DRAGNET Cluster Usage

Hope this helps... If not let me (Alexander) know (amesfoort@astron.nl).

Access and Login

To get an account, ask Jason Hessels (`hessels@astron.nl`).
With permission from Jason, ask Teun Grit (`grit@astron.nl`) to add access to DRAGNET (via NIS). If you don't have access to the LOFAR portal, tell him. Idem for the ASTRON portal, i.e. if you are not working for ASTRON.

Having an account, ssh to hostname `dragnet.control.lofar` or easier, just **`dragnet`**, from the LOFAR portal (or tunnel through it):

```
ssh USERNAME@dragnet
```

Hostname Hell and Routing Rampage

If you are just running some computations on DRAGNET, skip this section. But if you need fast networking, or are already deep in the slow data transfers and rapid-fire connection errors, here is some info that may save you time wrt the multiple networks and network interfaces. (Or just tell us your needs.)

Hostnames

- dragnet(.control.lofar)
- dragproc(.control.lofar)
- drg01(.control.lofar) - drg23(.control.lofar)

Networks

```
Control/Management network: NODENAME.control.lofar (1 Gb) (all nodes)
10G network:                NODENAME-10g.online.lofar (10 Gb) (all drgXX
nodes and ''dragproc'')
Infiniband network (IPoIB): NODENAME-ib.dragnet.infiniband.lofar (56 Gb)
(all drgXX nodes)
```

(There is also a 1 Gb IPMI network.)

Cross-Cluster

When going cross-cluster, prefer to use the fully-qualified domainnames (FQDN), esp. in scripts or

programs (i.e. drg11-10g.online.lofar instead of just drg11). See `/etc/hosts` on any node for the list.

In most cases, you will use the network as deduced from the destination hostname or IP. Indicate a 10G name to use the 10G network. Idem for infiniband (IPoIB). (Exception: CEP 2, see below.)

Note: Copying large data sets at high bandwidth to/from other clusters (in particular CEP 2) may interfere with running observations as long as CEP 2 is still in use. If you are unsure, ask us. It is ok to use potentially oversubscribed links heavily, but please coordinate with Science Support!

CEP 2

Initiate connections for e.g. data transfers from CEP 2 to HOSTNAME-10g.online.lofar and you will go over 10G.

The reverse, connecting from DRAGNET to CEP 2, by default will connect you via DRAGNET 1G (e.g. for login). To use 10G (e.g. to copy datasets), you need to bind to the local 10G interface name or IP. The program you are using has to support this via e.g. a command-line argument.

SLURM Job Submission

To utilize the cluster efficiently, we use the SLURM workload manager. This is also supposed to ensure that batch jobs do not interfere with observations that DRAGNET participates in (as in: micromize observation data loss).

Random notes:

- SLURM does not enforce accessing nodes through it; one can access any node via ssh. Depending on the intention and the current workload, that may be fine or less desirable.
- SLURM has a ton of options that we haven't all set up. In particular, atm it does not enforce exclusive access to GPUs via cgroups (although it does set `CUDA_VISIBLE_DEVICES` if you explicitly request GPUs). Once a node is (partially) assigned to your program, your program can in principle use any resource on that node.

If you are having trouble using SLURM, please contact Alexander (amesfoort@astron.nl).

Introduction: the trivial stuff

From any DRAGNET node (typically the `dragnet` head node), you can submit compute (or perhaps also separate data transfer) jobs.

Run a single task, see output as it is produced, and wait for completion. Note that in this case the `ls` program must be available on any node that may be used.

```
$ srun -n 1 ls dir1 dir2
file1
file2
[...]
```

Show list of jobs queued:

```
$ squeue
           JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
               9   workers       ls amesfoor CD       0:01      1 drg
```

Show list of recently completed jobs:

```
$ squeue -t COMPLETED
           JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
               9   workers       ls amesfoor CD       0:01      1 drg
```

Show list and state of nodes. When submitting a job, you can indicate one of the partitions listed or a (not necessarily large enough) set of nodes that must be used. Please hesitate indefinitely when trying to submit insane loads to the head partition. :)

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
workers*     up   infinite     23   idle drg[01-23]
proc         up   infinite      1   idle dragproc
head         up   infinite      1   idle dragnet
```

If you get an error on job submission that there are no resources in the cluster to ever satisfy your job, and you know this is wrong (no typo), you can see with the `sinfo` if there are nodes out of service. (SLURM may remove a node from a partition on misconfiguration or hardware malfunctioning.)

Hints on using more SLURM capabilities

The sbatch(1) command offers to:

- take a user-supplied job (batch) script, not just to start your script, but also to set up a job array or workflow
- have stdout/stderr go to a file
- copy the program (and possibly library and data dependencies) to the to be used nodes
- run the job without blocking your terminal on its completion. This is useful for e.g. substantial processing jobs
- auto-restart on failure (not sure when/how that applies)

Apart from nodes, it is also possible to indicate scheduling constraints on CPU cores, GPUs, memory, or network bandwidth (if we set that up).
Atm, you have to indicate constraints for:

- either number of nodes or CPUs
- number of GPUs, if any needed. If no GPUs are requested, any GPU program will fail. (Btw, this policy is not fully as intended, so if technically it can be improved, we can look into it.)

You do not have to indicate memory size, but if you don't, SLURM will grant you all the memory of a node, preventing other jobs from running on the same node(s). This may or may not be the intention. (If the intention, better use --exclusive.)

Note that a CPU is to SLURM a hardware resource that the OS can schedule a task on. This is typically a hardware thread, or if no hyperthreading, a CPU core.

To indicate a scheduling resource constraint on 2 GPUs, use the –gres option (*gres* stands for *generic resource*):

```
$ srun --gres=gpu:2 -n 1 your_gpu_prog
```

To indicate a list of nodes that must be used (list may be smaller than number of nodes requested). Some examples:

```
$ srun --nodelist=drg02 ls
$ srun --nodelist=drg05-drg07,drg22 -n 8 ls
$ srun --nodelist=./nodelist.txt ls   # with a '/' in the arg value
```

For the moment, see more explanation and examples at
http://www.umbc.edu/hpcf/resources-tara-2013/how-to-run.php

Please see the manual pages on srun(1), sbatch(1), salloc(1) and the SLURM website for more info.