Università della Svizzera italiana
Institute of Computing CI

Access to the Cluster

Preparing the environment

Juraj Kardoš,
Olaf Schenk

Università della Svizzera italiana

September 20, 2022

ICS cluster - icsmaster

- 1 login node
  - `icslogin01`
- 42 compute nodes
  - `icsnode[01-42]`
  - 32 nodes only with CPU
    - 2 x Intel E5-2650 v3, 20 (2 x 10) cores
    - 24x64 GB, 8x128 GB RAM
  - 8 nodes with GPU
    - 2 x Intel E5-2650 v3, 20 (2 x 10) cores
    - 1 x NVIDIA GeForce GTX 1080, 2560 CUDA cores
    - 128 GB RAM
  - 2 multi GPU nodes
    - 2 x Intel Xeon Silver 4114, 20 (2 x 10) cores
    - 2 x NVIDIA GeForce GTX 1080 Ti or GTX 2080 Ti
    - 100 GB RAM

Accessing icsmaster

- You will receive your login credentials by email
- Connect to the cluster using ssh

```
$ ssh studXX@hpc.ics.usi.ch
```

- To avoid typing the password you can generate a ssh-key and copy it to the cluster. Execute this **on your laptop**!

```
$ ssh-keygen
$ ssh-copy-id -i ~/.ssh/id_rsa.pub studXX@hpc.ics.usi.ch
```

Note: Copying the command from the pdf slide might not work due to the '~' character!

Accessing icsmaster

- Add host configuration to `~/.ssh/config` **on your laptop**!

```
Host icsmaster
    Hostname hpc.ics.usi.ch
    Port 22
    User studXX
    IdentityFile ~/.ssh/id_rsa
```

- Now you can connect to icsmaster without password

```
$ ssh icsmaster
```

Moving data

- laptop → icsmaster

```
$ scp file.c icsmaster:~/remote_dir/
$ scp -r local_dir icsmaster:~/remote_dir/
```

- icsmaster → laptop

```
$ scp icsmaster:~/remote_dir/file.c local_dir/
$ scp -r icsmaster:~/remote_dir/ local_dir/
```

Modules

- Software on the cluster is organized into modules
- Before using some program, you have to load a module

```
$ module load gcc
```

- You can load a specific version of a module

```
$ module load gcc/10.1.0
```

- You can unload modules you don't want anymore

```
$ module unload gcc
```

- List currently loaded modules

```
$ module list
```

- List all available modules

```
$ module avail
```

Editing code

- To edit files on the cluster you can use vim

```
$ vim hello.c
```

- You can use remote session in IDEs (e.g. VS Code)
- For easier moving (as well es editing) of the files, you can mount home directory from cluster to your local machine. This allows you to work with the files on the cluster as if they are located on your laptop.
  - First install FUSE and SSHFS from `https://osxfuse.github.io`
  - Then you can mount the remote directory

    ```
    $ sshfs icsmaster: <mountpoint>
    ```

    where `<mountpoint>` is an empty directory on your laptop

  - To close the session use

    ```
    $ umount <mountpoint>
    ```

Compiling code

- Load modules before compiling the code

```
$ module load gcc
```

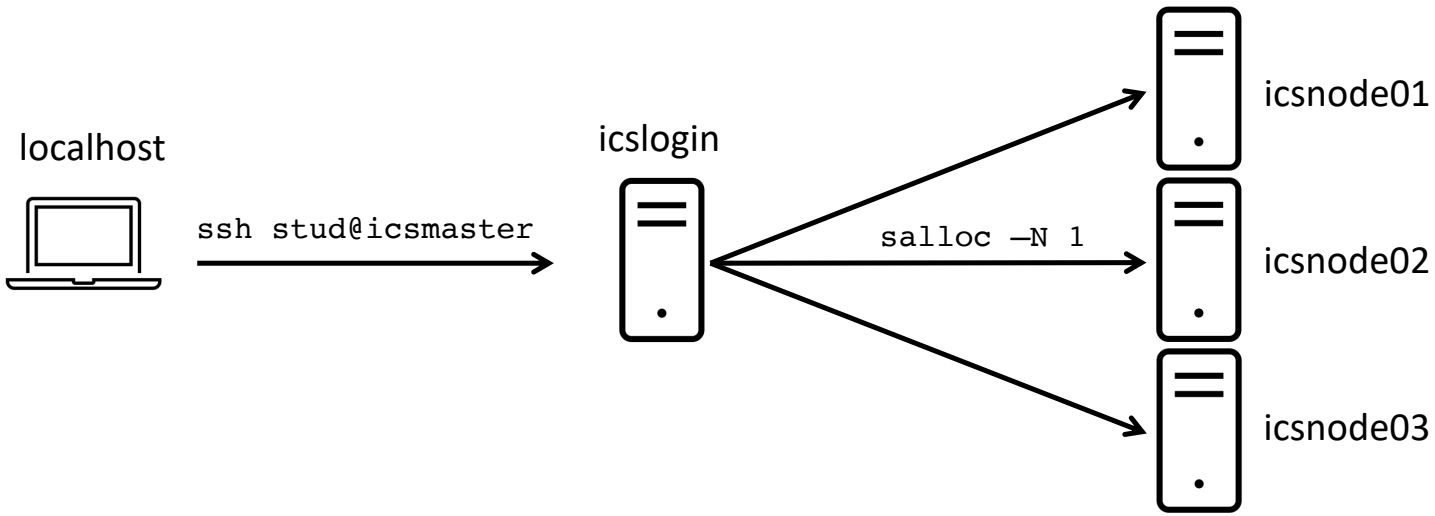
- Then you can compile the code

```
$ gcc hello.c -o hello
```

- Pro tip: add `module load ...` to your `~/.bashrc`
  - It will load modules automatically when you log in

Running code

- When you log in to the cluster, you are on login node `icslogin01`
  - You can use login node to edit and compile your code
  - **But you should never run the code on login node**
- For running your code, there are 42 compute nodes
  - `icsnode[01-42]`
- There are two ways how you can work on compute node
  - Interactive session
  - Batch job



Interactive session

- In interactive session you have direct access to compute node from your terminal
- Interactive session is useful especially for debugging
- When you allocate a node, nobody else can use it at the same time
- But it can take a long time to get access to the node

- First you have to allocate the node
- Let's say you want 1 node for 1 hour

```
$ salloc --nodes=1 --time=01:00:00
```

- Then you can run your app on the compute node

```
$ ./your_app
```

- Or you can use srun that does the allocation automatically

```
$ srun --nodes=1 ./your_app
```

Batch job

- When running batch job, you don't have direct access to the compute node
- You write a script with commands you want execute on the node
- The script is added to a queue and executed later
- Output of the script is written to a file. You can look at is when the job is finished
- Batch job is useful when you have working code and you want to run your app on large data

Batch job

- Job script template

```
#!/bin/bash -l

#SBATCH --job-name=my_job
#SBATCH --time=01:00:00
#SBATCH --nodes=1
#SBATCH --output=%j.out
#SBATCH --error=%j.err

# load modules

# your commands
```

- Add job to a queue

```
$ sbatch job.sh
```

- Show your running and queued jobs

```
$ squeue -u studXX
```

- Cancel job

```
$ scancel <job_id>
```

Reservation

- During the HPC lectures there are several nodes reserved only for you
- Use argument `--reservation=HPC_tuesday` or `--reservation=HPC_wednesday`

```
$ salloc --reservation=HPC_tuesday
$ srun --reservation=HPC_tuesday ./your_app
```

- Use the reservation only in the class
- When the reservation is not active, your job will stay in queue until the next class

- In class (with reservation)

```
$ srun --reservation=HPC_tuesday ./your_app
```

- At home (no reservation)

```
$ salloc --nodes=1 --time=01:00:00
$ ./your_app
```

Specifying resources

- You can call `salloc`, `sbatch` or `srun` with the following parameters

`-N` or `--nodes`	set number of nodes
`-n` or `--ntasks`	set number of tasks
`--mem=MB`	Specify the real memory required per node
`--exclusive`	Only your job is allowed on the nodes allocated to this job

- For debugging use `--mem=10GB`
  - You can share a node with other users
  - It's easier to get a node for you and the others
- For running benchmarks use `--exclusive`
  - To make sure only your job is running on the node
  - Other jobs can't influence your measurements

Git repository with source codes

- We prepared for you a git repository with source codes for this class
  - `https://github.com/oschenk/hpc2022`
- Clone the repository on both icsmaster and your laptop

```
$ git clone https://github.com/oschenk/hpc2022.git
```

- Later we will update the repository with source codes for other lectures and assignments
- To download the latest version use

```
$ git pull
```