Introduction to MareNostrum IV

LifeSciences startup guide

Carlos Tripiana <carlos.tripiana@bsc.es>
Oscar Hernández <oscar.hernandez@bsc.es>
David Vicente <david.vicente@bsc.es>
BSC Support Team

Barcelona, September 27th 2017
Requesting supercomputing resources

Through the “resource petitions” app available at:

https://webapps.bsc.es/resource-petitions/

Team leaders can issue “special petitions”

Intended for non-BSC personnel
BSC HPC access

Access through SSH

- OpenSSH for Linux / OSX
- PuTTY for Windows

On the facilities granted you can authenticate by:

- password
- SSH public keys
Shared Credentials

Shared users among:

- Data Transfer (dt01.bsc.es) >> Data management
- MareNostrum IV (mn1.bsc.es) >> 48 core nodes | 166000cpus
- Ultraviolet (bscsm02.bsc.es) >> 96 cpus | 1.5 TB RAM
- CTE-POWER (plogin1.bsc.es) >> P8(160 cpus) | 2xTesla P100
- CTE-KNL (klogin1.bsc.es) >> 16*64 core xeon-phi
- Nord3 (nord3.bsc.es) >> 16 core nodes
- Minotauro (mt01.bsc.es) >> 16/12 core nodes | 2* M2090/K80

Same $HOME, $USER and password
General Parallel Filesystem (GPFS)

- High performance parallel filesystem

GPFS Filesystems on the cluster

- /gpfs/apps (Support vetted applications)
- /gpfs/home (User's home, backup)
- /gpfs/projects (Inputs, custom installations, backup)
- /gpfs/scratch (Temporary files, NO backup)
- /gpfs/archive (Long term storage, batch interaction)
Filesystem limits (Quota)

- Filesystem limit per user and/or group.
- Check with: bsc_quota

Typical related problems:
- Job submission failure when $HOME is over quota
- Job execution failure when writing to over-quota filesystem

Group-shared quota on /gpfs/projects and /gpfs/scratch
/gpfs/home Filesystem usage

** Few space (~40 GB per user) 

/gpfs/home Do:

- Store source code
- Store personal scripts

/gpfs/home Don't:

- Use as production directory
You can check availability with bsc_quota and dtquota

/gpfs/archive Do:

- Store data you are not going to use soon
- Store processed final results

/gpfs/archive Don't:

- Execute commands interactively (cp, mv, ...)
- Try to put ACLs
Data Transfer Commands

Set of commands to send data transfer jobs to queues

- Available in MareNostrum and dt01 & dt02

Commands

- File movement: dtcp & dtmv
- Archiving & synchronizing: dttar & dtrsync
- Job control: dtq & dtcancel
Node's local disk (/scratch)

- All nodes have disk for temporary files
  - Accessible via $TMPDIR
  - Not shared between nodes (different to /gpfs/scratch)
  - Content erased after execution

- Useful for temporary files
  - Temporal data from MPI communication

- 200 GB disk
MareNostrum logins

3 external accessible logins:
- mn1.bsc.es
- mn2.bsc.es
- mn3.bsc.es

2 internal accessible logins:
- Login4
- Login5

No outgoing connections
- No downloads or uploads
- 5 minutes cpu time limit
## Login usage

<table>
<thead>
<tr>
<th>✅</th>
<th>Run production executions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage &amp; edit files</td>
<td>Copy large amount of files</td>
</tr>
<tr>
<td>Small &amp; medium compilations</td>
<td>Long and heavy load graphical interfaces</td>
</tr>
<tr>
<td>Submit jobs to batch system</td>
<td></td>
</tr>
<tr>
<td>Check results and prepare scripts</td>
<td></td>
</tr>
</tbody>
</table>
Compilers

Intel and GNU compiler suites available

Intel compilers available in:

- login1
- Interactive nodes ( $ salloc -p interactive)

Several versions, managed by modules

- Fortran, C, C++
- Intel (licensed)
- GCC (Free Software)

MPI compilation also managed by modules through wrappers

- mpicc, mpifort...
Module Environment (I)

- Open Source project
- Environment variables and software dependencies management
- Several versions of same program side-to-side (/gpfs/apps only)
- Typical dependencies:
  - MPI libraries
  - Mathematical libraries
### Module commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Option</th>
<th>Example</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>avail</td>
<td>[program]</td>
<td>module avail</td>
<td>List modules available</td>
</tr>
<tr>
<td>list</td>
<td></td>
<td>module list</td>
<td>List loaded modules</td>
</tr>
<tr>
<td>purge</td>
<td></td>
<td>module purge</td>
<td>Unload all modules</td>
</tr>
<tr>
<td>load</td>
<td>&lt;program[/version]&gt;</td>
<td>module load gcc/5.1.0</td>
<td>Load a module</td>
</tr>
<tr>
<td>switch</td>
<td>&lt;old&gt; &lt;new&gt;</td>
<td>module switch intel gcc</td>
<td>Change a module by another</td>
</tr>
</tbody>
</table>
MareNostrum IV uses Platform SLURM as batch system

Benefits of using jobscripts

- Defines resources needed
- Reusable
- Documents needs and requests
- Jobscripts are shellscripts with special markings

Each submission is a job
SLURM commands

- Submit a job defined in job_script.cmd
  - sbatch job_script.cmd

- Check status of jobs submitted:
  - User's: squeue

- Cancel a job:
  - scancel JobID
## SLURM Common Parameters

<table>
<thead>
<tr>
<th>Option</th>
<th>Comment</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n</td>
<td>Number of tasks</td>
<td>#SBATCH -n 32</td>
</tr>
<tr>
<td>-t</td>
<td>Wallclock limit</td>
<td>#SBATCH -t 01:00</td>
</tr>
<tr>
<td>-J</td>
<td>Job name</td>
<td>#SBATCH -J myjob</td>
</tr>
<tr>
<td>-o</td>
<td>Output file</td>
<td>#SBATCH -o %j.out</td>
</tr>
<tr>
<td>-e</td>
<td>Error file</td>
<td>#SBATCH -e %j.err</td>
</tr>
<tr>
<td>--qos</td>
<td>Queue</td>
<td>#SBATCH --qos debug</td>
</tr>
<tr>
<td>--exclusive</td>
<td>Exclusive mode</td>
<td>#SBATCH --exclusive</td>
</tr>
<tr>
<td>-D</td>
<td>Current working dir</td>
<td>#SBATCH -D=/my/path/</td>
</tr>
<tr>
<td>--reservation</td>
<td>Reservation</td>
<td>#SBATCH --reservation reserv_name</td>
</tr>
</tbody>
</table>
### SLURM Extra Parameters: Process layout

#### How to define specific load balance configurations:

<table>
<thead>
<tr>
<th>Option</th>
<th>Comment</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>--ntasks-per-core</td>
<td>Tasks per core</td>
<td>#SBATCH --ntasks-per-core 1</td>
</tr>
<tr>
<td>--ntasks-per-node</td>
<td>Tasks per node</td>
<td>#SBATCH --ntasks-per-node 48</td>
</tr>
<tr>
<td>-c</td>
<td>--cpus-per-task</td>
<td>Cpus per task</td>
</tr>
</tbody>
</table>

**Generic MNIV example:**

```bash
#SBATCH --ntasks-per-core 1
#SBATCH --ntasks-per-node 48
```
2 nodetypes:

**HIGH MEMORY NODES**

- Total of 384GBytes per node (8G per core)
- Only 216 nodes available

#SBATCH --constraint=highmem

**LOW MEMORY NODES**

- Total of 96GBytes per node (2G per core)
- Default nodes
Job queues

- Jobs are assigned to queues (QoS)
- Default queue automatically selected.
- Specify when special need: debug, interactive, graphical...
- Different queues have different limits and goals
- Check your available queues and their limits:
  - bsc_queues
- Example: #SBATCH --qos debug
Job Examples: Sequential

Sequential

#!/bin/bash

#SBATCH -n 1
#SBATCH -o %J.out
#SBATCH -e %J.err
#SBATCH -t 01:00

hostname
#!/bin/bash

#SBATCH -n 1
#SBATCH --exclusive
#SBATCH -o %j.out
#SBATCH -e %j.err
#SBATCH -t 01:00

export OMP_NUM_THREADS=16
Job Examples: Typical MPI

MPI (multiple nodes, OpenMPI)

```
#!/bin/bash

#SBATCH -n 96
#SBATCH -o %j.out
#SBATCH -e %j.err
#SBATCH -t 01:00

module purge
module load openmpi

mpirun ...
```
Job Examples: MPI + OpenMP

MPI + Threads

#!/bin/bash

#SBATCH -n 96
#SBATCH -o %j.out
#SBATCH -e %j.err
#SBATCH --ntasks-per-node=4
#SBATCH -t 01:00
export OMP_NUM_THREADS=4
module purge
module load openmpi
mpirun ...
Container executions

We do know that it is a common practice
We do know that it is a common practice Docker the most used platform
Container executions

We do know that it is a common practice

Docker the most used platform

Not available in our clusters
Container executions

We do know that it is a common practice

Docker the most used platform

Not available in our clusters

Why?

HPC Complex environment setup
We do know that it is a common practice
Docker the most used platform
Not available in our clusters

Why?

HPC Complex environment setup
- Root privileges
- High level abstraction features
Singularity

Singularity is available in Nord3, Minotauro & Marenosstrum4
Singularity

Singularity is available in Nord3, Minotauro & Marenostrom4

Why Singularity?
- Application level virtualization
- No root privileges
Singularity

Singularity is available in Nord3, Minotauro & Marenosrurum4

Why Singularity?

- Application level virtualization
- No root privileges

more info at http://singularity.lbl.gov/faq
We suggest

Install singularity locally

Available via git
We suggest

- Install singularity locally
  - Available via git

- Generate and edit your containers locally
  - $ sudo singularity shell --writable container.img
  
  **Always create folders in container for mountpoints:**

  - $ mkdir /gpfs/home /gpfs/scratch /gpfs/projects /gpfs/apps
We suggest

- Install singularity locally
  
  Available via git

- Generate and edit your containers locally
  
  $ sudo singularity shell --writable container.img

- Upload them to GPFS for production runs
  
  $ scp /local/path/container bscXX@dt01.bsc.es:/gpfs/path
  
  - > Singularity run [container]
  
  - > Singularity exec [container] [executable + args]
From Docker to Singularity

- Singularity supports pulling containers from docker repos

- Docker2singularity
  
  Docker script that converts containers

  Run the following script from your local machine

  https://github.com/singularityware/docker2singularity
Support Contact practices

When contacting support remember to:

- Specify Job Ids, software version and environment (if applies), machine, username
- Not take for granted we know what you know, want or need

We don't know who you are but we care you do fine

We have no favorites
Thank you!

For further information please contact support@bsc.es