9th Advanced School on PARALLEL COMPUTING

Production environment on FERMI

Alessandro Marani – a.marani@cineca.it
Silvia Giuliani – silvia.giuliani@cineca.it
SuperComputing Applications and Innovation Department

CINECA

February 11 - 15, 2013

So let's say you have compiled your executable and you want to launch it…

The question is…HOW TO DO THAT????

Before you do that, let's take a look at your operational space…

Programming space

- HOME

>cd $HOME

/fermi/home/userexternal/….

- 50 GB **quota**
- Yes **backup**
- Used for **programming environment** (compilation, small debugging sessions…)

Production space

- SCRATCH

>cd $CINECA_SCRATCH

/gpfs/scratch/userexternal/….

- No **quota**
- No **backup**
- Used for **production** (i.e. job running)
- **Cleaning** procedure (everyday the clean procedure deletes all files older than 30 days) → NOT YET IMPLEMENTED

Archiving space

- CINECA_DATA

>cd $CINECA_DATA

/shared/data/userexternal/….

- **100 GB quota**
- **No Backup**
- Used for **data storage** and archiving
- **Not seen** by LoadLeveler!
- **$CINECA_PROJECT** → COMING SOON
  (no more user space, but project space)

Command “cindata” gives you the situation of your space occupancy

PROFILES

> module load <profile_name>

Available modules (“module av profile”):

- **profile/base (default)**: it contains the applications modules compiled for back-end nodes and ready to be used

- **profile/front-end**: it contains the applications modules compiled for front-end nodes and ready to be used

- **profile/advanced**. Testing profile. It contains the applicantions modules that have to be tested yet. Usable but not guaranteed

Application modules

>module available (or just “> module av”)

Shows the full list of the modules available in the profile you’re into, divided by: environment, libraries, compilers, tools, applications

Below is the list of the application modules available on profile/base, updated to 7th february, 2013

---------- /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)	siesta/3.1-TS
cp2k/2.3(default)	lammps/20120816	vasp/5.2.12
cpmd/3.15.3_hfx(default)	namd/2.9	vasp/5.3.2

Using modules

- Load a specific module

> module load <module_name>

- Show the environment variables set by a specific module

> module show <module_name>

- Get all informations about how to use a specific module

> module help <module_name>

Execution modalities

- Via command line

```
>./myexe
```

On Front-end nodes only

- Via batch

```
>llsubmit job.cmd
```

On Front-end and Back-end nodes

Execution: Front-end nodes

- **Pre and Post** processing
- **Data transfer**
- **Serial** execution (1 core)
- Executables compiled with serial **FE compilers**
  - >front-end-gnu/4.4.6
  - >front-end-xl/1.0
- **Command line** execution (10 min)
- **Batch execution** (up to 6 h)

Batch execution: Front-end nodes

- USER EXECUTABLE

>edit job.cmd

  - **Shell** interpreter path

    #!/bin/bash
  - **Load Leveler Scheduler** Keywords (we'll check them later ;-))

    # @

    # @

    # @

    ……
  - **Variables** initialization
  - **Execution** line

    ./myexe <options>

Batch execution: Front-end nodes

- MODULE EXECUTABLE
  - **Shell** interpreter path

    #!/bin/bash
  - **Load Leveler Scheduler** Keywords (we'll check them later ;-))

    # @

    # @

    # @

    ……
  - **Variables** inizialization

    module load profile/front-end

    module load <module_name>
  - **Execution** line

    exe <options>

Batch execution: Front-end nodes

LL KEYWORDS

```
# @ job_name = serial.$(jobid)
# @ output = $(job_name).out
# @ error = $(job_name).err
# @ wall_clock_limit = 0:10:00 # h:m:s
execution time up to 6 hours
# @ class = serial
# @ queue
```

Execution: Back-end nodes

- **Serial** (WARNING: 64 compute nodes are still required) and **Parallel** execution
- Executable compiled with serial and parallel **BE compilers**

  >bgq-gnu/4.4.6

  >bgq-xl/1.0
- NO **command line** execution
- **Batch** execution (from 64 compute nodes up to 2048 compute nodes, wall clock time up to 24 h)
- **Runjob** command

  >runjob <options>

  >man runjob

Batch execution: Back-end nodes

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

Batch execution: Back-end nodes

- 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>

Batch execution: Back-end nodes

LL KEYWORDS

```
# @ job_name = check
# @ 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 = 10:00:00 #execution time h:m:s, up to 24h
# @ bg_size =  64 # compute nodes number
# @ notification = always|never|start|complete|error
# @ notify_user = <email_address>
# @ account_no = <budget_name>    #saldo –b
# @ queue
```

Highlighted are the mandatory keywords, the others are highly suggested

Runjob options

EXECUTABLE

There are two ways of setting runjob with the informations about your executable:

1) Use ":" and provide executable infos how you're used to
   runjob : <exe_name> <arg_1> <arg_2>

2) Use specific runjob flags

   --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> --args <option2>

MPI TASKS SETTING

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

bg_size = 64

runjob --ranks-per-node 1 : ./exe <options> #64 nodes used, 1 task per node

runjob --ranks-per-node 4 : ./exe <options> #64 nodes used, 4 tasks per node

--np Number of ranks (MPI task) in the entire job (default=max)

bg_size = 64

runjob -n 64 -- ranks-per-node 1: ./exe <options> #64 tasks, 1 per node

runjob -n 256 -- ranks-per-node 4: ./exe <options> #256 tasks, 4 per node

runjob -n 1 --ranks-per-node 1: ./exe <options> #1 task, 1 node (serial job)

Formula: np <= bg_size*ranks-per-node

ENVIRONMENT VARIABLES

--envs Sets the environment variables for exporting them on the compute nodes

```
#MPI/OpenMP job (16 threads for each MPI task)
runjob -n 64 --ranks-per-node 1 --envs OMP_NUM_THREADS = 16 : ./exe
```

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

```
export OMP_NUM_THREADS = 16
runjob -n 64 --ranks-per-node 1  --exp-env OMP_NUM_THREADS : ./exe
```

LoadLeveler commands

Your job script is ready! How to launch it?

llsubmit

llsubmit job.cmd

Your job will be submitted to the LL scheduler and executed when there will be nodes availble (according to your priority)

llq

llq -u $USER

Shows the list of all your scheduled jobs, along with their status (idle, running, closing,…)
Also, shows you the job id required for other llq options

llq -s <job_id>

Provides information on why a selected list of jobs remain in the NotQueued, Idle, or Deferred state.

“llq –s” output

- **[sgiulian@fen07 ~]$ llq -s fen04.7334.0**
- ===== EVALUATIONS FOR JOB STEP fen04.fermi.cineca.it.7334.0 =====
- Step state : Idle
- Considered for scheduling at : Mon 24 Sep 2012 10:31:45 AM CEST
- Top dog estimated start time : Tue 25 Sep 2012 08:48:07 AM CEST
- Minimum initiators needed: 1 per machine, 1 total.
- 8 machines can run at least 1 tasks per machine, 128 tasks total.
- Not enough resources to start now.
- Shape 1x1x1x4 does not fit machine 1x5x2x2.
- Shape 1x1x4x1 does not fit machine 1x5x2x2.
- Shape 4x1x1x1 does not fit machine 1x5x2x2.
- Shape 2x1x1x2 does not fit machine 1x5x2x2.
- Shape 2x1x2x1 does not fit machine 1x5x2x2.
- Shape 2x2x1x1 does not fit machine 1x5x2x2.
- MP "R00-M0" is busy.
- MP "R00-M1" is busy.
- MP "R01-M0" is busy.
- MP "R01-M1" is busy.
- MP "R20-M0" is busy.
- MP "R20-M1" is busy.
- MP "R21-M0" is busy.
- MP "R21-M1" is busy.
- MP "R40-M0" is busy.
- MP "R30-M0" is busy.
- MP "R10-M0" is busy.
- MP "R41-M0" is busy.
- MP "R31-M0" cannot be used by job class.
- MP "R40-M1" is busy.
- MP "R30-M1" is busy.
- This step is a top-dog.

BG_SIZE = 2048 # 4 MD
BG_CONNECTIVITY = MESH

The job is a top dog.

“llq –s” output


```
[sgiulian@fen07 proveMPI]$ llq -s fen03.7942.0

===== EVALUATIONS FOR JOB STEP fen03.fermi.cineca.it.7942.0 =====

Step state                  : Idle
Considered for scheduling at     : Tue 25 Sep 2012 09:52:23 AM CEST

Minimum initiators needed: 1 per machine, 1 total.
8 machines can run at least 1 tasks per machine, 128 tasks total.
Not enough resources to start now.
Shape 2x1x1x1 does not fit machine 1x5x2x2.
MP "R00-M0" is busy.
MP "R01-M0" is busy.
MP "R20-M0" is busy.
MP "R21-M0" is on drain list.
MP "R40-M0" is not AVAILABLE (state="LoadLeveler Drained").
MP "R41-M0" is busy.
MP "R30-M0" is not AVAILABLE (state="LoadLeveler Drained").
MP "R31-M0" cannot be used by job class.
MP "R10-M0" is busy.
MP "R11-M0" cannot be used by job class.
MP "R00-M1" is busy.
MP "R21-M1" is on drain list.
MP "R40-M1" is not AVAILABLE (state="LoadLeveler Drained").
MP "R30-M1" is not AVAILABLE (state="LoadLeveler Drained").
MP "R10-M1" is busy.
MP "R01-M1" is busy.
MP "R41-M1" is busy.
MP "R31-M1" cannot be used by job class.
Not enough resources for this step to be backfilled.
This step can not become a top-dog. Global MAX_TOP_DOGS limit of 1 reached.
```

BG_SIZE =1024 # 2 MD
BG_CONNECTIVITY = MESH

The job is not a top dog and it can not be backfilled.

“llq –s” output

- [sgiulian@fen07 proveMPI]$ **llq -s fen04.7546.0**
- ===== EVALUATIONS FOR JOB STEP fen04.fermi.cineca.it.7546.0 =====
- Step state : Idle
- Considered for scheduling at : Mon 24 Sep 2012 01:56:00 PM CEST
- Minimum initiators needed: 1 per machine, 1 total.
- 8 machines can run at least 1 tasks per machine, 128 tasks total.
- Not enough resources to start now.
- Shape 1x1x1x3 does not fit machine 1x5x2x2.
- Shape 1x1x3x1 does not fit machine 1x5x2x2.
- Shape 3x1x1x1 does not fit machine 1x5x2x2.
- MP "R00-M0" is busy.
- MP "R00-M1" is busy.
- MP "R01-M0" is busy.
- MP "R01-M1" is busy.
- MP "R20-M0" is busy.
- MP "R20-M1" is busy.
- MP "R21-M0" is busy.
- MP "R21-M1" is busy.
- MP "R40-M0" is busy.
- MP "R41-M0" is busy.
- Not enough resources for this step as top-dog.
- Shape 1x1x1x3 does not fit machine 1x5x2x2.
- Shape 1x1x3x1 does not fit machine 1x5x2x2.
- Shape 3x1x1x1 does not fit machine 1x5x2x2.
- MP "R00-M0" is busy.
- MP "R00-M1" is busy.
- MP "R01-M0" is busy.
- MP "R01-M1" is busy.
- MP "R20-M0" is busy.
- MP "R20-M1" is busy.
- MP "R21-M0" is busy.

BG_SIZE = 1536 # 3 MD
BG_CONNECTIVITY = TORUS

The job will not start. It’s not possible to have the TORUS connection for all directions.

LoadLeveler commands

llq -l <job_id>

- Specifies that a long listing will be generated for each job for which status is requested.
- 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>

Removes the job from the scheduler, killing it

LoadLeveler classes

Debug

2 racks with 16 I/O nodes

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

Longdebug

2 racks with 16 I/O nodes

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

 # @ wall_clock_limit = up to 24 h

 # @ bg_size = 64

Parallel

8 racks with 8 I/O nodes

PRODUCTION (from 128 to 2048 compute notes)

Special

2 racks with 16 I/O nodes

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

Keyproject

8 racks with 8 I/O nodes

Very parallel jobs (authorized from the user support superc@cineca.it)

 # @ wall_clock_limit = up to 24 h

 # @ bg_size = from 64 to 2048

 # @ class = special

Module “superc”

>module load superc

jobtyp (provides useful information about job in the LL queues - user, tasks, times, ...)

- For using

> jobtyp <job_id>

sstat/sstat2 (provides useful information about the system status - jobs in the LL queues, allocated nodes, Midplane status,...

- For using

> sstat

> sstat2

bgtop (draws a full-terminal display of nodeboards and jobs)

>bgtop

loadHPC (calculates aggregate statistics of LL jobs)

>loadHPC

Command “saldo”

saldo -b

Prints budgets info of your username:

- Account name (the name you have to insert in your job script)
- validity ranges
- consumed resources both on the local cluster and on all clusters
- percentage for accounts enabled for given usernames

---

account	start	end	total (local h)	localCluster Consumed(local h)	totConsumed (local h)	totConsumed %

Command “saldo”

saldo -r

Prints daily resources usage report on the local cluster for

- selected username (-u)
- selected account (-a)

```
----------------Resources used from 201101 to 201212-------------

date       username   account          localCluster      num.jobs
                                                Consumed/h
-------------------------------------------------------------------------
```

SMT

It is possible to improve the efficiency of every single CPU by activating **S**imultaneous **M**ulti **T**hreading (SMT)

Each CPU is divided into threads that act as separated tasks, sharing the CPU resources to work simultaneously (with some loss because of latency)

On FERMI, you can activate 2 or 4 simultaneous threads per CPU, meaning for example that you can launch a job with 2048 or 4096 tasks asking only for 1024 cores!

This is achieved by asking for ranks-per-node = 32 (2*16) or
ranks-per-node = 64 (4*16)

Sub-blocking

Remember that you are consuming the ALLOCATED resources and not necessarily the REQUESTED resources!!

(allocated compute nodes)*(16cores)*(exec. time)

There is, however, a technique that allows to launch multiple executables on a single 64 nodes allocation, partitioning it in sub-groups of nodes called **sub-blocks**

With sub-blocking, you can get advantage of the full number of resources you have to allocate, even with smaller or not very scalable applications. Nothing is wasted!

How to use sub-blocking

Some environment variables have to be set for sub-blocking usage:

> module load subblock

You can find a complete jobscript in our LL User Guide (link at final slide)

Jobscript USER SECTION:

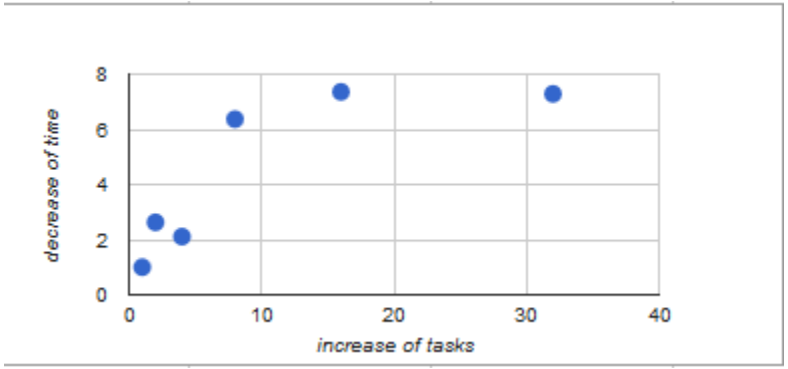
```
export N_SUBBLOCK=4              ### No. of sub-block you're asking (2,4,8,16,32,64)
export RANK_PER_NODE=16          ### No. of MPI tasks in each node.
### module load <your applications>
export WDR=$PWD
export EXE_1=$WDR/executable_1.exe
export EXE_2=$WDR/executable_2.exe
...
export EXECUTABLES="$EXE_1 $EXE_2 $EXE_3 $EXE_4"
```

Test-case: Crystal

Crystal is a general-purpose program for the study of crystalline solids (module load crystal09)

Scalability benchmark tests have been recently conducted for the software performance on FERMI:

# tasks	DFT SCF (sec)	tasks incr.	Time decr.
64	2802.25	1	1
128	1062.34	2	2.63
256	1325.76	4	2.11
512	439.44	8	6.37
1024	380.36	16	7.36
2048	384.28	32	7.29

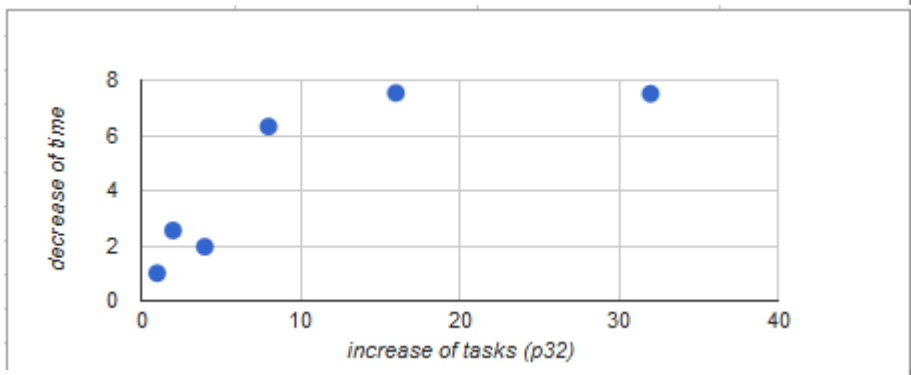


Model: phosphate BG from classical MD (10A side, model 1)
--ranks-per-node = 16

Test-case: Crystal

Another test with --ranks-per-node = 32 (SMT activated)

# tasks	DFT SCF (sec)	tasks incr.	Time decr.
64	3208.94	1	1
128	1255.35	2	2.55
256	1636.15	4	1.96
512	507.12	8	6.32
1024	425.67	16	7.54
2048	427.35	32	7.51



There is a small loss in time in terms of the number of tasks, but a great gain in terms of the actual resources allocated!

Sub-blocking test (4*512, SMT on)
DFT SCF (sec) = 539.69
small time loss, but 4x amount of work is done!

Useful 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