

## Table of Contents

<b>Information about the batch system (SLURM)</b>	<b>2</b>
Overview	2
Remark about environment	2
An introductory example	2
From a shell on a node	2
Running directly from the front ends	2
Batch script	3
Available Partitions	4
FAQ	4
Why's my job not running?	4
How can I check which jobs are running in the machine?	4
How do I do chain jobs with dependencies?	4
How can get a list of broken nodes?	4
Can I join stderr and stdout like it was done with -joe in Torque?	4
What's the equivalent of qsub -l nodes=x:ppn=y:cluster+n_b:ppn=p_b:booster?	5
pbs/slurm dictionary	6

## Information about the batch system (SLURM)

Please confer `/etc/slurm/README`.

The 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).

### 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. You would like to run your mpi task on 4 machines with 2 tasks per machine:

```
niessen@deepl:src/mpi > srun --partition=sdv -N 4 -n 8 --pty /bin/bash -i
niessen@deeper-sdv04:/direct/homec/zdvex/niessen/src/mpi >
```

The environment is transported to the remote shell, no `.profile`, `.bashrc`, ... are sourced (especially not the modules default from `/etc/profile.d/modules.sh`).

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):

```
niessen@deeper-sdv04:/direct/homec/zdvex/niessen/src/mpi > srun ./hello_cluster
srun: cluster configuration lacks support for cpu binding
Hello world from process 6 of 8 on deeper-sdv07
Hello world from process 7 of 8 on deeper-sdv07
Hello world from process 3 of 8 on deeper-sdv05
Hello world from process 4 of 8 on deeper-sdv06
Hello world from process 0 of 8 on deeper-sdv04
Hello world from process 2 of 8 on deeper-sdv05
Hello world from process 5 of 8 on deeper-sdv06
Hello world from process 1 of 8 on deeper-sdv04
```

You can ignore the warning about the cpu binding. ParaStation will pin you processes.

#### Running directly from the front ends

You can run the application directly from the frontend, bypassing the shell:

```
niessen@deepl:src/mpi > srun --partition=sdv -N 4 -n 8 ./hello_cluster
Hello world from process 4 of 8 on deeper-sdv06
Hello world from process 6 of 8 on deeper-sdv07
Hello world from process 3 of 8 on deeper-sdv05
Hello world from process 0 of 8 on deeper-sdv04
```

```
Hello world from process 2 of 8 on deeper-sdv05
Hello world from process 5 of 8 on deeper-sdv06
Hello world from process 7 of 8 on deeper-sdv07
Hello world from process 1 of 8 on deeper-sdv04
```

In this case, it can be useful to create an allocation which you can use for several runs of your job:

```
niessen@deepl:src/mpi > salloc --partition=sdv -N 4 -n 8
salloc: Granted job allocation 955
niessen@deepl:~/src/mpi>srunch ./hello_cluster
Hello world from process 3 of 8 on deeper-sdv05
Hello world from process 1 of 8 on deeper-sdv04
Hello world from process 7 of 8 on deeper-sdv07
Hello world from process 5 of 8 on deeper-sdv06
Hello world from process 2 of 8 on deeper-sdv05
Hello world from process 0 of 8 on deeper-sdv04
Hello world from process 6 of 8 on deeper-sdv07
Hello world from process 4 of 8 on deeper-sdv06
niessen@deepl:~/src/mpi> # several more runs
...
niessen@deepl:~/src/mpi>exit
exit
salloc: Relinquishing job allocation 955
```

### Batch script

Given the following script `hello_cluster.sh`: (it has to be executable):

```
#!/bin/bash

#SBATCH --partition=sdv
#SBATCH -N 4
#SBATCH -n 8
#SBATCH -o /homec/zdvex/niessen/src/mpi/hello_cluster-%j.log
#SBATCH -e /homec/zdvex/niessen/src/mpi/hello_cluster-%j.err
#SBATCH --time=00:10:00

srunch ./hello_cluster
```

This script requests 4 nodes with 8 tasks, specifies the stdout and stderr files, and asks for 10 minutes of walltime. Submit:

```
niessen@deepl:src/mpi > sbatch ./hello_cluster.sh
Submitted batch job 956
```

Check what it's doing:

```
niessen@deepl:src/mpi > squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
956	sdv	hello_cl	niessen	R	0:00	4	deeper-sdv[04-07]

Check the result:

```
niessen@deepl:src/mpi > cat hello_cluster-956.log
Hello world from process 5 of 8 on deeper-sdv06
Hello world from process 1 of 8 on deeper-sdv04
Hello world from process 7 of 8 on deeper-sdv07
Hello world from process 3 of 8 on deeper-sdv05
Hello world from process 0 of 8 on deeper-sdv04
Hello world from process 2 of 8 on deeper-sdv05
```

```
Hello world from process 4 of 8 on deeper-sdv06
Hello world from process 6 of 8 on deeper-sdv07
```

## 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:

- `dp-cn`: The DEEP-EST cluster nodes
- `dp-dam`: The DEEP-EST DAM nodes
- `sdv`: The DEEP-ER sdv nodes
- `kn1`: The DEEP-ER kn1 nodes (all of them, regardless of cpu and configuration)
- `kn1256`: the 256-core kn1s
- `kn1272`: the 272-core kn1s
- `snc4`: the kn1s configured in SNC-4 mode
- `ml-gpu`: the machine learning nodes equipped with 4 Nvidia Tesla V100 GPUs each
- `extoll`: the sdv nodes in the extoll fabric (**KNL nodes not on Extoll connectivity anymore!**)
- `dam`: prototype dam nodes, two of which equipped with Intel Arria 10G FPGAs.

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>
```

## FAQ

### 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.

### How do I do chain jobs with dependencies?

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

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

entry.

### How can get a list of broken nodes?

The command to use is

```
sinfo -Rl -h -o "%n %12U %19H %6t %E" | sort -u
```

See also the translation table below.

### 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.

### What's the equivalent of `qsub -l nodes=x:ppn=y:cluster+n_b:ppn=p_b:booster?`

As of version 17.11 of Slurm, heterogeneous jobs are supported. For example, the user can run:

```
srunk --partition=sdv -N 1 -n 1 hostname : --partition=kn1 -N 1 -n 1 hostname
deeper-sdv01
kn105
```

In order to submit a heterogeneous job, the user needs to set the batch script similarly to the following:

```
#!/bin/bash

#SBATCH --job-name=imb_execute_1
#SBATCH --account=deep
#SBATCH --mail-user=
#SBATCH --mail-type=ALL
#SBATCH --output=job.out
#SBATCH --error=job.err
#SBATCH --time=00:02:00

#SBATCH --partition=sdv
#SBATCH --constraint=
#SBATCH --nodes=1
#SBATCH --ntasks=12
#SBATCH --ntasks-per-node=12
#SBATCH --cpus-per-task=1

#SBATCH packjob

#SBATCH --partition=kn1
#SBATCH --constraint=
#SBATCH --nodes=1
#SBATCH --ntasks=12
#SBATCH --ntasks-per-node=12
#SBATCH --cpus-per-task=1

srunk ./app_sdv : ./app_kn1
```

Here the `packjob` keyword allows to define Slurm parameter for each sub-job of the heterogeneous job.

If you need to load modules before launching the application, it's suggested to create wrapper scripts around the applications, and submit such scripts with `srunk`, like this:

```
...
srunk ./script_sdv.sh : ./script_kn1.sh
```

where a script should contain:

```
#!/bin/bash

module load ...
```

```
./app_sdv
```

This way it will also be possible to load different modules on the different partitions used in the heterogeneous job.

### pbs/slurm dictionary

PBS command	closest slurm equivalent
qsub	sbatch
qsub -l	salloc + srun —pty bash -i
qsub into an existing reservation	... —reservation= <reservation> ...
pbsnodes	scontrol show node
pbsnodes (-ln)	sinfo (-R) or sinfo -RI -h -o "%n %12U %19H %6t %E"   sort -u
pbsnodes -c -N n <node>	scontrol update <a href="#">NodeName?</a> = <node> State=RESUME
pbsnodes -o <node>	scontrol update <a href="#">NodeName?</a> = <node> State=DRAIN reason="some comment here"
pbstop	smap
qstat	squeue
checkjob <job>	scontrol show job <job>
checkjob -v <job>	scontrol show -d job <job>
showres	scontrol show res
setres	scontrol create reservation <a href="#">ReservationName?</a> = <reservation> user=partec Nodes=j3c!053-056] <a href="#">StartTime?</a> =now duration=Unlimited Flags=IGNORE_JOBS
setres -u <user> ALL	scontrol create reservation <a href="#">ReservationName?</a> =\<some name> user=\<user> Nodes=ALL startTime=now duration=unlimited FLAGS=maint,ignore_jobs
releaseres	scontrol delete <a href="#">ReservationName?</a> = <reservation>