جامعة نيويورك أبوظبي

NYU | ABU DHABI

Getting Started with *Dalma*

High Performance Computing NYUAD

November 2017 v6.1

جامعة نيويورك أبوظبي
NYU | ABU DHABI

SLURM: Partitions

- A partition is a collection of nodes, they may share some attributes (CPU type, GPU, etc)
- Compute nodes may belong to multiple partitions to ensure maximum use of the system
- Partitions may have different priorities and limits of execution and may limit who can use them
- Dalma's partition (as seen by users)
  - **`serial`** run single core and multi-threaded jobs (eg single node)
  - **`parallel`** run MPI jobs (eg multi-node)
  - **`bigmem`** run jobs on large memory systems only
  - **`butinah`** run jobs on SSE2 (Westmere) processors only (for serial-integer apps)
  - **`visual`** run jobs on systems with a graphics card only
- There are other partitions, but these are reserved for specific groups and research projects
- For those who are experts in SLURM we use partitions to request GPUs, large memory, and visual instead of "constraints" as this approach gives us more flexibility for priorities and resource limits.

SLURM: Submitting Jobs

- To submit a job first you write a "job script"

```
#!/bin/bash
#SBATCH –p serial
#SBATCH –n 1
./myprogram
```

- Then you submit the script in any of the following manner

```
> sbatch job.sh
```

OR

```
> sbatch < job.sh
```

OR

```
> sbatch << EOF
#!/bin/bash
#SBATCH –p serial
#SBATCH –n 1
./myprogram
EOF
```

SLURM: Arguments

- Arguments to "sbatch" can be put on the command line or embedded in the job script
- Putting them in the job script is a better option as then it "documents" how to rerun your job

job1.sh

```
#!/bin/bash
#SBATCH –p serial
#SBATCH –n 1
./myprogram
```

```
> sbatch job1.sh
```

OR

job2.sh

```
#!/bin/bash
./myprogram
```

```
> sbatch –p serial –n 1 job2.sh
```

SLURM: Arguments

Common Job submission arguments:

`-n`	Select number of tasks to run (default 1 core per task)
`-C`	Select required system feature (eg avx2, sse, gpu)
`-N`	Select number of nodes on which to run
`-t`	Wallclock in hours:minutes:seconds (ex 4:00:00)
`-p`	Select partition (serial, parallel, gpu, bigmem)
`-o`	Output file (with no –e option, err and out are merged to the Outfile)
`-e`	Keep a separate error File
`-d`	Dependency with prior job (ex don't start this job before job XXX terminates)
`-A`	Select account (ex physics_ser, faculty_ser)
`-c`	Number of cores required per task (default 1)
`--tasks-per-node`	Number of tasks on each node
`--mail-type=type`	Notify on state change: BEGIN, END, FAIL or ALL
`--mail-user=user`	Who to send email notification
`--mem`	Maximum amount of memory per job (default is in MB, but can use GB suffix)
	(Note: not all memory is available to jobs, 8GB is reserved on each node for the OS)
	(So a 128GB node can allocate up to 120GB for jobs)

SLURM: Job Dependencies

- **Submitting with dependencies:** Useful to create workflows
  - Any specific job may have to wait until any of the specified conditions are met
  - These conditions are set with –d type:jobid where type can be:
    - **after** run after <jobid> has terminated
    - **afterany** if <jobid> is a job array run after any job in the job array has terminated
    - **afterok** run after <jobid> if it finished successfully
    - **afternotok** run after <jobid> if it failed to finish successfully

```
# Wait for specific job array elements
sbatch --depend=after:123_4 my.job
sbatch --depend=afterok:123_4:123_8 my.job2

# Wait for entire job array to complete
sbatch --depend=afterany:123 my.job

# Wait for entire job array to complete successfully
sbatch --depend=afterok:123 my.job

# Wait for entire job array to complete and at least one task fails
sbatch --depend=afternotok:123 my.job
```

SLURM: Listing Jobs

- Each submitted job is given a unique number
- You can list your jobs to see which ones are waiting (pending), running
- As well as how long a job has been running and on which node(s)

```
> squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
 435251_[1-50]   ser_std 151215_F     u123 PD       0:00      1 (Priority)
 435252_[1-50]   ser_std 151215_F     u123 PD       0:00      1 (Priority)
        435294   ser_std Merge5.s     u123 PD       0:00      1 (Priority)
435235_[20-50]   ser_std 151215_F     u123 PD       0:00      1 (Priority)
     435235_19   ser_std 151215_F     u123  R      12:55      1 compute-21-8
     435235_17   ser_std 151215_F     u123  R      47:34      1 compute-21-12
     435235_15   ser_std 151215_F     u123  R      49:04      1 compute-21-7
     435235_13   ser_std 151215_F     u123  R      50:34      1 compute-21-4
     435235_11   ser_std 151215_F     u123  R      54:35      1 compute-21-9
      435235_9   ser_std 151215_F     u123  R      56:35      1 compute-21-6
      435235_7   ser_std 151215_F     u123  R      58:35      1 compute-21-5
      435235_5   ser_std 151215_F     u123  R      59:36      1 compute-21-1
      435235_3   ser_std 151215_F     u123  R    1:00:36      1 compute-21-11
      435235_1   ser_std 151215_F     u123  R    1:04:37      1 compute-21-3
```

SLURM: Listing Jobs

- You can look at completed jobs using the "sacct" command
- To look at jobs you ran since July 1, 2017

```
> squeue –starttime=2017-07-01
```

- You can retrieve the following informations about a job after it terminates:

```
JobID             JobIDRaw          JobName     Partition   MaxVMSize   MaxVMSizeNode
MaxVMSizeTask     AveVMSize         MaxRSS      MaxRSSNode  MaxRSSTask  AveRSS
MaxPagesNode      MaxPagesTask      AvePages    MinCPU      MinCPUNode  MinCPUTask
NTasks            AllocCPUS         Elapsed     State       ExitCode    AveCPUFreq
MaxPages          AveCPU            ReqCPUFreqMin           ReqCPUFreqMax
ReqCPUFreqGov     ReqMem            ConsumedEnergy          MaxDiskRead
MaxDiskReadNode   MaxDiskReadTask   AveDiskRead             MaxDiskWrite
MaxDiskWriteNode  MaxDiskWriteTask  AveDiskWrite            AllocGRES
ReqGRES           ReqTRES           AllocTRES
```

- To retrieve specific informations about a job

```
> sacct -j 466281 -format=partition,alloccpus,elapsed,state,exitcode
             JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
466281          job3.sh    par_std   cpcm_par         56  COMPLETED      0:0
466281.batch      batch              cpcm_par         28  COMPLETED      0:0
466281.0            env              cpcm_par         56  COMPLETED      0:0
```

SLURM: Job Progress

- You can see your job's progress by looking at the output and error files
- By default output and error files are named "slurm-XXX.out" and "slurm-XXX.err" where XXX is the job id
- `"tail –f"` allows you to track new output as it is produced

```
> cat slurm-435563.out

> more slurm-435563.out

> tail –f slurm-435563.out
```

SLURM: Killing Jobs

- Sometimes you need to kill your job when you realise it is not working as expected
- Note that your job can be killed automatically when it reaches its maximum time/memory allocation

```
> scancel 435563
```

SLURM: tasks

SLURM "tasks"

#SBATCH –n 2 VS #PBS –l nodes=1:ppn=2

In SLURM users specify how many tasks - not cores! - they need (-n). Each task by default uses 1 core. But this can be redefined by users using the "-c" option.

For example #SBATCH –n 2 is requesting 2 cores, while #SBATCH –c 3 #SBATCH –n 2 is requesting 6 cores.

On Dalma/SLURM we implement an exclusive policy on nodes being used to run parallel jobs - eg no other jobs may run on nodes allocated for running parallel jobs.

When submitting parallel jobs on Dalma you need not specify the number of nodes. The number of tasks and cpus-per-task is sufficient for SLURM to determine how many nodes to reserve.

SLURM: node list

Sometimes applications require a list of nodes where they are to run in parallel to start.

SLURM keeps the list of nodes within the environment variable "$SLURM_JOB_NODELIST".

To retrieve the list of nodes in a PBS format file use "generate_pbs_nodefile".

For example:

```
export NODEFILE=$(generate_pbs_nodefile)
```

Then $NODEFILE contains the name of a temporary file containing the list of nodes used following the PBS hostfile format.

SLURM: accounts

SLURM maintains user associations which include user, account, qos, and partition. Users may have several associations. Moreover, accounts are hierarchical. For example, account "physics" maybe be a sub-account of "faculty", which may be a sub-account of "institute", etc.

When submitting jobs users with multiple associations must explicitely list the account, qos, partition details they wish to use.

```
sbatch –p serial –a physics –q normal –u benoit  job
```

Dalma specific job submission tools extend SLURM's associations to define a "default" association. So you only need to specify accounts is, for example, you belong to multiple accounts – ex faculty and research-lab – and you want to execute using your non-default account. So at most you'll need to specify:

```
sbatch –p <partition> -a <account> job
```

Moreover, accounts, partitions, qos and users may each be configured with resource usage limits. Thus the administrators can impose limits to the number of jobs queued, jobs running, cores usage, and run time.

جامعة نيويورك أبوظبي
NYU | ABU DHABI

SLURM: account limits

To see you SLURM associations (and their parents) as well as your resource usage limits use the following Dalma specific tool:

```
> slurm-show-my-limits.sh
------- --------- ------------ ------------ --------- ------- --------- -------- ------- --------- --------- ---------
   User   Account    Partition          QOS GrpSubmit GrpJobs   GrpTRES  MaxTRES MaxJobs MaxSubmit   MaxWall  Par Name
------- --------- ------------ ------------ --------- ------- --------- -------- ------- --------- --------- ---------
 benoit  qaad_par      par_std      par_std                     cpu=700  cpu=700     100       200  12:00:00
 benoit  qaad_ser       visual       visual                                         100       200  12:00:00
 benoit  qaad_ser      butinah      butinah                                         100       200  12:00:00
 benoit  qaad_ser       bigmem       bigmem                      cpu=10             100       200  12:00:00
 benoit  qaad_ser  preempt_std  preempt_std                                         100       200  12:00:00
 benoit  qaad_ser      ser_std      ser_std                      cpu=70             100       200  12:00:00
         qaad_par                    normal                     cpu=112             100       200  12:00:00      qaad
             qaad                    normal                     cpu=252             100       200                nyuad
            nyuad                    normal     20000    1000  cpu=6608             100       200                 root
             root                    normal
         qaad_ser                    normal                     cpu=140             100       200  12:00:00      qaad
             qaad                    normal                     cpu=252             100       200                nyuad
            nyuad                    normal     20000    1000  cpu=6608             100       200                 root
             root                    normal
```

SLURM: account limits

```
> slurm-show-my-limits.sh
------- --------- ------------ --------- ------- --------- -------- ------- --------- --------- ---------
   User   Account    Partition GrpSubmit GrpJobs   GrpTRES  MaxTRES MaxJobs MaxSubmit   MaxWall  Par Name
------- --------- ------------ --------- ------- --------- -------- ------- --------- --------- ---------
 benoit  qaad_par      par_std                   cpu=700  cpu=350     100       200  12:00:00
 benoit  qaad_ser       visual                                         100       200  12:00:00
 benoit  qaad_ser      butinah                                         100       200  12:00:00
 benoit  qaad_ser       bigmem                    cpu=10               100       200  12:00:00
 benoit  qaad_ser  preempt_std                                         100       200  12:00:00
 benoit  qaad_ser      ser_std                    cpu=70                50        75   6:00:00
         qaad_par                                cpu=1000               100       200  12:00:00      qaad
             qaad                                 cpu=252               100       200                nyuad
            nyuad                20000    1000   cpu=6608               100       200                 root
             root
         qaad_ser                                 cpu=140               100       200  12:00:00      qaad
             qaad                                 cpu=252               100       200                nyuad
            nyuad                 20000    1000  cpu=6608               100       200                 root
             root
```

In this output we see:

- user "benoit" can submit up to 200 jobs on "par_std" (parallel) partition, but have at most 100 jobs running consuming a maximum of 700 cores total where each jobs is limited to a maximum of 350 cores for 12 hours
- user "benoit" can submit up to 75 jobs on "ser_std" (serial) partition, with at most 50 jobs running using a total of up to 70 cores for up to 6 hours
- account "qaad_par" is shared with other users and together they have a limit of 1000 cores, 200 jobs queued, and 100 jobs running (eg the sum of all cores used by running jobs using account "qaad_par" can't exceed 1000 cores)
- account "qaad_ser" is shared with other users and together they have a limit of 140 cores, 200 jobs queued, and 100 jobs running
- account "qaad" is a sub-account of "nyuad" and the sum of all parallel and serial jobs can't exceed 200 jobs queued, 100 jobs running and 252 cores

SLURM: account usage

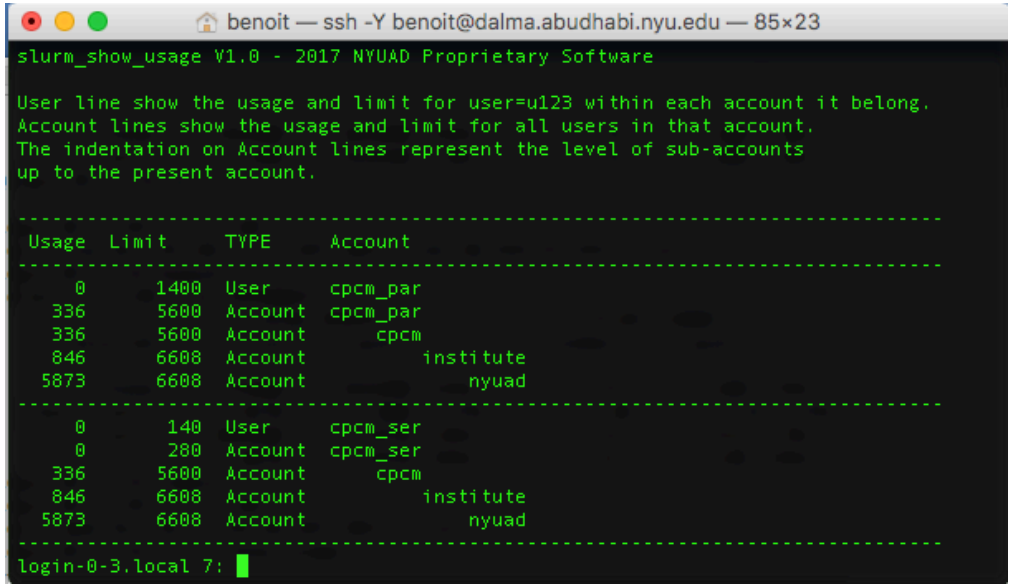
This next Dalma specific tool allows you to see how much resources you are using. This is useful when your job can't run because of "group resource limit" having been reached.

Here user "u123" has two accounts, "cpcm_par" and "cpcm_ser". On the "cpcm_par" (parallel partition) his limit is 1400 cores, and he's currently using 0 cores. However, other users from the same account are already using 336 cores out of the account maximum 5600.

The "cpcm_par" account is a sub-account of "cpcm", which currently is using 336 cores out of the 5600 permitted.

The "cpcm" account is also a sub-account of "institute". All "institute" users are presently using 846 cores out of the 6608 cores account limit.

Finally "institute" is a sub-account of "nyuad" where 5876 cores are being used - nearly at the 6608 limit.

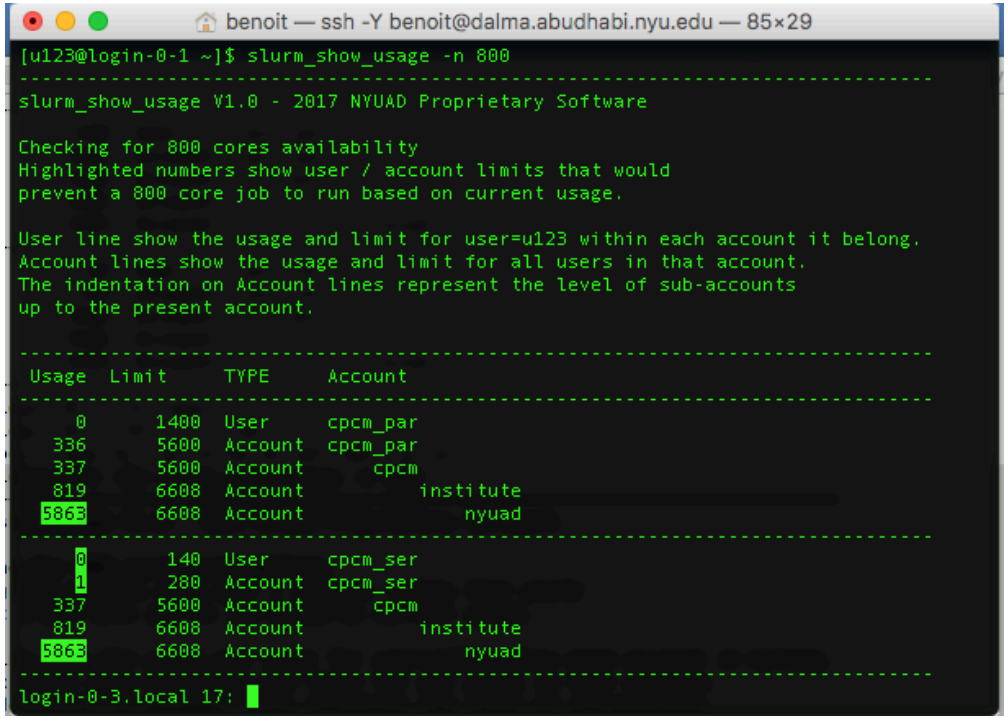


جامعة نيويورك أبوظبي
NYU | ABU DHABI

SLURM: account usage

The "slurm_show_usage" tool has an option to show you which account level would prevent you to run a job.

The "-n 800" option will show which account(s) would exceed the user or account core limit if you were to submit a job requiring 800 cores.



جامعة نيويورك أبوظبي
NYU | ABU DHABI

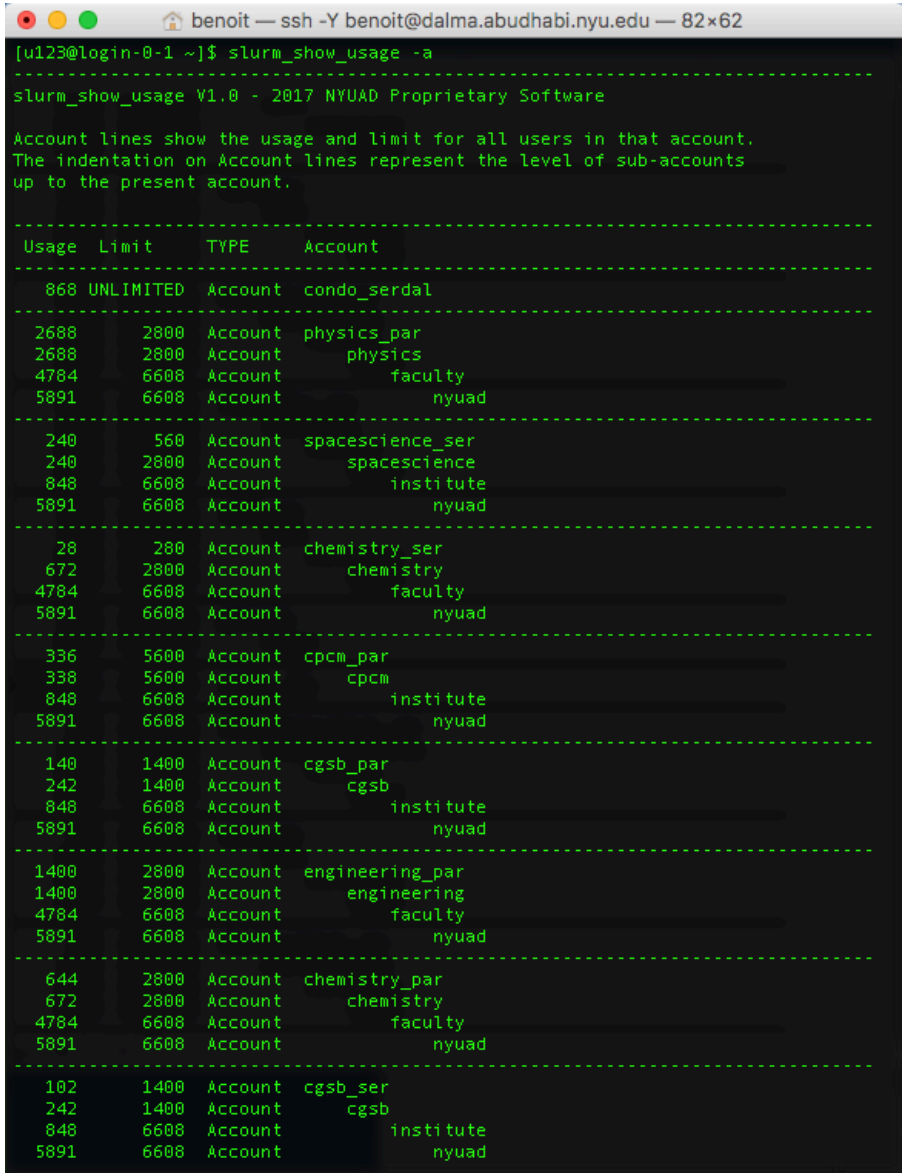
SLURM: account usage

The "-a" option will show all accounts usage and limit on Dalma, as well as their current usage.

The usage limits are defined by the academic steering committee in order to meet each group's computational needs, while allowing fairness to all groups.

The account limits are periodically revised based on prior usage statistics and inputs from the research groups about new project requirements.

Thus, the HPC support team role is limited to implementing the recommendations from the steering committee and to provide the steering committee with statistics and other key informations that help them define fair resource usage rules.



جامعة نيويورك أبوظبي
NYU | ABU DHABI

SLURM: system usage

The "dmap" tool (Dalma specific) will show you the utilization of each compute node on the cluster. The first numbers is a shorthand for the compute node name, so "12-3" actually means "compute-12-3". The second numbers represent the number of cores used and total number of cores in the system.

"white" highlight shows nodes that are down for maintenance.

"green" means a node is busy.

No highlight means a node is free.