22nd Summer School on PARALLEL COMPUTING

Introduction to BGQ-FERMI

m.cestari@cineca.it

CINECA

WORK ENVIRONMENT

when logged you are in your **home** space.

It is best suited for **programming** environment (compilation, small debugging sessions…)

Space available: **50 GB**

Environment variable: **$HOME**

It is also available a **scratch** space.

It is best suited for **production** environment (launch your jobs from there)

Space available: UNLIMITED (FERMI)

Environment variable: $CINECA_SCRATCH

"cindata" provides you info on your disk quotas

Everything comes with a price:

> saldo -b

Prints budget info of your username:

- validity range
- consumed resources both on the local cluster and on all clusters

Cost of a job =

(allocated compute nodes)*(16cores)*(execution time)

MODULES

CINECA's work environment is organized in modules, a set of installed tools and applications available for all users.

"loading" a module means setting some environment variables, like PATH, LD_LIBRARY_PATH, etc.

All the modules are organized in profiles

MODULE PROFILES

- PROFILES

>module av <profile_name>

  - **profile/base (default):** contains the basics and well tested modules
  - **profile/front-end:** contains the modules compiled for front-end nodes
  - **profile/advanced:** contains the applicantion modules to be tested

> module load <profile_name>

APPLICATION MODULES

List of applications

```
>module av
------------ /cineca/prod/modulefiles/base/applications --------
abinit/6.12.3                 crystal09/1.01          qe/5.0bgq
amber/12(default)       dl_poly/4.03(default)     siesta/3.1
bigdft/1.6.0                gromacs/4.5.5(default)  vasp/5.2.12
cp2k/2.3(default)         lammps/20120816       vasp/5.3.2
namd/2.9 cpmd/v3.15.3(default)
```

- Load a specific module, i.e. set specific env vars

  > module load <module_name>
- Show the variables set by a module

  > module show <module_name>
- Retrieve informations of a module

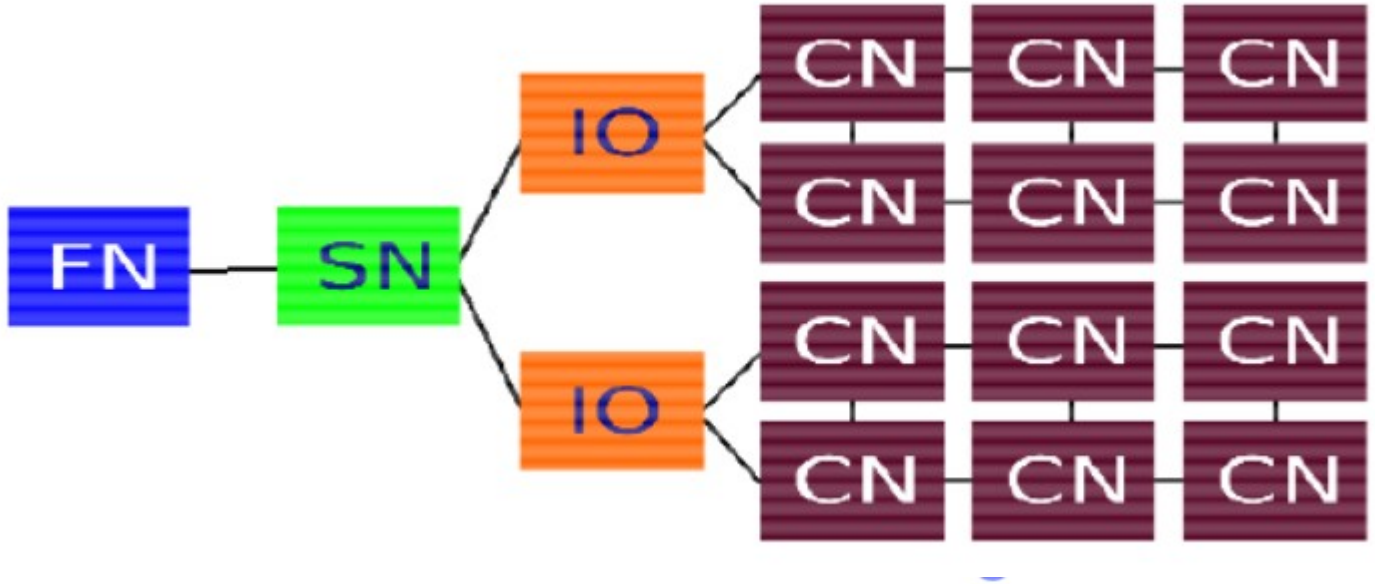
  > module help <module_name>

FERMI

Architecture: BlueGene/Q
Processor: IBM PowerA2, 1.6 GHz
Number of processors (cores): 163840
Number of nodes: 10240 (16 cores per node)
RAM: 160 TB (16 GB/core)
Interconnection network: Internal (5D torus)
Disk space: 2 PB
Power consumption: ~1 MW
Operative system: Linux (on surface)
Peak performance: 2 PFlop/s
Compilers: Fortran, C, C++
Parallel libraries: MPI,OpenMP

Login: ssh <username>@login.fermi.cineca.it

FERMI architecture



- **Front-end nodes (FN)**, dedicated for user's to login, compile programs, submit jobs, query job status, debug applications
- **Service nodes (SN)**, perform system management services, create and monitoring processes, initialize and monitor hardware, configure partitions, control jobs, store statistics
- **I/O nodes (IO)**, provide a number of OS services, such as files, sockets, process management, debugging
- **Compute nodes (CN)**, run user application, limited OS services

CROSS - COMPILATION

- Architectures of **compute nodes** and **login nodes** are different (both cpu and O.S.)
- ... and you cannot log into the compute nodes
- you need to **compile on the login nodes targeting the compute node** architecture **(*cross-compiling*)**
- you can rely on the available **wrappers** (they do all the dirty work for you)
  - it's only matter of **selecting the right wrapper**

COMPILER FAMILIES

Two Different compilers family for both front-end and back-end nodes

- **IBM Compilers**
- **GNU Compilers**

	Back-end Compilers		Front-end Compilers	
	XL family	GNU family	XL family	GNU family
C	bgxlc, mpixlc_r	gcc, mpicc	xlc	gcc
C++	bgxlc++, mpixlcxx	g++. mpicxx	xlc++, xlC	g++
Fortran	bgxlf,bgxlf90,… mpixlf90,…	gfortran, mpif90	xlf, xlf90,…	gfortran

Cross compilation:
mpixlc –O3 **–qarch=qp –qtune=qp** myprog.c

Compilation:
xlc –q64 myprog.c

EXECUTION

- command line

  > ./myexe

  - On Front-end nodes

- batch mode

  - On Back-end or Front-end nodes

  > llsubmit job.cmd

EXECUTION Back End nodes

- **Parallel** execution
- Executable compiled with serial or parallel
  BE compilers
  >bgq-gnu/4.4.6
  >bgq-xl/1.0
- NO **command line** execution
- **Batch** execution (from 64 compute nodes up to 4196 compute nodes, wall clock time up to 24 h)
- **Runjob** command
  >runjob <options>
  >man runjob

BATCH EXECUTION

- USER EXECUTABLE
  - **Shell** interpreter path
    #!/bin/bash
  - **Load Leveler Scheduler** Keywords
    # @
    # @
    # @
    .....
  - **Variables** inizialization
  - **Execution** line
    >runjob <runjob_options> : ./myexe <myexe_options>

BATCH EXECUTION

- MODULE EXECUTABLE
  - **Shell** interpreter path
    #!/bin/bash
  - **Load Leveler Scheduler** Keywords
    # @
    # @
    # @
    .....
  - **Variables** inizialization
    module load <module_name>
  - **Execution** line
    >runjob <runjob_options> : $MODULE_HOME/bin/exe <exe_options>

General LL KEYWORDS


```
# @ job_name = myname
# @ output = $(job_name).$(jobid).out
# @ error = $(job_name).$(jobid).err
# @ environment = COPY_ALL # export all variables
                                  from your submission shell
# @ job_type = bluegene
# @ wall_clock_limit = 00:10:00 # execution time h:m:s
# @ bg_size =  64 # compute nodes number
# @ notification = always|never|start|complete|error
# @ notify_user = <email_address>
# @ account_no = <budget_name> # saldo –b
# @ queue  # indicates keyword section is completed
```

RUNJOB OPTIONS

--exe path name for the executable to run

runjob –-exe <exe_name>

--args Arguments for the executable specified by --exe

runjob –-exe <exe_name> --args “<option1> <option2>”

It's easier to use ':' syntax instead of --exe and –args

runjob : ./executable arg1 arg2 ...

RUNJOB OPTIONS

--ranks-per-node number of ranks (MPI tasks) per compute node. Valid values are 1 (default), 2, 4, 8, 16, 32 and 64

```
bg_size = 128
runjob --ranks-per-node 1 : ./exe <options>
```

--np total number of ranks (MPI tasks)

```
bg_size = 128
runjob --np 128 --ranks-per-node 1: ./exe <options>
runjob --np 512 --ranks-per-node 4: ./exe <options>
#serial job:
runjob --np 1 --ranks-per-node 1: ./exe <options>
```

RUNJOB OPTIONS

--envs Sets the environment variables to export on the compute nodes

```
bg_size = 128
#MPI/OpenMP job (foreach MPI task 16 threads)
runjob -n 128 --ranks-per-node 1 --envs
OMP_NUM_THREADS=16 : ./exe <options>
```

--exp-env Exports an environment variable from the current environment to the job

```
bg_size = 128
export OMP_NUM_THREADS=16
runjob --np 128 --ranks-per-node 1  --exp-env
OMP_NUM_THREADS : ./exe <options>
```

RUNJOB OPTIONS

--envs Sets the environment variables to export on the compute nodes

```
bg_size = 128
#MPI/OpenMP job (foreach MPI task 16 threads)
runjob -n 128 --ranks-per-node 1 --envs
OMP_NUM_THREADS=16 : ./exe <options>
```

--exp-env Exports an environment variable from the current environment to the job

```
bg_size = 128
export OMP_NUM_THREADS=16
runjob --np 128 --ranks-per-node 1  --exp-env
OMP_NUM_THREADS : ./exe <options>
```

LL COMMANDS

llsubmit

llsubmit job.cmd

llq

llq -u $USER

```
[sgiulian@fen07 ~]$ llq -u amarani0

Id                       Owner      Submitted   ST PRI Class      Running On
------------------------ ---------- ----------- -- --- ------------ -----
fen04.7334.0             amarani0   9/21 15:11  I    50  parallel

1 job step(s) in query, 1 waiting, 0 pending, 0 running, 0 held, 0
    preempted
```

LL COMMANDS

llq -l <job_id>

- will print a more verbose output will be generated for job_id
- In particular you'll be notified about the bgsize you requested and the real bgsize allocated:

```
.....
.....

BG Size Requested: 1024
BG Size Allocated: 1024
BG Shape Requested:
BG Shape Allocated: 1x1x1x2
BG Connectivity Requested: Mesh
BG Connectivity Allocated: Torus Torus Torus Torus
.....
.....
```

llcancel

>llcancel <job_id>

LoadLeveler QUEUES

- Debug

2 racks with 16 I/O nodes

  - TEST - Short time (64 compute nodes, 30 min)

- Longdebug

2 racks with 16 I/O nodes

  - Long time (64 compute nodes, > 30 min)

 # @ wall_clock_limit = up to 24h

 # @ bg_size = 64

- Parallel

8 racks with 8 I/O nodes

  - PRODUCTION (from 128 to 2048 compute notes)

- bigpar

8 racks with 8 I/O nodes

  - I/O intensive jobs (from 64 to 512 compute nodes)

 # @ wall_clock_limit = up to 24h

 # @ bg_size = from 64 to 4196

Useful links documentation

Job command file keyword descriptions IBM

- http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.loadl.v5r1.load100.doc/am2ug_sbmbgjbs.htm

FERMI's User guides

- http://www.hpc.cineca.it/content/ibm-fermi-user-guide
- http://www.hpc.cineca.it/content/batch-scheduler-loadleveler-0