جامعة نيويورك أبوظبي

NYU | ABU DHABI

Getting Started with *Dalma*

High Performance Computing NYUAD

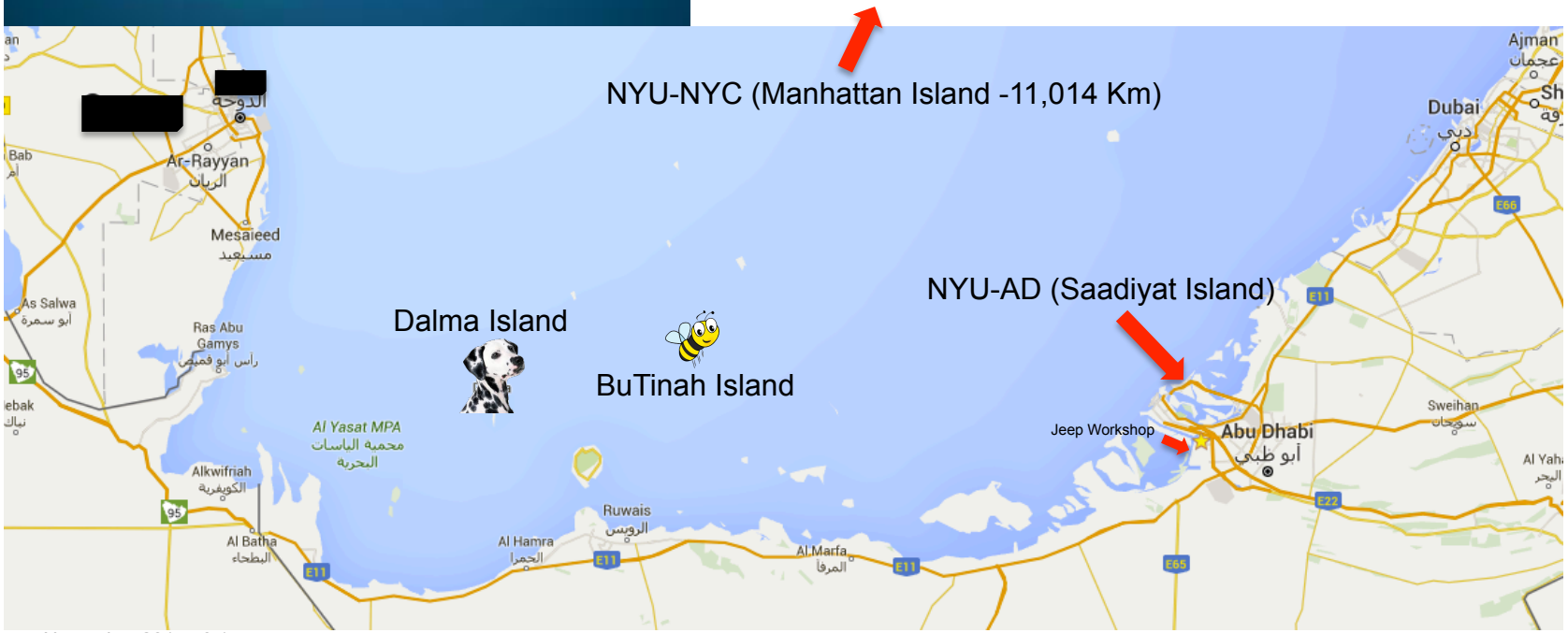
November 2017 v6.1

Dalma

Dalma is a guitar shaped island off the U.A.E. coast near.

Using BuTinah (previous HPC system) NYUAD researchers discovered evidences of agriculture (date palm) being practiced on Dalma some 7000 years ago.

Dalma is a super-computer owned and operated by NYUAD's Research Institute.

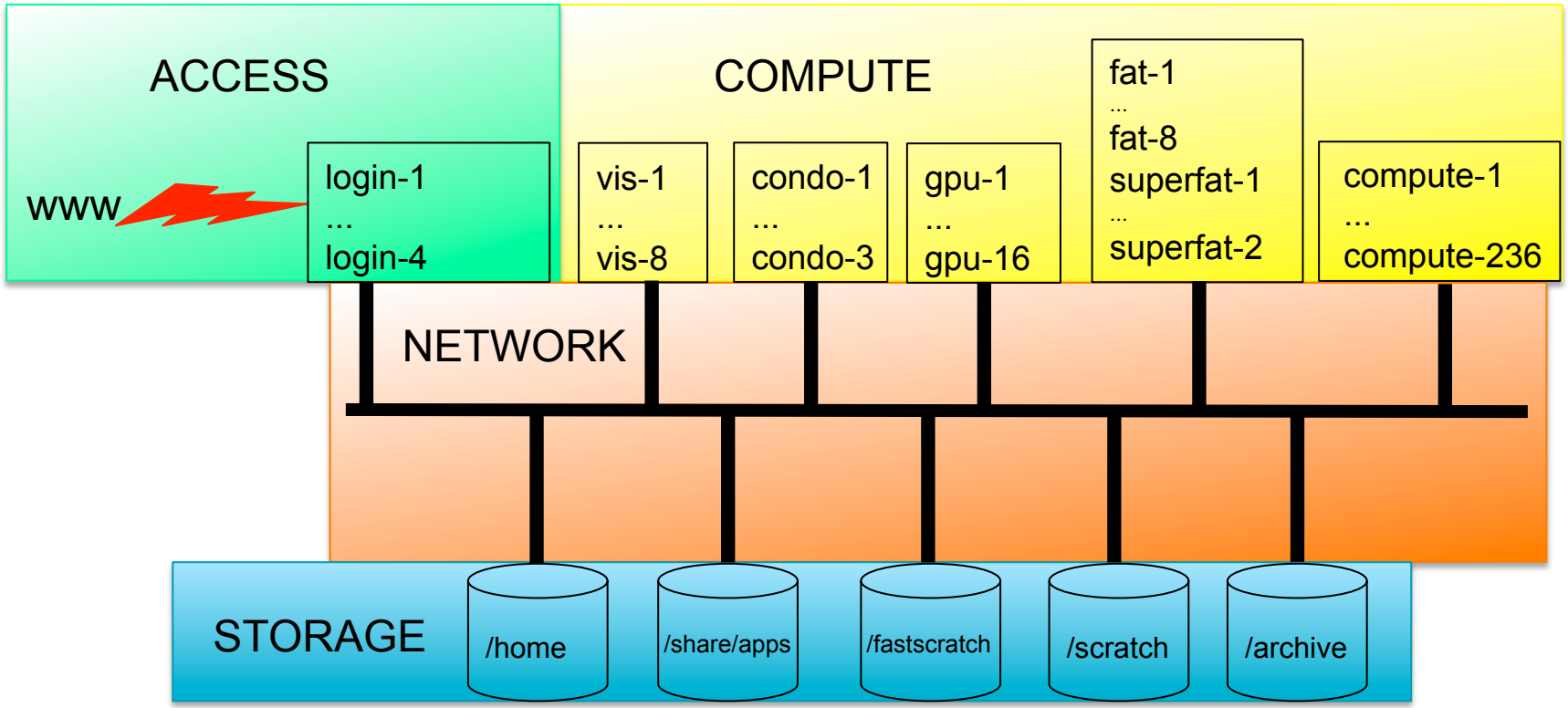


Why This "Intro" to Dalma?

- HPC systems are very different from desktop computers
- There are 5 components where they differ the most:
    - *Disk quota*
    - *Job submission*
    - *Processing quota*
    - *Software modules*
    - *Archival System*
- In this introduction we will review these components
- Note that these components vary from one HPC system to another
- The IT owned and managed HPC system in NYC – Prince - differs from Dalma
- Moving between Dalma and Prince requires adjustments to your job scripts

Dalma Block Diagram

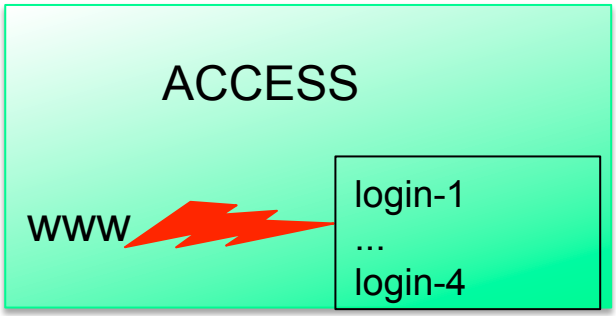
These are Dalma's hardware components. We are using a classic cluster architecture which is ideal for varied workloads. Each node is a Symmetric Multi Processor (SMP) system connected to the other nodes, and the storage system, through a 100Gbps Infiniband network.



Dalma Block Diagram

If connected to NYU's NYC or AD intranet you access Dalma using:

```
ssh <netid>@dalma.abudhabi.nyu.edu
```



If going through Internet you connect via "hpc":

```
ssh <netid>@hpc.abudhabi.nyu.edu –p 4410
```

then from the "hpc" node:

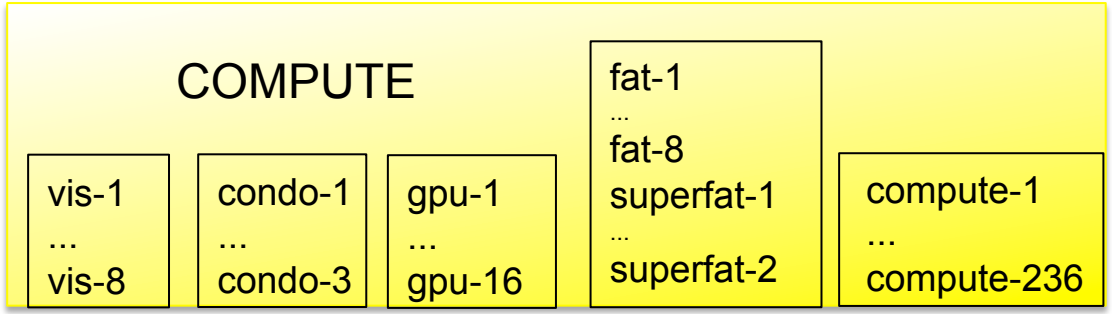
```
ssh <netid>@dalma.abudhabi.nyu.edu
```

In both cases you will start a shell session on a randomly selected login node. The login nodes arewhere you prepare your jobs to be submitted to the system, and where you manage your data. Login nodes have access to Internet, so you can use "ftp", "git", "gem", "wget", "pip", etc. to access remote data and software repositories.

Dalma Block Diagram

The computing nodes is where you applications run. You can't access them directly. You interact with them through a software tool called a "workload management utility". There are various workload managers out there, NYUAD uses "SLURM". We cover that subject later.

Dalma hosts a few types of compute nodes: visualization, GPU, large memory (fat), privately owned (condo), and general purpose (compute). Together they provide over 213TFlops of compute power using more than 7'000 processor cores.

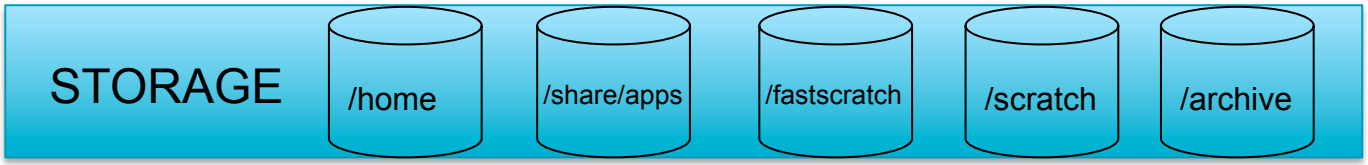


Node Type	CPU Type	#CPU Cores	Memory
gpu	sse	12@2.7GHz	96GB
fat	sse	12@2.7GHz	192GB
superfat	sse/avx2	32@2.7GHz/ 72@2.4GHz	1TB / 2TB
compute	avx2	28@2.4GHz	128GB

Dalma Block Diagram

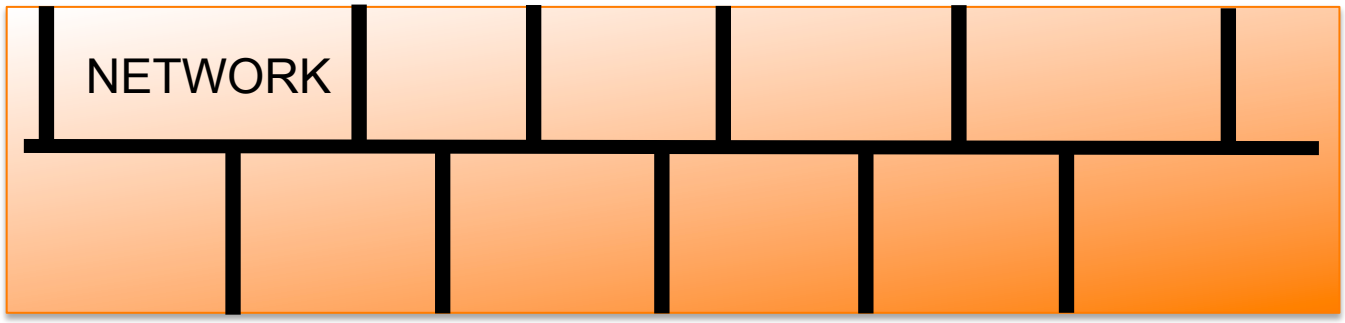
There are 5 types of storage on Dalma, each serves a specific purpose.
(/share/apps not mentioned below is where application software is stored)

Mount Point	/home	/archive	/fastscratch	/scratch
Type	BeeGFS	DMF + Tape	BeeGFS	Lustre
Default Quota	5GB 100K files	2TB 500K files	2TB 500K files	5TB 500K files
Capacity	4.3 TB	750 TB	216 TB	965 TB
Deletion Policy	None	None	Files not accessed for 90 days	Files not accessed for 90 days
Intended Use	bashrc, ssh files, Perl, Python, Java modules	permanently storing data you no longer use (or unlikely to access soon)	data mining applications, segmented I/O, access is granted for specific applications only	storage for your data, location from where you submit jobs



Dalma Block Diagram

The network is 100Gbps EDR Infiniband - non-blocking topology. It transports data to / from disks, as well as MPI traffic (distributed applications). It can sustain data traffic reaching up to 1.4TB/s!



Storage: Quota

When storage is shared among many users rules must be set to prevent users to consume disk space excessively at the expense of others. For this purpose HPC systems enforce disk quotas to users and groups of users.

On Dalma disk quotas are enforced on /home and /scratch (not /archive or /fastscratch). There are two quota constraints: the total amount of disk space and the total number of files. Once you reach a quota limit your jobs may be killed. So it is a good practice to check your quota before submitting a job that will generate a lot of data.

myquota (Dalma specific) will show your storage utilisation and your limits.

```
benoit — ssh -Y benoit@dalma.abudhabi.nyu.edu — 88×11
[login-0-3.local 1: myquota
Disk quotas for benoit (uid 1028):
                                DISK SPACE                  # FILES (1000's)
          filesystem        size      quota              number      quota
                      -------------------------     -------------------------
               /home       13GB      107GB (12%)            3         500 ( 1%)
        /fastscratch        0KB        0KB (-nan%)           0           0 (-nan%)
            /scratch      309GB     5261GB ( 6%)          319        2000 (16%)
            /archive      419GB     5261GB ( 8%)           11         500 ( 2%)
```

Storage: Policies

The following data policies are enforced on Dalma:

1. Backup: there is no backup performed by our staff, each user is responsible to archive his/her data.

2. Files count: no user is allowed to have more than 1'000 files per directory on /scratch, /home, and /fastscratch, and 500 on /archive. We periodically inspect storage devices and block user accounts if needed.

3. Files which have not been used for more than 90 days on /scratch and /fastscratch may be deleted. Users will be notified and given a chance to archive their data if they wish.

Storage: Archiving

Backup VS Archive

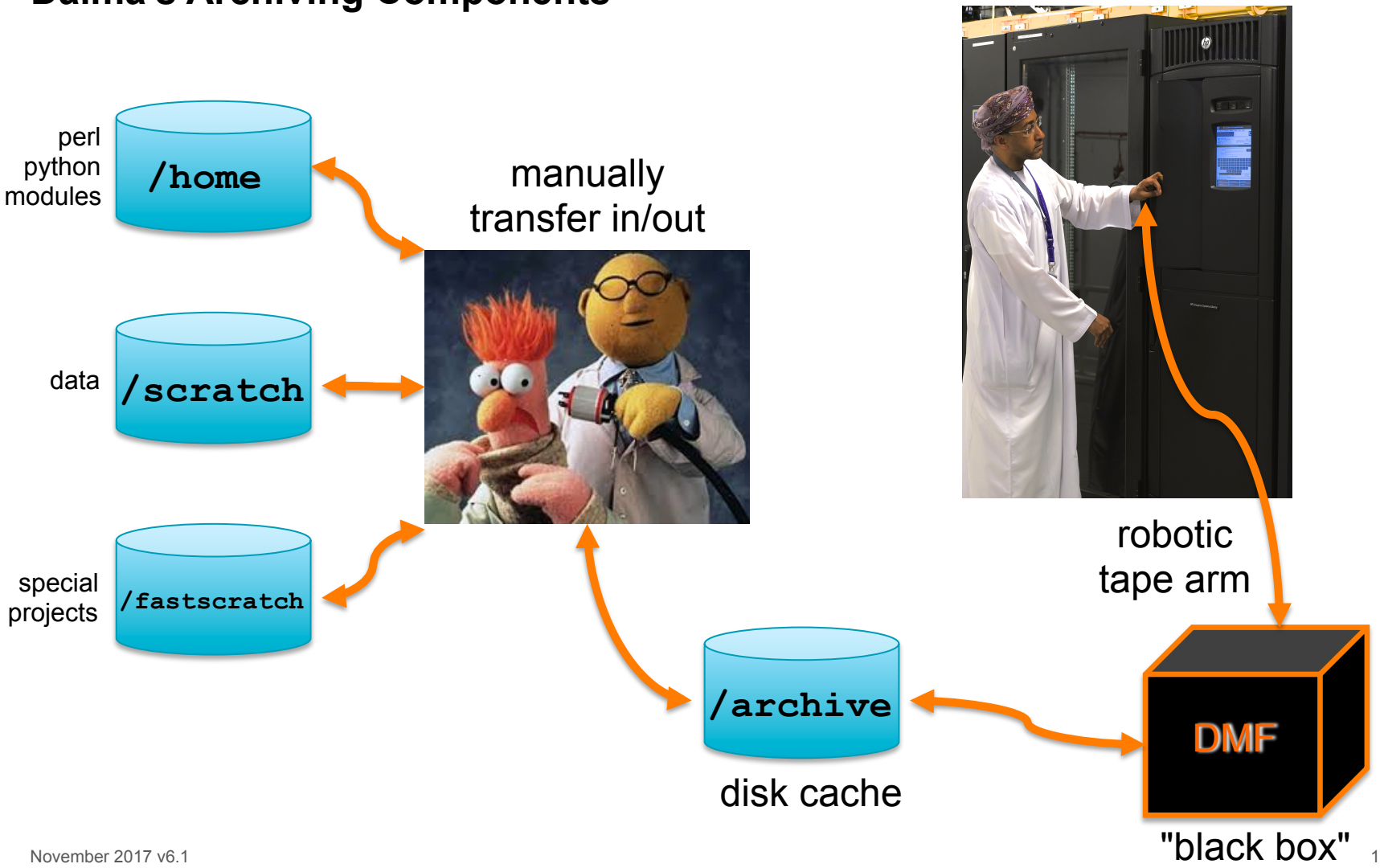
- A backup is an operation where a disk's contents is automatically copied to tape
- Archiving is an operation where a disk's contents is manualy moved to a long-term storage device

archive: manually **MOVE** data to long-term storage (tape or other)

backup: automatically & periodically **COPY** data to tape

Storage: Archiving

Dalma's Archiving Components



Storage: Archiving

What `/archive` does

- DMF (**D**ata **M**igration **F**acility) is an automatic data archiving tool
- It buffers data to a disk cache before copying it (eg migration) to the tape drive
- Old data on disk cache is eventually removed to make room for more data
- Once removed from disk cache files still appear in cache – file hierarchy remains, but not data
- Accessing such files automatically triggers their migration back to the disk cache
- Migration to tape drive actually makes two tape copies
- One copy is periodically sent to an outside data recovery site
- The tapes have a data retention of 30+ years
- Each time you modify a file on the disk cache two new copies of the latest version are made to tape
- There are no limits on how many file versions that can be made
- There is no disk quota for the amount of space used on tape or disk cache

Storage: Archiving

DO NOT MISUSE `/archive`

- Don't archive directories with more than 500 files
- Don't archive files that aren't meaninful
- Don't move files in and out
- Don't use it to backup your work weekly / monthly
- Don't request a file restore unless absolutely necessary
- Don't use /archive as a way around your /scratch disk quota

Storage: Archiving

CORRECT DISK USAGE `/archive`

- Periodically clean your /scratch & /fastscratch space
- Once a project is completed move the data over to /archive
- Use tar files to archive directories with large file count
- Use "git" or similar versioning tool to keep track of changes in source code
- Remove any data from /archive which you are sure will not be used ever again

Storage: Archiving

List /archive files

- dmfls
- Same options to "ls"
- "-l" option display state of file

REG	File not managed by DMF
MIG	Migrating
ARC	Archiving
DUL	Dual-state (both on tape and disk cache)
OFL	Offline
UNM	Unmigrating
NMG	Nonmigratable file
PAR	Partial-state file
N/A	DMF can't determine state
INV	Invalid state

```
> dmfls -l /archive/zidane/
total 16
drwxrwxr-x  112 apps    apps        4096 2015-04-30 08:25 (REG) builds
-rw-rw-r--    1 apps    apps      693985 2015-06-22 12:49 (OFL) eigen-intel14.tar.gz
-rw-rw-r--    1 apps    apps    10546302 2015-06-22 12:46 (OFL) eigensoft-intel14.tar.gz
-rw-rw-r--    1 apps    apps     2438932 2015-06-22 12:40 (OFL) fftw2-intel14.tar.gz
-rw-rw-r--    1 apps    apps     7921210 2015-06-22 12:40 (OFL) fftw3-intel14.tar.gz
-rw-rw-r--    1 apps    apps     2730147 2015-06-22 12:43 (OFL) gsl-intel14.tar.gz
-rw-rw-r--    1 apps    apps    25386040 2015-06-22 12:42 (OFL) hdf5-intel14.tar.gz
drwxrwxr-x    3 apps    apps          26 2015-02-08 08:39 (REG) jags
-rw-rw-r--    1 apps    apps      223502 2015-06-22 12:46 (OFL) mpiblacs-intel14.tar.gz
-rw-rw-r--    1 apps    apps    10244400 2015-06-22 12:44 (OFL) mvapich2-intel14
-rw-rw-r--    1 apps    apps     7083606 2015-06-22 12:45 (OFL) netcdf-c-intel14.tar.gz
-rw-rw-r--    1 apps    apps    33848579 2015-06-22 12:44 (OFL) openmpi-intel14.tar.gz
-rw-rw-r--    1 apps    apps    10295250 2015-06-22 12:48 (OFL) scalapack-intel14.tar.gz
drwxrwxr-x   67 apps    apps        4096 2016-01-21 09:08 (REG) scripts
drwxrwxr-x    2 apps    apps        4096 2016-01-21 09:02 (REG) tars
drwxrwxr-x    3 apps    apps          24 2015-02-05 10:16 (REG) toolshed
-rw-rw-r--    1 apps    apps      147937 2015-06-22 12:43 (OFL) zlib-intel14.tar.gz
```

NOTE: all paths to archive must be absolute (eg /archive/...)

Storage: Archiving

Remove /archive files

- dmfrm
- Similar options to "rm"

Remove a file:

```
> dmfrm /archive/beckham/myfile
```

Remove a directory:

```
> dmfrm -r /archive/beckham/mydir
```

NOTE: deleted files can be recovered within two weeks

Storage: Archiving

Show /archive disk use

- dmfdu
- Similar options to "du"

blocks used by a directory / file:

```
> dmfdu /archive/ronaldinho
27567617  /archive/ronaldinho/WIEN2K
27567617  /archive/ronaldinho
```

Human readable (-h) and grand total (-c):

```
> dmfdu -ch  /archive/ronaldinho
27G /archive/ronaldinho/WIEN2K
27G /archive/donaldinho
27G total
```

Storage: Archiving

Transfer files to /archive

- dmfput
- Copy data to /archive
- Usage: dmfput <source> <destination>
- Audit of directories before archiving
- If destination directory exists then only synchronize with new source content (rsync)

```
> dmfput /scratch/neymar/mydir /archive/neymar

> dmfput /scratch/neymar/myfile /archive/neymar/dir1

> dmfput /scratch/neymar/dir1/myfile /archive/neymar/dir1/myfile

> dmfput /scratch/neymar/mycrazydir /archive/neymar
Auditing /scratch/neymar/mycrazydir
Directory /scratch/neymar/mycrazydir has too many entries (1000 max=500)
Archiving terminated on account of directories failing to meet limits
```

Storage: Archiving

Retrieve files from /archive

- dmfget
- Copy data from /archive
- Usage: dmfget <source> <destination>

```
> dmfget /archive/ibrahimovic/dir1/fileX /scratch/ibrahimovic/dir2
```

Storage: Archiving

Archiving directories with large number of files

- Very wasteful of resources
- Very very slow to scan (eg "ls", "find", etc)
- **DO NOT ATTEMPT TO ARCHIVE SUCH DIRECTORIES**
- Create "compressed tar files" of them instead and archive those tar files

```
> tar -czf mycrazydir.tgz mycrazydir

> rm -fr mycrazydir

> dmfput /scratch/messi/mycrazydir.tgz /archive/messi
```

Storage: Archiving

Archiving several versions of the same directory

- Write the data into individual directories

```
> dmfput /scratch/ronaldo/mydir /archive/ronaldo/mydir.20170822
```

some *time later...*

```
> dmfput /scratch/ronaldo/mydir /archive/ronaldo/mydir.20170905
```

SLURM: Workload Manager

Simple **L**inux **U**tility for **R**esource **M**anagement

Lawrence Livermore National Laboratory

- All HPC systems use a workload management tool to distribute jobs to compute nodes
- Users submit a job (shell script) to the tool which then waits for the first available node(s)
- The aim is to keep the system as busy as possible while ensuring fairness of use
- SLURM is open-source software developped by Lawrence Livermore National Laboratory
- Industry Consortium pursues development (HPE, Atos, SchedMD)
- Designed for Linux clusters of all sizes
- Used by 60% of TOP500 Super Computers

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

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

job1.sh

```
> sbatch job1.sh
```

OR

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

job2.sh

```
> 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.

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.

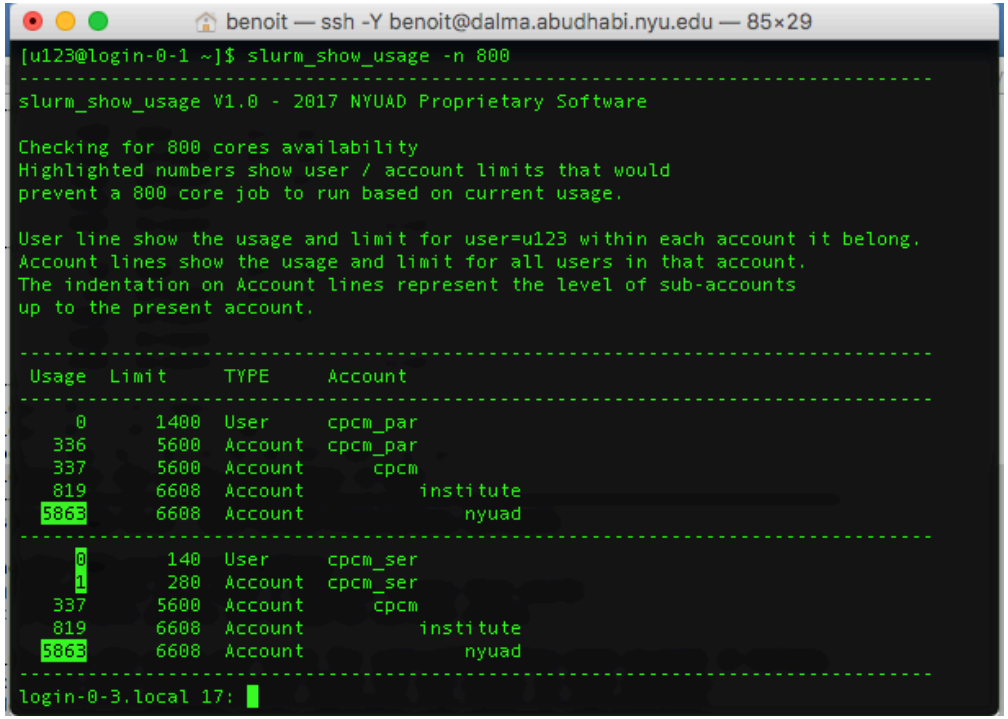
Finally "institute" is a sub-account of "nyuad" where 5876 cores are being used - nearly at the 6608 limit.

```
benoit — ssh -Y benoit@dalma.abudhabi.nyu.edu — 85×23
slurm_show_usage V1.0 - 2017 NYUAD Proprietary Software

User line show the usage and limit for user=u123 within each account it belong.
Account lines show the usage and limit for all users in that account.
The indentation on Account lines represent the level of sub-accounts
up to the present account.

-------------------------------------------------------------------------------
 Usage  Limit     TYPE     Account
-------------------------------------------------------------------------------
     0      1400  User     cpcm_par
   336      5600  Account  cpcm_par
   336      5600  Account      cpcm
   846      6608  Account          institute
  5873      6608  Account              nyuad
-------------------------------------------------------------------------------
     0       140  User     cpcm_ser
     0       280  Account  cpcm_ser
   336      5600  Account      cpcm
   846      6608  Account          institute
  5873      6608  Account              nyuad
-------------------------------------------------------------------------------
login-0-3.local 7:
```

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.



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.

benoit — ssh -Y benoit@dalma.abudhabi.nyu.edu — 82×62

```
[u123@login-0-1 ~]$ slurm_show_usage -a
--------------------------------------------------------------------------------
slurm_show_usage V1.0 - 2017 NYUAD Proprietary Software

Account lines show the usage and limit for all users in that account.
The indentation on Account lines represent the level of sub-accounts
up to the present account.

--------------------------------------------------------------------------------
 Usage  Limit     TYPE     Account
--------------------------------------------------------------------------------
   868 UNLIMITED  Account  condo_serdal
--------------------------------------------------------------------------------
  2688      2800  Account  physics_par
  2688      2800  Account      physics
  4784      6608  Account          faculty
  5891      6608  Account              nyuad
--------------------------------------------------------------------------------
   240       560  Account  spacescience_ser
   240      2800  Account      spacescience
   848      6608  Account          institute
  5891      6608  Account              nyuad
--------------------------------------------------------------------------------
    28       280  Account  chemistry_ser
   672      2800  Account      chemistry
  4784      6608  Account          faculty
  5891      6608  Account              nyuad
--------------------------------------------------------------------------------
   336      5600  Account  cpcm_par
   338      5600  Account      cpcm
   848      6608  Account          institute
  5891      6608  Account              nyuad
--------------------------------------------------------------------------------
   140      1400  Account  cgsb_par
   242      1400  Account      cgsb
   848      6608  Account          institute
  5891      6608  Account              nyuad
--------------------------------------------------------------------------------
  1400      2800  Account  engineering_par
  1400      2800  Account      engineering
  4784      6608  Account          faculty
  5891      6608  Account              nyuad
--------------------------------------------------------------------------------
   644      2800  Account  chemistry_par
   672      2800  Account      chemistry
  4784      6608  Account          faculty
  5891      6608  Account              nyuad
--------------------------------------------------------------------------------
   102      1400  Account  cgsb_ser
   242      1400  Account      cgsb
   848      6608  Account          institute
  5891      6608  Account              nyuad
```

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.

```
benoit — ssh -Y benoit@dalma.abudhabi.nyu.edu — 156×53
*************************************************************************
'dmap' showing:
                cpus alloc, for all users, with all jobs
*************************************************************************
[1-1  : 28/28] [1-2  : 28/28] [1-3  : 01/28] [1-4  : 00/28] [1-5  : 28/28] [1-6  : 28/28] [1-7  : 28/28] [1-8  : 28/28] [1-9  : 28/28]
[1-10 : 28/28] [1-11 : 28/28] [1-12 : 28/28] [1-13 : 28/28] [1-14 : 28/28] [1-15 : 28/28] [1-16 : 28/28] [1-17 : 28/28] [1-18 : 28/28]
[2-1  : 28/28] [2-2  : 28/28] [2-3  : 28/28] [2-4  : 28/28] [2-5  : 28/28] [2-6  : 28/28] [2-7  : 28/28] [2-8  : 28/28] [2-9  : 28/28]
[2-10 : 28/28] [2-11 : 28/28] [2-12 : 28/28] [2-13 : 28/28] [2-14 : 28/28] [2-15 : 00/28] [2-16 : 00/28] [2-17 : 28/28] [2-18 : 28/28]
[3-1  : 28/28] [3-2  : 28/28] [3-3  : 28/28] [3-4  : 28/28] [3-5  : 28/28] [3-6  : 28/28] [3-7  : 28/28] [3-8  : 28/28] [3-9  : 28/28]
[3-10 : 28/28] [3-11 : 28/28] [3-12 : 28/28] [3-13 : 28/28] [3-14 : 28/28] [3-15 : 28/28] [3-16 : 28/28] [3-17 : 28/28] [3-18 : 28/28]
[4-1  : 28/28] [4-2  : 28/28] [4-3  : 28/28] [4-4  : 28/28] [4-5  : 28/28] [4-6  : 28/28] [4-7  : 28/28] [4-8  : 28/28] [4-9  : 28/28]
[4-10 : 00/28] [4-11 : 28/28] [4-12 : 28/28] [4-13 : 28/28] [4-14 : 28/28] [4-15 : 28/28] [4-16 : 28/28] [4-17 : 28/28] [4-18 : 28/28]
[5-1  : 28/28] [5-2  : 28/28] [5-3  : 28/28] [5-4  : 28/28] [5-5  : 28/28] [5-6  : 28/28] [5-7  : 28/28] [5-8  : 28/28] [5-9  : 28/28]
[5-10 : 28/28] [5-11 : 28/28] [5-12 : 28/28] [5-13 : 28/28] [5-14 : 28/28] [5-15 : 28/28] [5-16 : 28/28] [5-17 : 28/28] [5-18 : 28/28]
[6-1  : 28/28] [6-2  : 28/28] [6-3  : 28/28] [6-4  : 28/28] [6-5  : 28/28] [6-6  : 28/28] [6-7  : 28/28] [6-8  : 28/28] [6-9  : 28/28]
[6-10 : 28/28] [6-11 : 28/28] [6-12 : 28/28] [6-13 : 28/28] [6-14 : 28/28] [6-15 : 28/28] [6-16 : 28/28] [6-17 : 28/28] [6-18 : 28/28]
[7-1  : 28/28] [7-2  : 28/28] [7-3  : 28/28] [7-4  : 28/28] [7-5  : 28/28] [7-6  : 28/28] [7-7  : 28/28] [7-8  : 28/28] [7-9  : 28/28]
[7-10 : 28/28] [7-11 : 28/28] [7-12 : 28/28] [7-13 : 28/28] [7-14 : 28/28] [7-15 : 28/28] [7-16 : 28/28] [7-17 : 28/28] [7-18 : 28/28]
[8-1  : 00/28] [8-2  : 28/28] [8-3  : 28/28] [8-4  : 28/28] [8-5  : 28/28] [8-6  : 00/28] [8-7  : 00/28] [8-8  : 00/28] [8-9  : 28/28]
[8-10 : 28/28] [8-11 : 28/28] [8-12 : 28/28] [8-13 : 00/28] [8-14 : 00/28] [8-15 : 00/28] [8-16 : 28/28] [8-17 : 00/28] [8-18 : 00/28]
[9-1  : 28/28] [9-2  : 28/28] [9-3  : 28/28] [9-4  : 28/28] [9-5  : 26/28] [9-6  : 28/28] [9-7  : 28/28] [9-8  : 00/28] [9-9  : 28/28]
[9-10 : 28/28] [9-11 : 28/28] [9-12 : 28/28] [9-13 : 28/28] [9-14 : 28/28] [9-15 : 28/28] [9-16 : 28/28] [9-17 : 24/28] [9-18 : 28/28]
[10-1 : 28/28] [10-2 : 28/28] [10-3 : 28/28] [10-4 : 28/28] [10-5 : 28/28] [10-6 : 28/28] [10-7 : 28/28] [10-8 : 28/28] [10-9 : 28/28]
[10-10: 28/28] [10-11: 28/28] [10-12: 28/28] [10-13: 28/28] [10-14: 28/28] [10-15: 28/28] [10-16: 28/28] [10-17: 28/28] [10-18: 28/28]
[11-1 : 00/28] [11-2 : 00/28] [11-3 : 00/28] [11-4 : 00/28] [11-5 : 00/28] [11-6 : 00/28] [11-7 : 00/28] [11-8 : 00/28] [11-9 : 26/28]
[11-10: 00/28] [11-11: 00/28] [11-12: 00/28] [11-13: 00/28] [11-14: 22/28] [11-15: 00/28] [11-16: 00/28] [11-17: 00/28] [11-18: 00/28]
[12-1 : 00/28] [12-2 : 00/28] [12-3 : 28/28] [12-4 : 28/28] [12-5 : 00/28] [12-6 : 00/28] [12-7 : 00/28] [12-8 : 00/28] [12-9 : 00/28]
[12-10: 28/28] [12-11: 28/28] [12-12: 28/28] [12-13: 28/28] [12-14: 00/28] [12-15: 28/28] [12-16: 28/28] [12-17: 00/28] [12-18: 28/28]
[13-1 : 28/28] [13-2 : 28/28] [13-3 : 28/28] [13-4 : 28/28] [13-5 : 28/28] [13-6 : 28/28] [13-7 : 28/28] [13-8 : 28/28] [13-9 : 28/28]
[13-10: 28/28] [13-11: 28/28] [13-12: 28/28] [13-13: 28/28] [13-14: 00/28] [13-15: 00/28] [13-16: 00/28] [13-17: 28/28] [13-18: 28/28]
[14-1 : 00/28] [14-2 : 28/28] [14-3 : 28/28] [14-4 : 28/28] [14-5 : 28/28] [14-6 : 28/28] [14-7 : 28/28] [14-8 : 00/28] [14-9 : 28/28]
[14-10: 28/28] [14-11: 28/28] [14-12: 28/28] [14-13: 28/28] [14-14: 28/28] [14-15: 28/28] [14-16: 28/28] [14-17: 28/28] [14-18: 28/28]
[15-5 : 24/28] [15-6 : 24/28] [15-7 : 24/28] [15-8 : 24/28] [15-9 : 24/28] [15-10: 24/28] [15-11: 24/28] [15-12: 22/28] [15-13: 28/28]
[15-14: 28/28] [15-15: 28/28] [15-16: 28/28] [15-17: 24/28] [15-18: 21/28] [16-1 : 28/28] [16-2 : 28/28] [16-3 : 28/28] [16-4 : 28/28]
[16-5 : 28/28] [16-6 : 28/28] [16-7 : 28/28] [16-8 : 28/28] [16-9 : 24/28] [16-10: 28/28] [16-11: 28/28] [16-12: 28/28] [16-13: 28/28]
[16-14: 28/28] [21-1 : 12/12] [21-2 : 12/12] [21-3 : 12/12] [21-4 : 12/12] [21-5 : 12/12] [21-6 : 12/12] [21-7 : 12/12] [21-8 : 12/12]
[21-9 : 11/12] [21-10: 12/12] [21-11: 12/12] [21-12: 12/12] [21-13: 12/12] [21-14: 12/12] [21-15: 12/12] [21-16: 12/12] [22-1 : 12/12]
[22-2 : 12/12] [22-3 : 12/12] [22-4 : 12/12] [22-5 : 12/12] [22-6 : 12/12] [22-7 : 12/12] [22-8 : 12/12] [22-9 : 00/32] [22-10: 00/72]
[23-3 : 24/24] [23-4 : 22/24] [     : 00/6] [     : 00/6] [     : 00/6] [     : 00/6] [     : 00/6] [     : 00/6] [     : 00/6]
[     : 00/6]
 Total Cpus: 8280
 Alloc Cpus: 6851
CPU Utilization: 82.7%
```

SLURM: Interactive Job

- You can run interactively on either serial or parallel partitions
- For parallel partition the shell will be run on first allocated node
- The same limitations for resource limits apply as for non interactive jobs
- If the system is full you will wait...

```
$> srun --pty -n 1 /bin/bash
```

- Start an interactive bash on a compute node
- Node can be shared by someone elses job
- It will be allocated in the ‘serial’ partition

```
$> srun --pty -n 1 --exclusive  /bin/bash
```

- Start an interactive bash on a compute node
- Node is **NOT** shared with anyone
- It will be allocated in the ‘serial’ partition

```
$> srun --pty -N2 --exclusive  /bin/bash
```

- Start an interactive bash on 2 compute nodes
- Nodes are **NOT** available to anyone else
- It will be allocated in the ‘parallel’ partition

SLURM: Serial Job

```
#!/bin/bash

# Set maximum run time (default depends on partition)
# Here ask for 1 hour and 30 minutes
#SBATCH -t 1:30:00

# Set maximum memory (default depends on partition)
# Here asking for 2GB (default unit is MB, can use GB suffix too)
##SBATCH --mem=2000
#SBATCH --mem=2GB

# Ask for specific node type (sse or avx2)
# serial partition default "sse"
# parallel partition default "avx2"
#SBATCH -C avx2

# Ask SLURM to give you exclusive access to node
# You do this when you need to publish a paper and
# must quote performance / time measurements
# default is shared access
##SBATCH --exclusive

# Number of tasks to run (default is 1)
#SBATCH -n 1

# Number of cores per task (default is 1)
#SBATCH -c 1

# Partition (default=serial)
#SBATCH -p serial

# load modules (as needed by the application)
module load atlas

./serial-application
```

This is what a typical serial job looks like.

The "#SBATCH" lines tell SLURM how many cores to use, which partition to use, etc. Meanwhile the "module load" lines (discussed later) prepare the software environment that is needed to run the "serial-application".

The "#!/bin/bash" indicates that this is a shell script. It is a good practice to test your shell script prior to submitting as many times there may be errors in your script. For this you start an interactive session through SLURM (as seen the previous slide) and simply execute the script manually.

On Dalma we strongly suggest that users submit jobs from their /scratch disk space.

When submitted through SLURM the script will be launched in the directory where the "sbatch" command was used.

By default serial jobs share nodes (eg multiple serial jobs run on the same node concurrently). This can be overriden if you require time measurements for publication purposes.

SLURM: Multi-Threaded Job

```
#!/bin/bash

# Number of tasks to run (default is 1)
#SBATCH -n 1

# Number of cores per task (default is 1)
#SBATCH -c 6

# Partition (default=serial)
#SBATCH -p serial

# load modules (as needed by the application)
#module load XXX

# Use SLURM cores-per-task to set OMP_NUM_THREADS
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

./multi-threaded-application
```

This is what a multi-threaded / OpenMP job looks like.

All parameters, modules, etc. are identical as for serial jobs. Except that you need to define how many cores a task requires. Note that multi-threaded jobs run on the serial partition, and that nodes can be shared with other serial or multi-threaded applications.

We strongly suggest that when submitting OpenMP jobs you set "OMP_NUM_THREADS" to the value of "SLURM_CPUS_PER_TASK" to avoid any discrepancy in your execution – people often forget to change one when they update the other.

SLURM: OpenMPI Job

Method 1 : OpenMPI

```
#!/bin/bash

# Number of MPI threads (default 1 core per MPI thread)
#SBATCH –n 56

# Use parallel partition
# Parallel partition nodes are exclusive
# Parallel partition nodes are only avx2 types
#SBATCH -p parallel

# No need to load the MPI module
# But may need to load other modules here

# Safety always!
module purge

srun ./mpi-application
```

Method 2 : OpenMPI

```
#!/bin/bash

#SBATCH –n 56
#SBATCH -p parallel
module purge
mpiexec ./mpi-application
```

OpenMPI Jobs can be run in 2 ways.

Again all parameters, modules, etc. are identical as for serial jobs. And you also need to define how many cores a task requires. OpenMPI jobs can only be executed on the parallel partition, where all nodes are "avx2" type. The parallel partition will not allow more than 1 job per node, so you run in exclusive mode always.

There's a tight integration between openmpi and SLURM which allows you to use "srun" to launch OpenMPI applications without the need to use "mpiexec" / "mpirun".

The second method is to load the OpenMPI module and to use "mpiexec" to launch the MPI threads on all nodes.

In both cases you don't need to specify the number of MPI threads, or to specify the list of nodes to use.

Moreover, using this method the current working directory and environment variables (and loaded modules) are automatically transmitted to all MPI threads.

SLURM: mvapich2 Job

Method 1 : mvapich2

```
#!/bin/bash

# Number of MPI threads (default 1 core per MPI thread)
#SBATCH –n 56

# Use parallel partition
# Parallel partition nodes are exclusive
# Parallel partition nodes are only avx2 types
#SBATCH -p parallel

# No need to load the MPI module
# But may need to load other modules here

# Safety always!
module purge

srun ./mpi-application
```

Method 2 : OpenMPI

```
#!/bin/bash

#SBATCH –n 56
#SBATCH -p parallel
module purge
mpiexec ./mpi-application
```

OpenMPI Jobs can be run in 2 ways.

Again all parameters, modules, etc. are identical as for serial jobs. And you also need to define how many cores a task requires. OpenMPI jobs can only be executed on the parallel partition, where all nodes are "avx2" type. The parallel partition will not allow more than 1 job per node, so you run in exclusive mode always.

There's a tight integration between openmpi and SLURM which allows you to use "srun" to launch OpenMPI applications without the need to use "mpiexec" / "mpirun". Then you don't need to specify the number of MPI threads, or to specify the list of nodes to use. Using this method the current working directory and environment variables (and loaded modules) are automatically transmitted to all MPI threads.

The second method is to load the OpenMPI module and to use "mpiexec" to launch the MPI threads on all nodes.

TO DO

SLURM: Hybrid MPI Job

Hybrid jobs (MPI+ OpenMP) can be started in 4 different ways.

You can use "srun" or "mpiexec" to launch your jobs, and then you can set the number of cores per task (-c) or the number of tasks-per-node to control the number of MPI threads per node / number of cores per MPI thread.

These methods work for both OpenMPI and MVAPICH2.

Method 1

```
#!/bin/bash
#SBATCH –n 2
#SBATCH -p parallel
#SBATCH –c 28

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
module purge
srun ./mpi-application
```

Method 2

```
#!/bin/bash
#SBATCH –n 2
#SBATCH -p parallel
#SBATCH --tasks-per-node=1

export OMP_NUM_THREADS=28
srun ./mpi-application
```

Method 3

```
#!/bin/bash
#SBATCH –n 2
#SBATCH -p parallel
#SBATCH –c 28

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
module purge
mpiexec./mpi-application
```

Method 4

```
#!/bin/bash
#SBATCH –n 2
#SBATCH -p parallel
#SBATCH --tasks-per-node=1

export OMP_NUM_THREADS=28
mpiexec ./mpi-application
```

SLURM: Job Array

```
#!/bin/bash
# Processing for each experiment requires a single core
# The input files are named file.1, file.2, ..., file.50
#SBATCH -p serial
#SBATCH -n 1
#SBATCH -a 1-50

module purge
./my-code file.${SLURM_ARRAY_TASK_ID}
```

Frequently we need to run the same application / job script several times for either:

- processing multiple independent input files
- looking for an optimal set of input parameters for the same input file (eg parametric analysis)

For instance you may need to process the results of 50 experiments. Then you can (1) prepare a single job script which processes all 50 experiments, one at a time, or (2) prepare 50 job scripts which process concurrently the experiments, one experiment per job.

In either case this is not an optimal use of your time.

The solution is to process your experiments using a job array; the script will be executed 50 times but each time it runs the counter will be different. You can use the counter to refer to each experiment input as in the job script to the left.

SLURM: Job Array

```
> sbatch job.sh
Submitted batch job 466546
> squeue -u bm102
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
     466546_[1-50]   ser_std   ja1.sh    bm102 PD       0:00      1 (Priority)
> squeue -u bm102
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          466546_1   ser_std   ja1.sh    bm102 CG       0:07      1 compute-21-2
          466546_2   ser_std   ja1.sh    bm102 CG       0:06      1 compute-21-11
          466546_3   ser_std   ja1.sh    bm102 CG       0:07      1 compute-21-11
          466546_4   ser_std   ja1.sh    bm102 CG       0:06      1 compute-21-11
          466546_5   ser_std   ja1.sh    bm102 CG       0:09      1 compute-21-13
          466546_6   ser_std   ja1.sh    bm102 CG       0:10      1 compute-21-13
          466546_7   ser_std   ja1.sh    bm102 CG       0:08      1 compute-21-13
          466546_8   ser_std   ja1.sh    bm102 CG       0:08      1 compute-21-13
          466546_9   ser_std   ja1.sh    bm102 CG       0:10      1 compute-21-13
         466546_10   ser_std   ja1.sh    bm102 CG       0:10      1 compute-21-13
         466546_11   ser_std   ja1.sh    bm102 CG       0:09      1 compute-21-13
         466546_12   ser_std   ja1.sh    bm102 CG       0:07      1 compute-21-13
         466546_13   ser_std   ja1.sh    bm102 CG       0:05      1 compute-21-13
         466546_14   ser_std   ja1.sh    bm102 CG       0:08      1 compute-21-13
         466546_15   ser_std   ja1.sh    bm102 CG       0:07      1 compute-22-3
         466546_16   ser_std   ja1.sh    bm102 CG       0:06      1 compute-22-3
         466546_17   ser_std   ja1.sh    bm102 CG       0:08      1 compute-22-3
         466546_18   ser_std   ja1.sh    bm102 CG       0:09      1 compute-22-3
         466546_19   ser_std   ja1.sh    bm102 CG       0:09      1 compute-22-3
         466546_20   ser_std   ja1.sh    bm102 CG       0:05      1 compute-22-3
         466546_21   ser_std   ja1.sh    bm102 CG       0:08      1 compute-22-3
         466546_22   ser_std   ja1.sh    bm102 CG       0:05      1 compute-1-4
         466546_23   ser_std   ja1.sh    bm102 CG       0:06      1 compute-1-4
         466546_24   ser_std   ja1.sh    bm102 CG       0:06      1 compute-1-4
         466546_25   ser_std   ja1.sh    bm102 CG       0:09      1 compute-1-4
         466546_26   ser_std   ja1.sh    bm102 CG       0:07      1 compute-1-4
         466546_27   ser_std   ja1.sh    bm102 CG       0:06      1 compute-1-4
         466546_28   ser_std   ja1.sh    bm102 CG       0:05      1 compute-8-1
         466546_29   ser_std   ja1.sh    bm102 CG       0:08      1 compute-8-1
         466546_30   ser_std   ja1.sh    bm102 CG       0:09      1 compute-8-1
         466546_31   ser_std   ja1.sh    bm102 CG       0:08      1 compute-8-1
         466546_32   ser_std   ja1.sh    bm102 CG       0:07      1 compute-8-1
         466546_33   ser_std   ja1.sh    bm102 CG       0:07      1 compute-8-1
         466546_34   ser_std   ja1.sh    bm102 CG       0:07      1 compute-9-15
         466546_35   ser_std   ja1.sh    bm102 CG       0:07      1 compute-9-15
         466546_36   ser_std   ja1.sh    bm102 CG       0:06      1 compute-9-15
         466546_37   ser_std   ja1.sh    bm102 CG       0:06      1 compute-9-15
         466546_38   ser_std   ja1.sh    bm102 CG       0:06      1 compute-9-15
         466546_39   ser_std   ja1.sh    bm102 CG       0:08      1 compute-9-15
         466546_40   ser_std   ja1.sh    bm102 CG       0:09      1 compute-9-15
         466546_41   ser_std   ja1.sh    bm102 CG       0:07      1 compute-9-15
         466546_42   ser_std   ja1.sh    bm102 CG       0:08      1 compute-11-13
         466546_43   ser_std   ja1.sh    bm102 CG       0:09      1 compute-11-13
         466546_44   ser_std   ja1.sh    bm102 CG       0:10      1 compute-11-13
         466546_45   ser_std   ja1.sh    bm102 CG       0:09      1 compute-11-13
         466546_46   ser_std   ja1.sh    bm102 CG       0:08      1 compute-11-13
         466546_47   ser_std   ja1.sh    bm102 CG       0:09      1 compute-11-13
         466546_48   ser_std   ja1.sh    bm102 CG       0:06      1 compute-11-13
         466546_49   ser_std   ja1.sh    bm102 CG       0:10      1 compute-11-13
         466546_50   ser_std   ja1.sh    bm102 CG       0:09      1 compute-4-9
```

A job array looks like a normal job, except that the jobid is expended with a taskid – eg 466546_[1-50] and 466565_1.

As you can see in this example we got all 50 jobs to run concurrently as the system was idle during this test. But in practice you will see some jobs running, and some jobs waiting for their turn to run.

Job array processing can be used for serial, multi-threaded and MPI jobs alike.

You can kill all jobs in a job array with "scancel 466546" or an individual job with "scancel 466546_42" for example. You can also refer to a range of taskids such as "scancel 466546_[3-17]".

Each job in the job array will have its own output file (and error file if you are submitting with "-e"). So you have to be careful with not exceeding the 1000 files per directory limit! You may split the outputs in multiple directories or use the parallel job array tool described next.

```
> ls slurm-466546_*
slurm-466546_10.out  slurm-466546_1.out   slurm-466546_29.out  slurm-466546_38.out  slurm-466546_47.out
slurm-466546_11.out  slurm-466546_20.out  slurm-466546_2.out   slurm-466546_39.out  slurm-466546_48.out
slurm-466546_12.out  slurm-466546_21.out  slurm-466546_30.out  slurm-466546_3.out   slurm-466546_49.out
slurm-466546_13.out  slurm-466546_22.out  slurm-466546_31.out  slurm-466546_40.out  slurm-466546_4.out
slurm-466546_14.out  slurm-466546_23.out  slurm-466546_32.out  slurm-466546_41.out  slurm-466546_50.out
slurm-466546_15.out  slurm-466546_24.out  slurm-466546_33.out  slurm-466546_42.out  slurm-466546_5.out
slurm-466546_16.out  slurm-466546_25.out  slurm-466546_34.out  slurm-466546_43.out  slurm-466546_6.out
slurm-466546_17.out  slurm-466546_26.out  slurm-466546_35.out  slurm-466546_44.out  slurm-466546_7.out
slurm-466546_18.out  slurm-466546_27.out  slurm-466546_36.out  slurm-466546_45.out  slurm-466546_8.out
slurm-466546_19.out  slurm-466546_28.out  slurm-466546_37.out  slurm-466546_46.out  slurm-466546_9.out
```

SLURM: Parallel Job Array

```
matlab --nodisplay --nodesktop -r "mycode.m 2.1;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.15;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.17;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.18;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.2;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.4;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.7;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.8;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.81;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.9;exit;"

...

matlab --nodisplay --nodesktop -r "mycode.m 2.91;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.92;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.93;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.94;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.95;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.96;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.97;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.98;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.99;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.991;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.992;exit;"
matlab --nodisplay --nodesktop -r "mycode.m 2.993;exit;"
```

There are times, however, when even a job array doesn't really fit your needs. For instance, you may be looking at running 300,000 very short running jobs. Or maybe, the complexity to search through all the output files orhaving to manually "glue" them back together into a single output file is too demanding. Or maybe it's too difficult to match SLURM's taskids to actual input file names or parametric analysis. Or more probably the time delay between each jobs being scheduled to run is slowing down your processing as your jobs are really short.

For these cases Dalma has the "parallel job array" system– an NYU Abu Dhabi in-house tool – designed to make running large number of jobs even simpler.

```
./process-one-resuls file.1
./process-one-resuls file.2
./process-one-resuls file.3

...

./process-one-result file.50000
```

SLURM: Parallel Job Array

```
> cat list3
echo processing 1
echo processing 2
...
echo processing 100000

> slurm_parallel_ja_submit.sh list3
Input: list3
Actual maximum number of nodes that will be used: 8
Submitting parallel job array using the following modules:
Currently Loaded Modulefiles:
  1) NYUAD/3.0
Submitting parallel job array with OMP_NUM_THREADS= 4
Submitted batch job 466656

> squeue -u bm102
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
      466656_[1-8]   ser_std   sbatch    bm102 PD       0:00      1 (Priority)

> ls list3-466656_*
list3-466656_1.err  list3-466656_2.err  list3-466656_3.err  list3-466656_4.err
list3-466656_5.err  list3-466656_6.err  list3-466656_7.err  list3-466656_8.err
list3-466656_1.out  list3-466656_2.out  list3-466656_3.out  list3-466656_4.out
list3-466656_5.out  list3-466656_6.out  list3-466656_7.out  list3-466656_8.out

> cat list3-466656_1.out
============= job 1 ============
processing 1
============= job 2 ============
processing 2
...
processing 12499
============= job 12500 ============
processing 12500

> cat list3-466656_8.out
============= job 87501 ============
processing 87501
============= job 87502 ============
processing 87502
...
============= job 99999 ============
processing 99999
============= job 100000 ============
processing 100000
```

To submit a parallel job array you need to prepare a file where each line represents a single job to execute. You can have multiple Linux commands on a line as:

```
myprogram1
export V=42; myprogram2 | myprogram3
myprogram4; myprogram5 < inp
myprogram6 > out
```

Then you pass the file to the tool and it automatically submits the job array for you.

The tool splits all jobs into groups and proceeds to run them as a normal job array. During execution all jobs within a group are split across the cores of a compute node and run in parallel.

The output is gathered into files, 1 output file per group and 1 error file per group; the output is ordered to make things easy for you.

You get significant performance gains by eliminating the job scheduling overhead and the queuing delay for each job.

SLURM: Parallel Job Array

```
> slurm_parallel_ja_submit.sh –h
slurm_parallel_ja_submit.sh options:
    -C <constraint> (avx2 or sse)
    -t <time limit HH:MM:SS>
    -p <partition>
    -N <number of nodes>
```

The parallel job array tool allows you to set the time limit for each node – not each job!, the processor type (sse, or avx2), the partition – useful if your research group has its "compute condo", and the number of nodes to use.

The parallel job array tool determines an optimal number of nodes to use (actually job array tasks). It groups all jobs into groups processes them using conventional job array. The number of "nodes" corresponds actually to the number of groups, which is the number of tasks for the job array. The time limit parameter applies to the groups – eg an entire group must complete within the time limit.

SLURM: Parallel Job Array

```
> module load matlab
> slurm_parallel_job_array_submit.sh matlabjobs


> export OMP_NUM_THREADS=4
> slurm_parallel_job_array_submit.sh jatest
```

The parallel job array tool propagates your environment, and loaded modules, to all jobs. So to execute the previous matlab example you need to load the matlab software module prior to launching the parallel job array.

The tool also support OpenMP jobs, so you can set the number of threads before launching your parallel job array.

SLURM: Parallel Job Array

```
> slurm_parallel_ja_submit.sh list
Input: list
Actual maximum number of nodes that will be used: 8
Submitting parallel job array using the following modules:
Currently Loaded Modulefiles:
  1) NYUAD/3.0                  2) all                        3)
gcc/4.9.3                  4) mvapich2/gcc_4.9.3/2.2b
Submitting parallel job array with OMP_NUM_THREADS= 1
Submitted batch job 466638



> slurm_parallel_ja_submit.sh -N 12 list
Entered number of nodes to use: 12
Input: list
Actual maximum number of nodes that will be used: 12
Submitting parallel job array using the following modules:
Currently Loaded Modulefiles:
  1) NYUAD/3.0                  2) all                        3)
gcc/4.9.3                  4) mvapich2/gcc_4.9.3/2.2b
Submitting parallel job array with OMP_NUM_THREADS= 1
Submitted batch job 466644



> export OMP_NUM_THREADS=4
> slurm_parallel_ja_submit.sh -N 12 list
Entered number of nodes to use: 12
Input: list
Actual maximum number of nodes that will be used: 12
Submitting parallel job array using the following modules:
Currently Loaded Modulefiles:
  1) NYUAD/3.0                  2) all                        3)
gcc/4.9.3                  4) mvapich2/gcc_4.9.3/2.2b
Submitting parallel job array with OMP_NUM_THREADS= 4
Submitted batch job 466646
```

By default the tool will allow up to 8 "nodes" (groups). You can increase the number of nodes when there is a very large number of jobs to process to run faster, or when the groups can't finish within the time limit.

SLURM: Parallel Job Array

```
> cat ja.sh
#!/bin/bash
#SBATCH -p serial
#SBATCH -n 1
#SBATCH -a 1-100
sleep 5
echo -n $SLURM_ARRAY_TASK_ID " "
date

> sbatch ja.sh



> cat list4
./job.sh
./job.sh
...
./job.sh

> slurm_parallel_ja_submit.sh list4
```

For compatibilty with existing job array scripts you can use the SLURM_ARRAY_TASK_ID environment variable in your parallel job array processing.

Here the list4 input file contains the line "./ja.sh" 100 times.

An additional benefit of parallel job array processing is that you are not limited to SLURM's MaxSubmit and MaxJobs account / user limits.

Whereas you can submit a maximum of 200 jobs, and have up to 100 running concurrently using regular job array processing, using parallel job array you have no such limits.

Software Modules

```
benoit — ssh -Y benoit@dalma.abudhabi.nyu.ec
[login-0-3.local 89: which matlab
/usr/bin/which: no matlab in (/home/benoit/bin:/share/app
mins/tools/scripts:/share/apps/admins/tools/bin:/usr/loca
al/sbin:/usr/sbin:/opt/ibutils/bin:/opt/slurm/default/bin
n:/share/apps/dalma/tools:/home/benoit/bin)
[login-0-3.local 90: module avail matlab

------------------------- /share/apps/NYUAD/modules/ALL --
matlab/R2015b matlab/R2016a matlab/R2016b matlab/R2017a
[login-0-3.local 91: module load matlab/R2016a
Loading module 'matlab/R2016a'
[login-0-3.local 92: which matlab
/share/apps/NYUAD/matlab/R2016a/bin/matlab
login-0-3.local 93:
```

HPC systems differ from desktop/laptop computers in how software is installed. In a desktop/laptop you install software in a pre-defined centralized location where all users can access it. For instance the command "yum install python" will install python version 2.7.5 (on CentOS 7) under /usr/bin.

But in a large system hosting a multitude of users it is not practical to have a single version of "python" installed. Some users may need access to the latest version, while others have hard dependencies on an earlier version. And of course, this applies to all applications. So in an HPC system software is installed as independent modules which users must explicitly load into their shell environment.

Here's an example with matlab.

When you login your shell environment is empty of all software modules – **we strongly discourage loading modules automatically in your .bashrc!!**

"module avail matlab" shows you all available versions of matlab on Dalma. "module load matlab/R2016a" loads the R2016a version into your shell environment. So when you invoke "matlab" you actually use the correct version.

Modules: Commands

Software module environment are set on a per shell instance. So if you open a new terminal on a login node the environment you may have set in another terminal will not be propagated.

You can control and monitor the contents of your software modules environment with the following commands:

module list	- Display contents of your software modules environment
module purge	- Reset your software modules environment
module avail	- Display the list of all software modules available
module load XXX	- Load software module XXX into your environment
module unload XXX	- Unload software module XXX from your environment

As a guideline you should:

- Never add "module load" into your .bashrc file
- Use "module purge" before loading a new module environment to run an application
- Use a shell script to run an application and include the necessary modules in the script:

```
#!/bin/bash
module purge
module load matlab
matlab –nodisplay –nodesktop –r "mycode.m"
```

If you use "csh" then you need to initialize the software module environment before using "module" commands:

```
#!/bin/csh
source /usr/share/Modules/init/csh
module purge
module load matlab
matlab –nodisplay –nodesktop –r "mycode.m"
```

Modules: Dalma Specific

Dalma implements a one-of-a-kind software module environment – developed by the HPC team in Abu Dhabi – which is designed to make using software modules simpler and safer.

By simpler we mean that you need not load dependent software modules. For instance if you load netcdf into your environment, you need not load hdf5, nor zlib. All dependencies are automatically handled. So everytime you use netcdf you get the same software modules and get the same results.

Things are also simpler as when compiling applications and libraries you do not neet to worry about specifying the directories from which to pick up libraries (`-L`) or include files (`-I`), and you do not need to list all dependent libraries. For example when linking with netcdf you do not need to list the dependent libraries, such as hdf5 and zlib; it's as simple as "mpicc –o wrf wrf-main.o wrflibs.a –lnetcdf"!

We also simplify things by making nearly all libraries compiler independent. So math libraries such as "lapack" can be used with both GNU and Intel compilers equally.

Finally, we simplify things further by deploying computationally intensive applications and libraries using an NYU Abu Dhabi designed method which automatically switches to the most efficient implementation based on the processor where you are currently executing. For instance if you load the "fftw2" math library at execution time the module will run the "sse", "avx", or "avx2" compiled version based on the processor you are currently using.

Modules: Dalma Specific

The following chart shows some software modules available on Dalma – the list is way to long to include it here...

The compiler agnostic column shows software modules can be used with GNU (GCC) or Intel compilers.

"SSE", "AVX", and "AVX2" tags show whether a software modules can automatically switch between a variety of processor architectures to maximize performance – eg can the tool benefit from floating-point calculation enhancements.

MPI tools can't be made compiler agnostic as they are tightly integrated with compilers and they include compiler specific Fortran90 components.

NYUAD 3								
NAME	VERSION	Compiler Agnostic	GCC (4.9.3)			Intel (16.1)		
		serial	serial	openmpi 1.10.2	mvapich 2.2b	serial	openmpi 1.10.2	mvapich 2.2b
admirable	2015.01	SSE						
allinea	5.0	SSE						
amber	14		AVX2	AVX2	AVX2	AVX2	AVX2	AVX2
atlas	3.10.2	AVX2						
autoconf	2.69	SSE						
autodock	4.2.6	SSE						
automake	1.15	SSE						
bedtools	2.25.0	SSE						
blas	3.6.0	SSE-AVX-AVX2						
boost	1.57.0			AVX2	AVX2		AVX2	AVX2
bzip2	1.0.6	SSE						
cblas	3.5.0	SSE-AVX-AVX2						
cfitsio	3.380	SSE-AVX2						

Modules: Dalma Specific

Dalma's one-of-a-kind software module environment is also safer to use than normal software module environments because it .

The use of software modules is not without risk. In a classical HPC system if you forget to load a module, change the order in which you load modules, or even merely load extra unrequired modules you may end up combining software components which weren't intended to be used together. Or maybe you will unintentionally pick up a different version of a library which will produce slightly different results. Software cross-contamination is usually the result of careless practices by users.

Dalma's software environment literally eliminates such risks by hardcoding within each application and library their software dependencies and by disabling the use of LD_PRELOAD and LD_LIBRARY_PATH. So users can't override the software modules.

Finally, all software modules installed on Dalma undergo testing prior to be deployed. In all, millions of tests are performed to verify the quality of the software environment at your disposal.

The result is a simple-to-use software module environment which gives you piece-of-mind as you are garanteed to be able to reproduce computational results at will.

Modules: Dalma Specific

The software module environment on Dalma exists in 2 flavors. The default environment has automatically switches between "sse", "avx", and "avx2" versions based on the processor used, and it prevents cross-contamination between software modules. The extended flavor ("all") disables all safety measures and automatic switching of processor optimization and puts you in total control. This latter mode of use is for experts and is intended for specific non-standard problems.

Here's the list of ever growing software modules supported on Dalma in the "default" flavor.

```
------------------------------------------------------------------ /share/apps/NYUAD/modules/SOFTWARE ------------------------------------------------------------------
CCfits/2.5                cegma/2.5                 gaussian/09.D.01          jemalloc/4.1.0            mvapich2/2.2bmxm          pio/2.2.3                 szip/2.1
Desmond-Maestro/2014.02   cfitsio/3.380             gaussview/5.0.9           julia/0.5.0               namd/2.10                 pipelines/3.4.0           tangram/0.3.1
R/3.1.2(default)          charm/6.7.0               gcc/4.9.3                 lammps/2014.12            nbo/6                     platanus/1.2.4            tcltk/8.5.17
R/3.4.1                   clapack/3.2.1             geant4/10.3.2             lapack/3.6.0              ncap2/4.4.4               pnetcdf/1.7.0             tedna/1.2.2
admirable/2015.01         cmake/3.4.2               genblasta/1.0.4           libffi/3.2.1              ncl/6.3.0                 presto/2017.04            tempo/2017.05
allinea/5.0               comsol/4.4                gencore/1                 libgd/2.1.1               nco/4.6.3                 psrchive/6.0              thrax/1.2.3
allpathslg/52488          cp2k/2.6.0                glib/2.44.0               libpng/1.6.24             ncurses/6.0               pyferret/7.0.0            trimmomatic/0.32
amber/14                  cpmd/3.17.1               gmp/6.0.0                 libtool/2.4.6             ncview/2.1.7              python/2.7.11             udunits/2.2.24
anaconda/2-4.1.1          crystal14/v1.0.1          gnuplot/5.0.0             libunwind/1.2             netcdf/4.4.0              qchem/4.4                 valgrind/3.11.0
ansys/v172                cuda/7.0                  gperftools/170515         libxml2/2.9.2             netcdf-fortran/4.4.2      readline/6.3              vapor/2.5.0
apbs/1.5                  cuda/7.5                  grads/2.0.1               m4/1.4.17                 ngsTools/1.0.0            root/6.06.02              vcftools/0.1.14
arpack/96                 cufflinks/2.2.1           grads/2.1.0               maps/4.0                  oasis/3.0                 ruby/2.3.0                velvet/1.2.10
atlas/3.10.2              curl/7.54.1               greasy/2.1                mathematica/10.3.0        openfoam/2.4.0            scalapack/2.0.2           velvet-k127/1.2.10
autoconf/2.69             discovar/52488            gromacs/5.0.4             matlab/R2015b             openfst/1.6.1             scotch/6.0.4              wgrib/1.8
autodock/4.2.6            discovardenovo/52488      gsl/2.1                   mbedtls/2.6.0             openmpi/1.10.2            screen/4.3.1              wien2k/13(default)
automake/1.15             eigen/3.2.8               gvim/7.4                  mct/2.9.0                 openssl/1.0.1q            soapdenovo2/r240          wien2k/16
bedtools/2.25.0           expat/2.1.0               hdf5/1.8.16               metis/5.1.0               orca/3.0.3                soapdenovo2-prepare/2.0   workbench/1.0
bison/3.0.4               ferret/7.0                help2man/1.47.3           mgiza/2015.01             oxdna/2.2.2               sofia-ml/2015.01          xconv/1.93
blas/3.6.0                fftw2/2.1.5               hmmer/3.1b2               montjoie/2017.04          papi/5.5.1                spades/3.7.1              xios/1.0
blast/2.3.0               fftw3/3.3.4               idl/8.6                   mosesdecoder/3.0          parallel/20160222         sparsehash/2.0.2          zlib/1.2.8
boost/1.57.0              fftw3_mpi/3.3.4           idv/5.3                   mpfr/3.1.5                paraview/5.3              sqlite/3.11.0
bzip2/1.0.6               fregrid/08082013          intel/16.1                mpiblacs/3                parmetis/4.0.3            srilm/1.6.0
castep/16.1               fsl/5.0.6                 jags/3.4.0                mumps/5.0.1               pdb2pqr/2.1.1             stata/14
cblas/3.6.0               fuse/2.9.3                jdk/1.8.0_31              musket/1.1                perl/5.22.1               superlu/5.0
cdo/1.7.2                 gapcloser/1.12r6          jellyfish/2.2.5           mvapich2/2.2b             pgplot/5.2                svmtool/1.3.1
```

جامعة نيويورك أبوظبي
NYU | ABU DHABI

Modules: Dalma Specific

Here's the extended list of modules. To switch from "default" to "extended" use "module purge;module load all" and to switch back to "default" use "module purge".

```
------------------
```

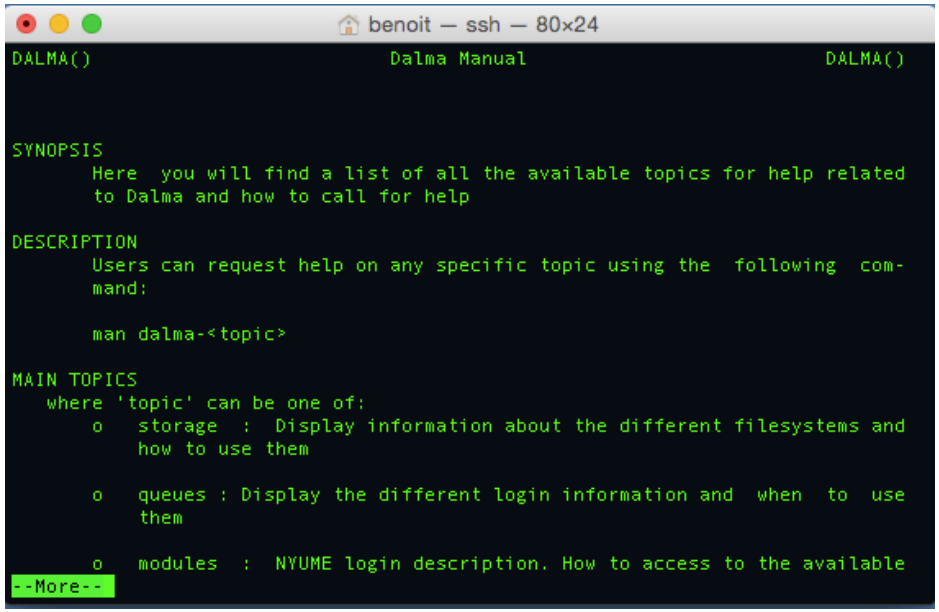
Documentation

You can find documentation, tutorials, presentations, and examples on how to use Dalma on the login nodes, and on Dalma's support web site.

Our support is divided into 2 categories:

1. User Support : for your everyday questions about how to do things on Dalma. You can reach us through raising a ticket **nyuad.it.help@nyu.edu** or walking in to A2-132B.

2. Application Support : application porting, tuning, development, and research collaboration. We work on a project basis, you can reach us in A2-131.



- Login Nodes: "man dalma"
- Web Site: https://wikis.nyu.edu/display/ADRC/HPC