Wikiprint Book

Title: Information about the batch system (SLURM)

Subject: DEEP - Public/User_Guide/Batch_system

Version: 63

Date: 19.05.2024 20:16:34

Table of Contents

Information about the batch system (SLURM)

The DEEP prototype systems are running SLURM for resource management. Documentation of Slurm can be found ?here.

Overview

Slurm offers interactive and batch jobs (scripts submitted into the system). The relevant commands are `srun` and `sbatch`. The `srun` command can be used to spawn processes (**please do not use mpiexec**), both from the frontend and from within a batch script. You can also get a shell on a node to work locally there (e.g. to compile your application natively for a special platform or module).

Available Partitions

Please note that there is no default partition configured. In order to run a job, you have to specify one of the following partitions, using the `--partition=...` switch:

Name	Description
dp-cn	dp-cn[01-50], DEEP-EST Cluster nodes (Xeon Skylake)
dp-dam	dp-dam[01-16], DEEP-EST Dam nodes (Xeon Cascadelake + 1 V100 + 1 Stratix 10)
dp-dam-ext	dp-dam[09-16], DEEP-EST Dam nodes connected with Extoll Tourmalet
dp-esb-ib	dp-esb[26-75], DEEP-EST ESB nodes connected with IB EDR (Xeon Cascadelake + 1 V100)
dp-esb-ext	dp-esb[01-25], DEEP-EST ESB nodes connected with Extoll Fabri3)
dp-sdv-esb	dp-sdv-esb[01-02], DEEP-EST ESB Test nodes (Xeon Cascadelake + 1 V100)
ml-gpu	ml-gpu[01-03], GPU test nodes for ML applications (4 V100 cards)
knl	knl[01,04-06], KNL nodes
knl256	knl[01,05], KNL nodes with 64 cores
knl272	knl[04,06], KNL nodes with 68 cores
snc4	knl[05], KNL node in snc4 memory mode
debug	all compute nodes (no gateways)

Anytime, you can list the state of the partitions with the `sinfo` command. The properties of a partition can be seen using

```
scontrol show partition <partition>
```

Remark about environment

By default, Slurm passes the environment from your job submission session directly to the execution environment. Please be aware of this when running jobs with `srun` or when submitting scripts with `sbatch`. This behavior can be controlled via the `--export` option. Please refer to the ?Slurm documentation to get more information about this.

In particular, when submitting job scripts, **it is recommended to load the necessary modules within the script and submit the script from a clean environment.**

An introductory example

Suppose you have an mpi executable named `hello_mpi`. There are three ways to start the binary.

From a shell on a node

First, start a shell on a node. Assume you would like to run your mpi task on 4 cluster nodes with 2 tasks per node:

```
[kreutz1@deepv /p/project/cdeep/kreutz1/Temp]$ srun -A deep -p dp-cn -N 4 -n 8 -t 00:30:00 --pty --interactive /bin/bash -
[kreutz1@dp-cn01 /p/project/cdeep/kreutz1/Temp]$
```

The environment is transported to the remote shell, no `.profile`, `.bashrc`, ... are sourced (especially not the modules default from `/etc/profile.d/modules.sh`). As of March 2020, an account has to be specified using the `--account` (short `-A`) option, which is "deep" for the project members. For people not included in the DEEP-EST project, please use the "Budget" name you received along with your account creation.

Once you get to the compute node, start your application using `srun`. Note that the number of tasks used is the same as specified in the initial `srun` command above (4 nodes with two tasks each):

```
[kreutz1@deepv Temp]$ salloc -A deep -p dp-cn -N 4 -n 8 -t 00:30:00 srun --pty --interactive /bin/bash -i
[kreutz1@dp-cn01 Temp]$ srun -N 2 -n 8 ./MPI_HelloWorld
Hello World from rank 3 of 8 on dp-cn02
Hello World from rank 7 of 8 on dp-cn04
Hello World from rank 2 of 8 on dp-cn02
Hello World from rank 6 of 8 on dp-cn04
Hello World from rank 0 of 8 on dp-cn01
Hello World from rank 4 of 8 on dp-cn03
Hello World from rank 1 of 8 on dp-cn01
Hello World from rank 5 of 8 on dp-cn03
```

You can ignore potential warnings about the cpu binding. ParaStation will pin your processes.

If you just need to one node to run your interactive session on you can simply use the `srun` command (without `salloc`), e.g.:

```
[kreutz1@deepv ~]$ srun -A deep -N 1 -n 8 -p dp-cn -t 00:30:00 --pty --interactive bash -i
[kreutz1@dp-cn22 ~]$ srun -n 8 hostname
dp-cn22
dp-cn22
dp-cn22
dp-cn22
dp-cn22
dp-cn22
dp-cn22
dp-cn22
```

Running directly from the front ends

You can run the application directly from the frontend, bypassing the shell. Do not forget to set the correct environment for running your executable on the login node as this will be used for execution with `srun`.

```
[kreutz1@deepv Temp]$ ml GCC/10.3.0 ParaStationMPI/5.4.9-1
[kreutz1@deepv Temp]$ srun -A deep -p dp-cn -N 4 -n 8 -t 00:30:00 ./MPI_HelloWorld
Hello World from rank 7 of 8 on dp-cn04
Hello World from rank 3 of 8 on dp-cn02
Hello World from rank 6 of 8 on dp-cn04
Hello World from rank 2 of 8 on dp-cn02
Hello World from rank 4 of 8 on dp-cn03
Hello World from rank 0 of 8 on dp-cn01
Hello World from rank 1 of 8 on dp-cn01
Hello World from rank 5 of 8 on dp-cn03
```

It can be useful to create an allocation which can be used for several runs of your job:

```
[kreutz1@deepv Temp]$ salloc -A deep -p dp-cn -N 4 -n 8 -t 00:30:00
salloc: Granted job allocation 69263
[kreutz1@deepv Temp]$ srun ./MPI_HelloWorld
Hello World from rank 7 of 8 on dp-cn04
Hello World from rank 3 of 8 on dp-cn02
Hello World from rank 6 of 8 on dp-cn04
```

```
Hello World from rank 2 of 8 on dp-cn02
Hello World from rank 5 of 8 on dp-cn03
Hello World from rank 1 of 8 on dp-cn01
Hello World from rank 4 of 8 on dp-cn03
Hello World from rank 0 of 8 on dp-cn01
...
# several more runs
...
[kreutz1@deepv Temp]$ exit
exit
salloc: Relinquishing job allocation 69263
```

Note that in this case the `-N` and `-n` options for the `srun` command can be skipped (they default to the corresponding options given to `salloc`).

Batch script

As stated above, it is recommended to load the necessary modules within the script and submit the script from a clean environment.

The following script `hello_cluster.sh` will unload all modules and load the modules required for executing the given binary:

```
#!/bin/bash

#SBATCH --partition=dp-esb
#SBATCH -A deep
#SBATCH -N 4
#SBATCH -n 8
#SBATCH -o /p/project/cdeep/kreutz1/hello_cluster-%j.out
#SBATCH -e /p/project/cdeep/kreutz1/hello_cluster-%j.err
#SBATCH --time=00:10:00

ml purge
ml GCC ParaStationMPI
srun ./MPI_HelloWorld
```

This script requests 4 nodes of the ESB module with 8 tasks, specifies the stdout and stderr files, and asks for 10 minutes of walltime. You can submit the job script as follows:

```
[kreutz1@deepv Temp]$ sbatch hello_cluster.sh
Submitted batch job 69264
```

… and check what it's doing:

```
[kreutz1@deepv Temp]$ squeue -u $USER
            JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            69264     dp-cn hello_cl  kreutz1 CG       0:04      4 dp-cn[01-04]
```

Once finished, you can check the result (and the error file if needed)

```
[kreutz1@deepv Temp]$ cat /p/project/cdeep/kreutz1/hello_cluster-69264.out
Hello World from rank 7 of 8 on dp-esb37
Hello World from rank 3 of 8 on dp-esb35
Hello World from rank 5 of 8 on dp-esb36
Hello World from rank 1 of 8 on dp-esb34
Hello World from rank 6 of 8 on dp-esb37
Hello World from rank 2 of 8 on dp-esb35
Hello World from rank 4 of 8 on dp-esb36
Hello World from rank 0 of 8 on dp-esb34
```

Job chains

Please refer to the FAQ for creation of job chains and implementing job dependencies. If you would like to implement workflows, take a look at the Workflows section.

Information on past jobs and accounting

The `sacct` command can be used to enquire the Slurm database about a past job.

```
[kreutz1@deepv Temp]$ sacct -j 69268
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
69268+0            bash      dp-cn deepest-a+         96  COMPLETED      0:0
69268+0.0    MPI_Hello+            deepest-a+          2  COMPLETED      0:0
69268+1            bash     dp-dam deepest-a+        384  COMPLETED      0:0
```

On the Cluster (CM) nodes it's possible to query the consumed energy for a certain job:

```
[kreutz1@deepv kreutz1]$ sacct -o ConsumedEnergy,JobName,JobID,CPUTime,AllocNodes -j 69326
ConsumedEnergy    JobName        JobID    CPUTime AllocNodes
-------------- ---------- ------------ ---------- ----------
      496.70K hpl_MKL_O+ 69326          16:28:48          1
            0      batch 69326.batch    16:28:48          1
      496.70K xlinpack_+ 69326.0        08:10:24          1
```

This feature will also be for the ESB nodes.

FAQ

Is there a cheat sheet for all main Slurm commands?

Yes, it is available ?here.

Why's my job not running?

You can check the state of your job with

```
scontrol show job <job id>
```

In the output, look for the `Reason` field.

You can check the existing reservations using

```
scontrol show res
```

How can I check which jobs are running in the machine?

Please use the `squeue` command (the "-u $USER" option to only list jobs belonging to your user id).

How do I do chain jobs with dependencies?

Please confer the `sbatch`/`srun` man page, especially the

```
-d, --dependency=<dependency_list>
```

entry.

Also, jobs can be chained after they have been submitted using the `scontrol` command by updating their `Dependency` field.

How can check the status of partitions and nodes?

The main command to use is `sinfo`. By default, when called alone, `sinfo` will list the available partitions and the number of nodes in each partition in a given status. For example:

```
[deamicis1@deepv hybridhello]$ sinfo
PARTITION    AVAIL  TIMELIMIT  NODES  STATE NODELIST
sdv             up   20:00:00     11   idle deeper-sdv[06-16]
knl             up   20:00:00      1  drain knl01
knl             up   20:00:00      3   idle knl[04-06]
knl256          up   20:00:00      1  drain knl01
knl256          up   20:00:00      1   idle knl05
knl272          up   20:00:00      2   idle knl[04,06]
snc4            up   20:00:00      1   idle knl05
extoll          up   20:00:00     11   idle deeper-sdv[06-16]
ml-gpu          up   20:00:00      3   idle ml-gpu[01-03]
dp-cn           up   20:00:00      1  drain dp-cn33
dp-cn           up   20:00:00      5   resv dp-cn[09-10,25,49-50]
dp-cn           up   20:00:00     44   idle dp-cn[01-08,11-24,26-32,34-48]
dp-dam          up   20:00:00      1 drain* dp-dam08
dp-dam          up   20:00:00      2  drain dp-dam[03,07]
dp-dam          up   20:00:00      3   resv dp-dam[05,09-10]
dp-dam          up   20:00:00      2  alloc dp-dam[01,04]
dp-dam          up   20:00:00      8   idle dp-dam[02,06,11-16]
dp-dam-ext      up   20:00:00      2   resv dp-dam[09-10]
dp-dam-ext      up   20:00:00      6   idle dp-dam[11-16]
dp-esb          up   20:00:00     51 drain* dp-esb[11,26-75]
dp-esb          up   20:00:00      2  drain dp-esb[08,23]
dp-esb          up   20:00:00      2  alloc dp-esb[09-10]
dp-esb          up   20:00:00     20   idle dp-esb[01-07,12-22,24-25]
dp-sdv-esb      up   20:00:00      2   resv dp-sdv-esb[01-02]
psgw-cluster    up   20:00:00      1   idle nfgw01
psgw-booster    up   20:00:00      1   idle nfgw02
debug           up   20:00:00      1 drain* dp-dam08
debug           up   20:00:00      4  drain dp-cn33,dp-dam[03,07],knl01
debug           up   20:00:00     10   resv dp-cn[09-10,25,49-50],dp-dam[05,09-10],dp-sdv-esb[01-02]
debug           up   20:00:00      2  alloc dp-dam[01,04]
debug           up   20:00:00     69   idle deeper-sdv[06-16],dp-cn[01-08,11-24,26-32,34-48],dp-dam[02,06,11-16],knl[04-06
```

Please refer to the man page for `sinfo` for more information.

Can I join stderr and stdout like it was done with `-joe` in Torque?

Not directly. In your batch script, redirect stdout and stderr to the same file:

```
...
#SBATCH -o /point/to/the/common/logfile-%j.log
#SBATCH -e /point/to/the/common/logfile-%j.log
...
```

(The `%j` will place the job id in the output file). N.B. It might be more efficient to redirect the output of your script's commands to a dedicated file.