Using HLRS Hazel Hen supercomputer Version 2

Theresa Pollinger[1]

[1]Universität Stuttgart

dx.doi.org/10.17504/protocols.io.ud8es9w

Theresa Pollinger

Version 2

Oct 08, 2018

In devel.

ABSTRACT

https://wickie.hlrs.de/platforms/index.php/Cray_XC40

PROTOCOL STATUS

In development
We are still developing and optimizing this protocol

Getting access to Hazel Hen

1 Request access to hazel hen via the ID of your project
https://www.hlrs.de/solutions-services/academic-users/hlrs-systems-access/

Once the form is submitted (as real paper!), the admin(s) in your project can submit your username, password, IPs etc.

Test access and set up a workspace

2 Try access with

COMMAND
```
ssh ipvuser@hazelhen.hww.de
```

with `ipvuser` being your user name. You're in!

Create a workspace with

COMMAND
```
ws_allocate myWorkspace 60
```

or go to a workspace your group has set up for you. They are currently found under

```
/lustre/cray/ws8/ws/
```

cf. https://wickie.hlrs.de/platforms/index.php/Workspace_mechanism

Set up ssh

3 Now on hazel hen you can get a new ssh key with

COMMAND
```
ssh-keygen
```

and copy the public key (= the output of

COMMAND
```
cat ~/.ssh/id_rsa.pub
```

) to all the githubs and gitlabs you'll need. This will allow you to clone onto the machine.

Get code via ssh+git

4 Now back on your own computer (any one on the IP list), you can use a reverse ssh tunnel to clone the relevant repos

COMMAND
```
ssh -R 7890:simsgs.informatik.uni-stuttgart.de:22 ipvuser@hazelhen.hww.de
git clone ssh://git@localhost:7890/heenemo/combi.git /lustre/cray/ws8/ws/ipvuser-myWorkspace/combi
```

remember to replace `7890` by a different port for every git remote server!

Prepare and build

5 From here on, all steps will have to be well documented!

Enter your workspace on hazel hen, check out the versions of the code you need.
Load all the modules you need, e.g. the gnu compiler suite:

COMMAND
```
module load PrgEnv-gnu
```

You can see the available modules with

COMMAND
```
module avail
```

If your build tools / libraries are not there, install them locally for yourself or request them,
cf https://wickie.hlrs.de/platforms/index.php/CRAY_XC40_local_Modules .

Usually you will want to link the program dynamically. To do that, set the environment variable:

COMMAND
```
export CRAYPE_LINK_TYPE=dynamic
```

And you're set to build with your usual toolchain.
Document your modules (`module list`) and environment variables (`declare -p`), and all the changes you made to the code - or better still, `git push` them.

Running jobs

6 request an interactive job on a MOM node with

COMMAND

```
qsub -I -l nodes=1:ppn=24,walltime=01:00:00
```

From here, you can submit the job via aprun (not mpirun!), e.g. with a bash script:

```
#! /bin/bash
#PBS -N <job_name>
#PBS -l nodes=<number_of_nodes>:ppn=24
#PBS -l walltime=00:01:00
cd $PBS_O_WORKDIR # This is the directory where this script and the executable are located.  # You can choose any
other directory on the lustre file system.
export OMP_NUM_THREADS=<nt>
<aprun_command>
```

cf https://wickie.hlrs.de/platforms/index.php/CRAY_XC40_Using_the_Batch_System on how exactly this works