Cluster Clonetroop: HowTo 2011

This document contains information about how to access, compile and execute jobs on Clonetroop, Laboratori de Càlcul Numeric's Cluster.

Description

Laboratori de Càlcul Numeric's cluster is a HPC Beowulf cluster that consists of 48 compute nodes (18 acquired in 2009 and 30 acquired in 2010) and one master node. All nodes are connected using an Infiniband network.

Compute nodes:

- 2 x Dell Power Edge R900 4 x Quad-Core Xeon E7440 (2.4 GHz, 16MB L3 Cache, 1066MHz FSB) with 112 GB RAM - DDR Infiniband Network
- 16 x Dell Power Edge M600 2 x Quad-Core Xeon E5440 (2.8 GHz/2x6MB, 1333Mhz FSB) with 16 GB RAM - DDR Infiniband Network
- 16 x Dell Power Edge C6100 2 x Hexa-Core Xeon L5640 (2.26 Ghz/12MB cache, 1333Mhz FSB) with 36 GB RAM - QDR Infiniband Network
- 10 x Dell Power Edge C6100 2 x Quad-Core Xeon E5640 (2.66 Ghz/12MB cache, 1066Mhz FSB) with 36 GB RAM - QDR Infiniband Network
- 4 x Dell Power Edge C6100 2 x Quad-Core Xeon E5640 (2.66 Ghz/12MB cache, 1066Mhz FSB) with 72 GB RAM - QDR Infiniband Network

Access

The only method to connect to clonetroop is using SSH protocol (shell or SFTP). Username and password should be provided by LaCaN's systems admistrator.

This is the command to use. Flag '-X' is to tunnel X11 connections.

```
ssh -X clonetroop.upc.es
```

Connection is available from:

- All computers connected to LaCaN's network
- lordvader.upc.es (worlwide access)

You can download a SSH Client from http://www.putty.org/ [http://www.putty.org/]

Disk Quota

Disk use on clonetroop is controlled using a quota system. To check your quota status you can use the command “quota”

```
[ortin@clonetroop ~]$ quota
Disk quotas for user ortin (uid 1042):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
/dev/mapper/VolGroup00-LogVol03
                6529396  10000000 10000000           70275       0       0
lordvader.upc.es:/lordvader
                4720592  5000000 5100000           72663       0       0
```

User's Home directory is /clonetroop/*username*

For more disk space, we have a scratch disk without quota limit.

Available queues

This is the list of our available queues' properties.

serial2G

- Max jobs: 112
- Max memory/job: 2000 MB
- Max Cores/job: 1
- Max execution time: none
- CPU: Quad-Core Xeon E5440 (2.8 GHz/2x6MB, 1333Mhz FSB)

Every special matlab queue will be a sub-queue of serial2G

serial3G

- Max jobs: 16
- Max memory/job: 3000 MB
- Max Cores/job: 1
- Max execution time: 2 days
- CPU: Quad-Core Xeon E5640 (2.66 Ghz/12MB cache, 1066Mhz FSB)

serial9G

- Max jobs: 32
- Max memory/job: 9000 MB
- Max Cores/job: 1
- Max execution time: 2 days
- CPU: Quad-Core Xeon E5640 (2.66 Ghz/12MB cache, 1066Mhz FSB)

serial16G

- Max jobs: 14
- Max memory/job: 16000 MB
- Max Cores/job: 1
- Max execution time: 4 days
- CPU: Quad-Core Xeon E7440 (2.4 GHz, 16MB L3 Cache, 1066Mhz FSB)

parallel3G_A

- Max slots:192
- Max memory/job: none
- Max nodes: 16
- Max CPU/node: 12
- Max CPU/job: 192
- Max execution time: 2 days
- CPU: Hexa-Core Xeon L5640 (2.26 Ghz/12MB cache, 1333Mhz FSB)

parallel3G_B

- Max slots: 64
- Max memory/job: none
- Max nodes: 8
- Max CPU/node: 8
- Max CPU/job: 64
- Max execution time: 2 days
- CPU: Quad-Core Xeon E5640 (2.66 Ghz/12MB cache, 1066Mhz FSB)

parallel_master

- Max slots: 32
- Max memory/job: none
- Max nodes: 4

- Max CPU/node: 8
- Max CPU/job: 32
- Max execution time: 4 days
- CPU: Quad-Core Xeon E5440 (2.8 GHz/2x6MB, 1333Mhz FSB)

Submitting jobs

Basic commands

Submit a job

```
qsub -q queue_name my_job.sh
```

View running jobs

```
qstatus -> show all running jobs (you can see your job id)
qstat -> show only your jobs
```

Delete a job

```
qdel jobid
```

Jobs Scripts

Standard job

```
#!/bin/bash
#
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -V

# -cwd means to execute the job for the current working directory.
# -j y means to merge the standard error stream into the standard output stream instead of having two separate files
# -S /bin/bash specifies the interpreting shell for this job to be the Bash shell.
# -V imports current environment variables (path, libraries, etc.) to the new shell

echo "Current working directory is now: " `pwd`
echo "Starting job at `date`"

#Executable
./my_standard_job

echo "JOB run completed at `date`"
```

Parallel Jobs

In this section, we focus on submit MPI jobs to the queues. Basic compiling using MPI is explained in a section below.

We will send more information about MPI environment soon.

Matlab Jobs

Matlab licences are limited. So it's possible that you experience problems submitting your job.

Please contact cluster administrator **if you plan to run more than 2 matlab jobs at the same time.** We will make a especial queue to save matlab licenses.

Matlab job script

```
#!/bin/bash

#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -V

# SGE script for MATLAB batch job
```

```
# Check on some basics:

echo "Current working directory is now: " `pwd`
echo "Starting MATLAB at `date`"

matlab -nojvm -nodisplay -nosplash < input.m > output.out

echo "MATLAB run completed at `date`"
```

Castem Jobs

```
#!/bin/bash

#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -V

# SGE script for CASTEM batch job

export DISPLAY

echo "Current working directory is now: " `pwd`
echo "Starting CASTEM at `date`"

castem2007 -m XL DPT2Dfina.dgibi

echo "CASTEM run completed at `date`"

rm fort.98
```

Compiling

GNU and Intel compilers (version 12.02) are available in this cluster. Also, MKL, and other scientific libraries are installed and ready to be used.

GNU compilers:

- C: gcc
- c++: g++
- fortran: f95

Intel compilers:

- C and C++: icc
- fortran: ifort

More information about these compilers:

- http://www-lacan.upc.edu/doc/intel/documentation_c.htm [http://www-lacan.upc.edu/doc/intel/documentation_c.htm]
- http://www-lacan.upc.edu/doc/intel/documentation_f.htm [http://www-lacan.upc.edu/doc/intel/documentation_f.htm]
- http://gcc.gnu.org/onlinedocs/gcc-4.1.2/gcc/ [http://gcc.gnu.org/onlinedocs/gcc-4.1.2/gcc/]
- http://gcc.gnu.org/onlinedocs/gcc-4.1.2/gfortran/ [http://gcc.gnu.org/onlinedocs/gcc-4.1.2/gfortran/]

MPI Parallel Environment (distributed memory)

Before starting

If you are going to compile and execute parallel jobs, you should follow these steps (order is important):

1. Define your MPI environment
2. Compile your program
3. Write your job script (script to sending job to the queue)
4. Submit your job to a execution queue

NOTE: Is very mportant to compile and run programs using the same mpi implementation.

Set up your environment

We use **mpi-selector** to easily customize your shell environment (PATH, MANPATH, INCLUDE, LD_LIBRARY_PATH, etc). Using mpi-selector allows you to cleanly set and unset your paths and environment variables. This is the list of available configurations:

1. intelmpi_4.0.1.007
2. openmpi_gcc-1.2.4
3. openmpi_gcc-1.4.2

There are more implementations of MPI. Contact the administrator if you need a different one.

To view this list you can use the following command:

```
mpi-selector --list
```

Use the command **mpi-selector-menu** to change your environment:

1. Select the mpi implementation
2. Choose “per-user”
3. Choose overwrite “yes”
4. Quit
5. Logout and login to a new session to charge the new environment.

This is the example:

```
[ortin@clonetroop test-mpi]$ mpi-selector-menu
Current system default: <none>
Current user default:   intelmpi_4.0.1.007

    "u" and "s" modifiers can be added to numeric and "U"
    commands to specify "user" or "system-wide".

1. intelmpi_4.0.1.007
2. openmpi_gcc-1.2.4
3. openmpi_gcc-1.4.2
U. Unset default
Q. Quit

Selection (1-3[us], U[us], Q): 1
Operator on the per-user or system-wide default (u/s)? u
Defaults already exist; overwrite them? (y/N) y

Current system default: <none>
Current user default:   intelmpi_4.0.1.007

    "u" and "s" modifiers can be added to numeric and "U"
    commands to specify "user" or "system-wide".

1. intelmpi_4.0.1.007
2. openmpi_gcc-1.2.4
3. openmpi_gcc-1.4.2
U. Unset default
Q. Quit

Selection (1-3[us], U[us], Q): q

WARNING: Changes made to mpi-selector defaults will not be visible until
you start a new shell!

[ortin@clonetroop test-mpi]$
```

Compiling

To compile MPI programs, we use the following wrappers:

- mpic++
- mpicc
- mpiCC
- mpif77
- mpif90

Changing your MPI environment you can define which implementation of MPI is used to compile your programs (openmpi, intel, etc.)

Submit MPI jobs to the queue system SGE

This script is used to submit a MPI job with openmpi-intel-1.4.2 configuration.

```
#!/bin/bash
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -V
# Check on some basics:
echo "Running on host: " `hostname`echo "
Current working directory is now: " `pwd`echo "
Starting job at `date`"


#DEFINE YOUR PARALLEL ENVIRONMENT
# -pe <parallel-environment> min_procs-max_procs
#Use of <parallel_environment>:
# impi for Intel MPI
# orte for GNU OpenMPI
# min_procs means the minimum of processors that your program requires to start
# max_procs means the maximum of processors that your program will use
# it is possible to put only one number, that means that this number is the minimum and maximum
#$ -pe orte 24-48


#EXECUTABLE
mpirun -np $NSLOTS ./my_mpi_executable
echo "JOB run completed at `date`"
```

NOTE for openmpi_gcc-1.2.4 users: to use this old version we should add some flags in the mpirun call.

```
#EXECUTABLE
mpirun --mca pml ob1 -np $NSLOTS ./my_mpi_executable
```

OpenMP Parallel Environment (shared memory)

Compiling

This is a basic example witten in C to test compilation using OpenMP: “omp_hello.c”

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
int nthreads, tid;

/* Fork a team of threads giving them their own copies of variables */
#pragma omp parallel private(nthreads, tid)
  {

  /* Obtain thread number */
  tid = omp_get_thread_num();
  printf("Hello World from thread = %d\n", tid);

  /* Only master thread does this */
  if (tid == 0)
    {
    nthreads = omp_get_num_threads();
    printf("Number of threads = %d\n", nthreads);
    }

  }  /* All threads join master thread and disband */

}
```

And the fortran version: omp_hello.f

```
      PROGRAM HELLO

      INTEGER NTHREADS, TID, OMP_GET_NUM_THREADS,
     +        OMP_GET_THREAD_NUM

C     Fork a team of threads giving them their own copies of variables
!$OMP PARALLEL PRIVATE(NTHREADS, TID)


C     Obtain thread number
      TID = OMP_GET_THREAD_NUM()
      PRINT *, 'Hello World from thread = ', TID

C     Only master thread does this
      IF (TID .EQ. 0) THEN
        NTHREADS = OMP_GET_NUM_THREADS()
        PRINT *, 'Number of threads = ', NTHREADS
      END IF
```

```
C     All threads join master thread and disband
!$OMP END PARALLEL

      END
```

Intel compiler

C compiler

```
 icc -openmp omp_hello.c -o hello
```

Fortran compiler

```
ifort -openmp omp_hello.f -o hello
```

GNU compiler

C compiler

```
gcc -fopenmp omp_hello.c -o hello
```

Fortran compiler

```
gfortran -fopenmp omp_hello.f -o hello
```

Running your program

Before running your openMP executable you have to set the variable OMP_NUM_THREADS with the appropiate value. For example to execute 4 threads in our program, we can use the following command:

```
export OMP_NUM_THREADS=4
```

If we don't set this variable, by default it uses the maximum number of cores in the machine.

I recommend to use the maximum number of cores of a machine executing in queues, but not when we are testing in clonetroop.

Submit OpenMP jobs to the queue system SGE

This script is used to submit a OpenMP job. It takes the maximum number of cores available in the execution node.

```
#!/bin/bash
#
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -V

# -cwd means to execute the job for the current working directory.
# -j y means to merge the standard error stream into the standard output stream instead of having two separate files
# -S /bin/bash specifies the interpreting shell for this job to be the Bash shell.
# -V imports current environment variables (path, libraries, etc.) to the new shell

#$ -pe omp 8-12
export OMP_NUM_THREADS=$NSLOTS

echo "Current working directory is now: " `pwd`
echo "Starting job at `date`"
echo "Number of SLOTS: $NSLOTS"
echo "Number of THREADS: $OMP_NUM_THREADS"
echo "Running on host: " `hostname`

#Executable
./my_openmp_executable

echo "JOB run completed at `date`"
```