who goes first)

What is "MaRC3a"?

- Further problem: HPC clusters do not count as 'basic infrastructure'
  - Difficult to get funding (except for NHR centers or special national projects)
  - Common solution: Project-related funding by one or several departments / research groups / interdisciplinary projects, with the computing center offering consultation, housing, and support
- In case of MaRC3a: Primary stated purpose is "calculations in the field of bioinformatics, pharmaceutical chemistry and theoretical physics"
- Primary project initiator: SYNMIKRO, with additional contributions from several other departments and research groups
  - Hardware contributors get special "owner" privileges (higher priority, longer max. job runtimes) on their own hardware
  - Agreement: Unused compute power is made available to all members of the university

Figure: MaRC3a hardware overview (diagram by Marcus Lechner)

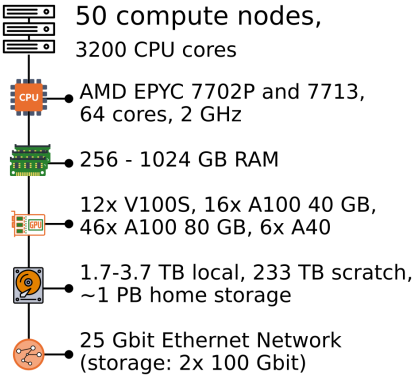
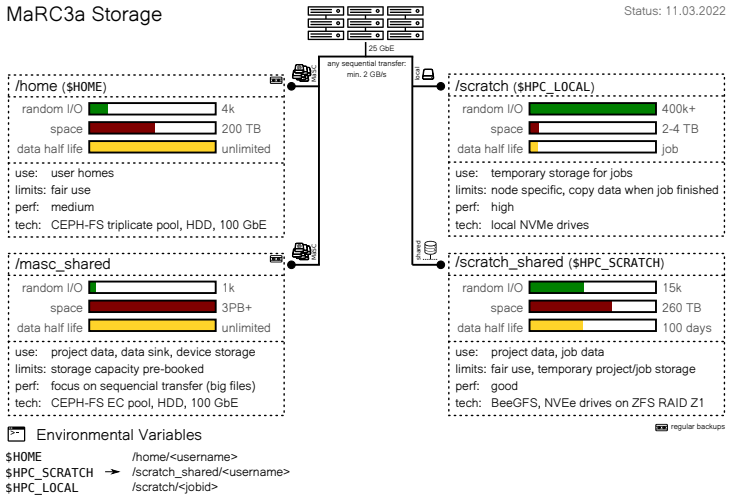


Figure: MaRC3a storage overview (diagram by Marcus Lechner)



Personal requirements

- Must have a university (mail) account (i.e. <user>@staff.uni-marburg.de)
- Leader of research group needs to sign cluster usage agreement
  - Via web form:
    `https://admin.staff.uni-marburg.de/hpc-agreement_en.html`
  - Also names one or more "HPC Managers"
- HPC Managers can then register members of the group for cluster access
  - Also via web form:
    `https://admin.staff.uni-marburg.de/hpc-user_en.html`
  - Need to *register themself as a user* if they want to work on the cluster
- Users need to have their Login Shell set to a value *other than 'rssh'*
  - Web form: `https://admin.staff.uni-marburg.de/aendshell.html`
  - Up to personal preference; default recommendation is 'bash'

Technical requirements

- Software to be used must be able to run on Unix/Linux systems
- Most software (with special exceptions) must be able to run *unattended* and from the commandline, i.e. *no UI*
- Ideally: Software is able to use computational resources *efficiently*
  - Some software packages are already parallelized
    - ⋆ Task: Find out how to explicitly control parallelization
    - ⋆ Task: Find out most efficient parallel settings
  - Some code can be parallelized with additional libraries, e.g. Python or R code
    - ⋆ Task: Find a feasible parallelization solution
  - Some projects just need to perform lots of calculations
    - ⋆ Task: Find a solution to manage job submission and organization of results
    - ⋆ Using builtin functions like job arrays, or scripting, or other additional tools

- Once access has been granted, connecting to the cluster is done via an *SSH client*
- All Linux distributions and current Windows versions (Win 10, Win 11) have a builtin terminal command (`ssh`)
- Alternatively, there are some UI clients:
  - puTTY (lightweight, already installed on many university computers)
    - ⋆ Download available at `https://www.putty.org/`
  - MobaXTerm (all-in-one solution including file browser and XServer for running UI programs on remote machines)
    - ⋆ Download available at `https://mobaxterm.mobatek.net/`

Other than for software on HRZ-installed computers (OPSI), we cannot give support nor warranty for privately installed software. It is *your* responsibility to keep this software up to date and to make sure that it does not pose a security risk!

- **The MaRC3a cluster can only be accessed from the university network!**
  - Computer is at the university and either connected via cable or through eduroam/UMRnet WLAN: No further action required
  - Connecting from home: Only through university VPN! See: `https://www.uni-marburg.de/en/hrz/services/vpn`

Connecting via commandline

- `ssh -X -C -p 223 <username>@marc3.hrz.uni-marburg.de`
- Enter your staff password when prompted

Connecting via UI client

- Hostname: "marc3.hrz.uni-marburg.de", port: 223
- User credentials: Staff username and password

Commandline options

- `scp -P 223 <source> <target>`, remote sources or targets have the format `<user>@<server>:<path>`
- E.g. (with my username "sittr") copying *to* the cluster:
  `scp -P 223 test/some_file.txt sittr@marc3.hrz.uni-marburg.de:/home/sittr/test`
- Copying *from* the cluster:
  `scp -P 223 sittr@marc3.hrz.uni-marburg.de:/home/sittr/test/some_file.txt ./test`
- Copying whole folders: Add '-r' (for "recursive"), i.e.
  `scp -P 223 -r <source> <target>`
- Alternative to SCP: `rsync` (more capabilities, but also more complicated)

GUI options

- SSH clients with builtin file browser (e.g. MobaXTerm)
- Linux file browsers (Nautilus, Nemo, ...) have a "connect to server" function - same settings (hostname, port, user credentials) apply as for connecting via ssh

MaSC storage

- MaRC3 has a direct connection to the MaSC storage cluster
- If you have files in the SMB share on MaSC, these are directly available on MaRC3 under `/masc_shared/<your_group>`
- **Avoid heavy IO directly between MaRC3 jobs and masc_shared!**

Recommended: Enabling passwordless login on MaRC3a

- On your local machine, generate a key pair:

  ```
  ssh-keygen -t ecdsa -b 521
  ```

  Setting a passphrase for the private key is optional but recommended
- Copy the public key to the cluster:

  ```
  scp -P 223 id_ecdsa.pub <user>@marc3.hrz.uni-marburg.de:/home/<user>/.ssh
  ```
- Connect to the cluster (via password) and add the public key to the authorized_keys file:

  ```
  ssh -p 223 <user>@marc3.hrz.uni-marburg.de
  ```

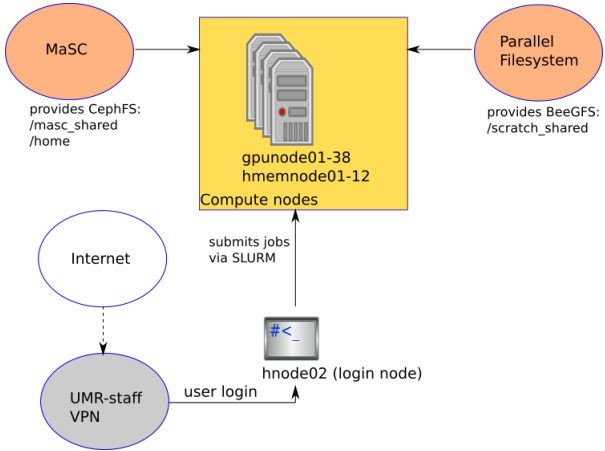
  ```
  cat .ssh/id_ecdsa.pub >> .ssh/authorized_keys
  ```
- Subsequent logins will automatically use the public key instead of the password

- After login, you arrive at the *command prompt*
- The default interface is Linux *bash* (*B*ourne-*A*gain *Sh*ell)
- Everything is done via typing in commands
- We can only give a very limited overview here

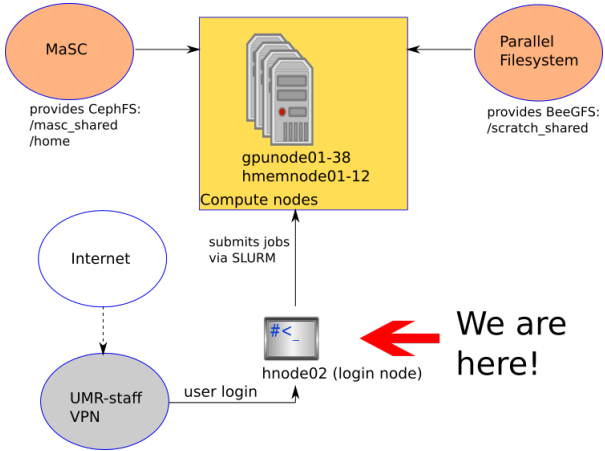
listings/marc3_prompt.sh

```
1 ##########################################################################
2 # Welcome to the MaRC3a system.                                          #
3 ##########################################################################
4 * [...some public messages and general info...]                         *
5 ##########################################################################
6
7 Last login: <date> from <address>
8 [<username>@hnode02 ~]$
```

Cluster Structure

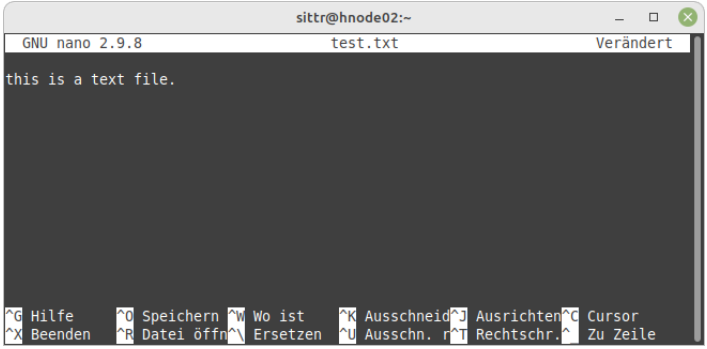


Cluster Structure



- `ls <path>` - list contents of *path*
- `cd <path>` - "change directory" / jump to *path*
- `cp <source> <destination>` - copy file from *source* to *destination*
- `cp -r <source> <destination>` - copy folder including its contents from *source* to *destination*
- `rm <path>` - "remove" / delete file at *path*
- `rm -r <path>` - delete folder at *path* including its contents (**be careful with this!**)
- `man <command>` - open the manual / help page for *command*
- `<command> --help` - print help text for *command*

- `nano` is a (comparatively) beginner-friendly commandline text editor
- Usage: `nano <filename>`
- Commands are displayed at the bottom (`"^"` stands for CTRL/STRG key!)
- Save with STRG+O, exit with STRG+X



- Lots of different software on the cluster means lots of different *dependencies*
- Might want to have several versions of a software or library installed side-by-side
- How to reliably establish a *runtime environment* with the correct versions?

→ **Environment Modules**

- Allows to mix and match software versions to build the correct runtime environment
- Can enforce dependencies and avoid mixing of mutually exclusive software
- Reproducible environment for every job run

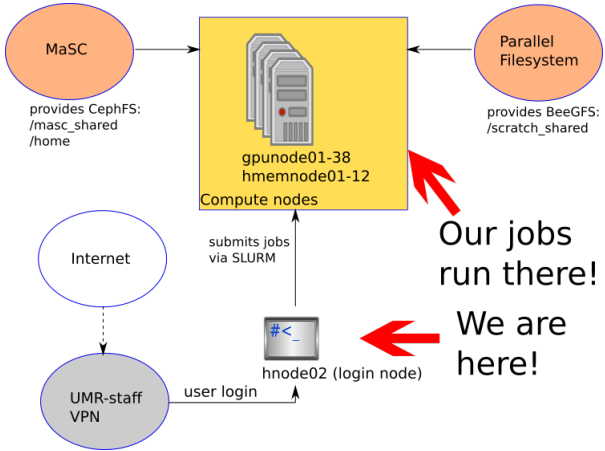
Module Commands

- `module load <modulename>` - load module *modulename*
- `module unload <modulename>` - remove module *modulename*
- `module list` - list loaded modules
- `module avail` - list available modules
- `module spider <modulename>` - list all available versions of *modulename*
- `module purge` - remove *all* currently loaded modules

- Many more commands available - see `module --help`
- Tip: In jobscripts (see below), always lead with a `module purge` to ensure a clean working evironment

So, how do I start my calculation?

- We need a system that distributes available computation resources:
  - According to preset priorities - reflect group membership, access to exclusive / special resources
  - Fairly - everyone gets a turn *eventually*
  - Efficiently - maximize calculation throughput and balance loads
  - Automatically - minimize the need for manual adjustment
- Solution: **Job Scheduling System**
  - Several implementations available, largely same feature set: Oracle Grid Engine (former Sun Grid Engine; used on MaRC2), SLURM, LSF, ...
  - On MaRC3a, we use the "Slurm Workload Manager" (formerly known as Simple Linux Utility for Resource Management)
  - To do its job, the scheduler needs informations about the calculation you want to run


MaSC
provides CephFS:
/masc_shared
/home
Parallel
Filesystem
provides BeeGFS:
/scratch_shared
gpunode01-38
hmemnode01-12
Compute nodes
Our jobs
run there!
Internet
submits jobs
via SLURM
We are
here!
hnode02 (login node)
UMR-staff
VPN
user login

You need to provide:

- *What* the calculation needs (# of processes, amount of memory, ...)
- *Which* environment it needs to run (→ Modules)
- *How* to run it (execution command and program parameters)

The Scheduler decides:

- *Where* to run it (whichever node can provide the stated amount of resources)
  - Less resources required → more opportunities to "fit in"
- *When* to run it (priority of waiting jobs depends on current cluster load, needed resources, user or group membership, ...)
  - Every job will run **eventually**

So, how do I tell the scheduler what it needs to know?

- Via *job script*

SLURM Job Scripts

- Format: Shell script
- Three sections:
  - SLURM parameters (lines start with `#SBATCH`)
  - Environment setup (lines (usually) start with `module`)
  - Calculation command(s) (can take the form of any shell command, most likely something like `<program> <parameters>`)

Job Script Example

listings/marc3_job_example.slurm

```
#!/bin/bash

#SBATCH --time=00:20:00
#SBATCH --ntasks=2
#SBATCH --output=example.out
#SBATCH --mem-per-cpu=100M

module purge
module load gnu9
module load openmpi4

mpicc --version
```

listings/marc3_job_example_comments.sh

```
-> "Shebang" (tells the system the type of script)

-> Set max. job runtime
-> Set amount of processes
-> Set where console output goes
-> Set the needed amount of memory per core

-> Remove all modules to start with a clean environment
-> Load the gcc 9.x module
-> Load the openmpi 4.x module

-> Print the version of "mpicc" and exit
```

Do we actually *need* the resources that are requested here?

Most important parameters (i.e. `#SBATCH` lines):

- `--time=<DD-HH:MM:SS>` → maximum job runtime; default: 3 days
- `--partition=<name>` → which node subset to run on; can be automatically chosen depending on `--time`, `--gpus`, and `--ntasks` requirements; default: "normal"
- `--ntasks=<num_tasks>` → amount of processes; default: 1 **or**
- `--ntasks-per-node=<num_tasks>` **and** `--nodes=<num_nodes>` → explicitly set process distribution
- `--cpus-per-task=<num_threads>` → how many threads to start for each process; default: 1
- `--gpus=[type:]<num>` → amount of GPUs (types: v100s, a100, a40; no type given: can run on any GPU); default: 0
- `--mem=<mem_in_mb>` → amount of *total* memory per node, **or**
- `--mem-per-cpu=<mem_in_mb>` → amount of memory *per cpu*; default: 1000

Some more important parameters:

- `--mail-user=<address>` **and** `--mail-type=<type>` → send a mail for certain events (commonly useful: END; also available: BEGIN, FAIL, ALL, ...)
  - *Only works with university addresses!* (i.e. `<user>@staff.uni-marburg.de`)
  - In case of END, also displays resource utilization (important to gauge reservation parameters for future jobs!)
- `--output=<filename>` → place console output (STDOUT, STDERR) into the set file name; default: `slurm-<job_ID>`
- `--error=<filename>` → place STDERR output into the set file separately; default: goes into the same file as STDOUT
- `--account=<account_name>` → use specified account for this job; only relevant if you are part of multiple groups with different cluster access rights

How do I choose the correct parameters for my job?

- If a job steps over its requirements (memory, time limit), it will be killed
- Goal: Set requirements so a job "fits into them"
- Initially: Approximate by order of magnitude
  - Will it take one hour, one day, ten days to finish? → `--partition=short|normal|long`
  - Will it need 1 GB, 10 GB, 100 GB of memory?
- Use `--mail-type=END` output to check resource utilization and adjust accordingly
- Advantage of setting memory and time limit restrictions: Scheduler can find idle resources where job "fits in" → Reduced waiting times until job start

Consider reserving a bit *less* memory than the node maximum (e.g. `--mem=250G` instead of 256G) - system kernel and OS also need memory, and the scheduler will take "free memory" into account, **not** "theoretically available memory"!

Okay, I have a job script with parameters now. How to submit it?

- `sbatch <job_script>` submits the job script to the scheduler
  - Pay attention to error messages!
  - SLURM will refuse jobs that cannot run in their current configuration
- `sbatch` can also take any SLURM parameters directly (i.e. `sbatch --ntasks=4 <job_script>`; these will *override* the parameters present in the script
- `scancel <job_ID>` cancels / kills a waiting or running job
- `squeue` displays currently running and waiting jobs
- `sinfo` displays cluster partitions and their current state
- `scontrol show job <job_ID>` displays a running job's details
- `sacct` displays finished jobs (default: only from current day)
- `seff <job_ID>` display resource usage statistics for a finished job

My calculation needs interactive commands, can I start interactive jobs?

- Generally yes, **but:**
- Interactive jobs have, by nature, low efficiency from an HPC perspective (i.e. idle times waiting for input)
- Thus, they are quite restricted in order to not lock down large amounts of compute resources:
  - One running job per user
  - Max. 12 hours of running time
  - Max. 16 CPU cores
  - Max. 1 GPU
- If this is not enough, we can offer support to try and find a solution that works for you
- Starting an interactive job works like this:

```
srun <needed resources and parameters, i.e. --ntasks etc.> --pty bash -i
```

- Local support in Marburg is handled through the "Request Tracker" ticket system
  - **marc3@hrz.uni-marburg.de**
- Central mail address that notifies all employees who are involved with the topic
- Can reach support staff even if individual people are on vacation / on sick leave / busy with other things
- MaRC3a staff:
  - René Sitt (HKHLR member, first- and second-level support, software maintenance, administration)
  - Marcus Lechner (MaRC3a + further hardware addons project coordination, administration, additional support)
  - Clemens Thölken (Additional SYNMIKRO position; administration support for MaRC3a and MaSC)
  - Thomas Gebhardt (Head of Central Systems Department, administration, additional support)
  - Manuel Haim (User management)

- Documentation is provided on the University's "share" pages
- Needs login with staff account
- Will be extended over time
- Address:
  `https://share.uni-marburg.de/de/hrz/it-mitarbeiter/hpc/hpc-user/marc3a-masc`
- Specialized subtopics have their own articles:
  `https://share.uni-marburg.de/de/hrz/it-mitarbeiter/hpc/hpc-user/marc3`