21st Summer School of PARALLEL COMPUTING

July 2 - 13, 2012 (Italian)
September 3 - 14, 2012 (English)

Using the IBM iDataPlex (PLX)

Marzia Rivi, m.rivi@cineca.it
Andrew Emerson, a.emerson@cineca.it

http://www.hpc.cineca.it/content/ibm-plx-gpu-user-guide

CINECA

Contents

Architecture

Users and accounting

Logging on and environment (modules)

Compiling programs

Running jobs with PBS

PLX characteristics

Model: IBM iDataPlex DX360M3
Architecture: Linux Infiniband Cluster
Processor Type:

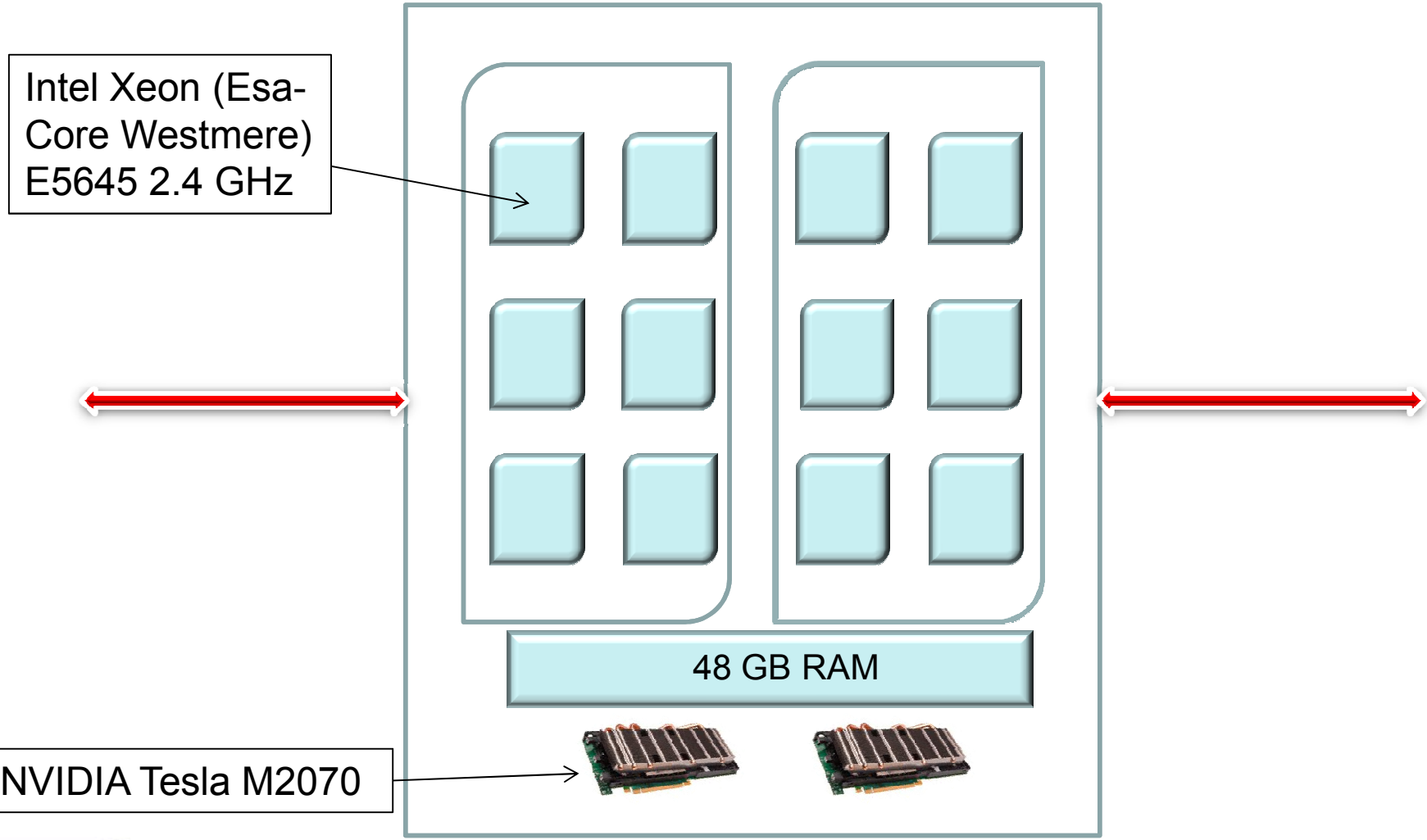
- Intel Xeon (Esa-Core Westmere) E5645 2.4 GHz (Compute)
- Intel Xeon (Quad-Core Nehalem) E5530 2.66 GHz (Service and Login)

Number of nodes: 274 Compute + 1 Login + 1 Service + 8 Fat (reserved) + 6 RVN + 8 Storage + 2 Management
All the nodes are interconnected through a Infiniband network, capable of a maximum bandwidth of 40Gbit/s.
Number of cores: 3288 (Compute)
Number of GPUs: 528 nVIDIA Tesla M2070 + 20 nVIDIA Tesla M2070Q
RAM: 14 TB (48 GB/Compute node + 128GB/Fat node)

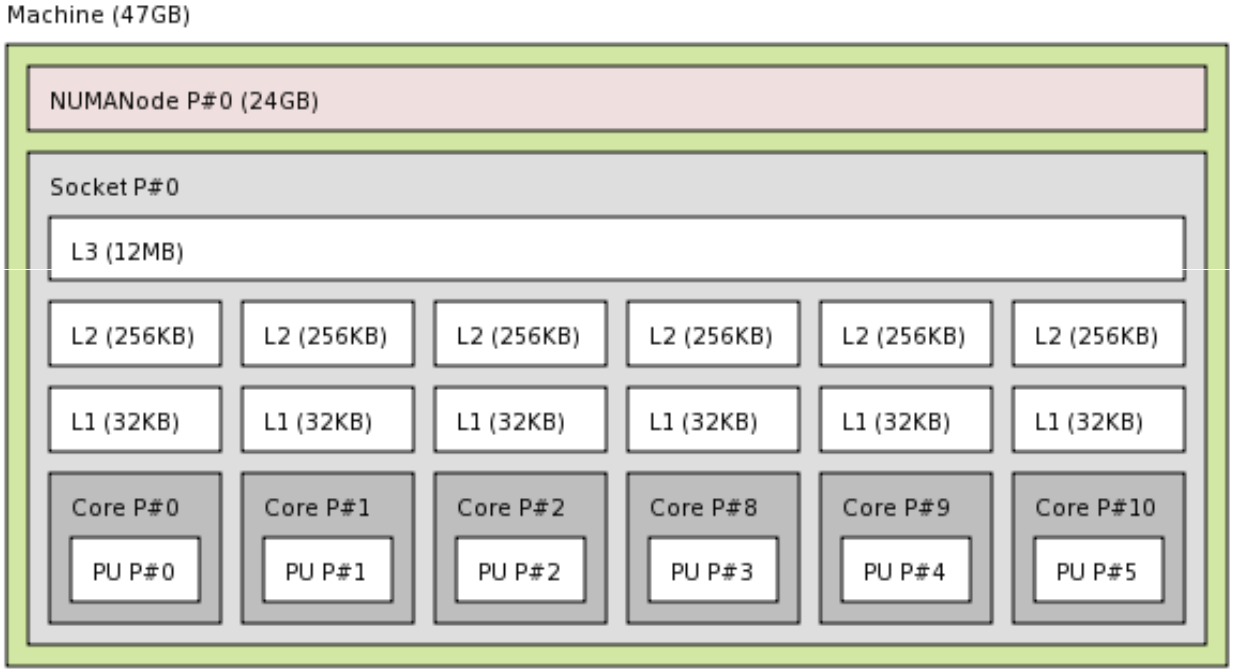
PLX system performance

Peak performance: 32 Tflops (3288 cores a 2.40GHz)
Peak performance: 565 TFlops SP or 283 TFlops DP (548 Nvidia M2070)

Compute nodes



Xeon E5645 - cache hierarchy



Disks and filesystems

CINECA Environment (Default)

$HOME:

- Permanent, backed-up,and local to PLX.
- Quota = 2GB
- For source code or important input files.

$CINECA_DATA:

- Permanent, no backup, and shared with other CINECA systems. Mounted only on login nodes (i.e. not visible in normal batch jobs)
- Quota=100Gb can be extended on request.
- Intended as a temporary backup area and file transfer between PLX-BGQ.

$CINECA_SCRATCH:

- Large, parallel filesystem (GPFS).
- Temporary (cleaning procedure on PLX being considered), no backup.
- No quota. Run your simulations and calculations here.

Who uses PLX ?

Not the main scientific resource of Cineca, but shared with industry and users with particular contracts (e.g. Weather forecasting).

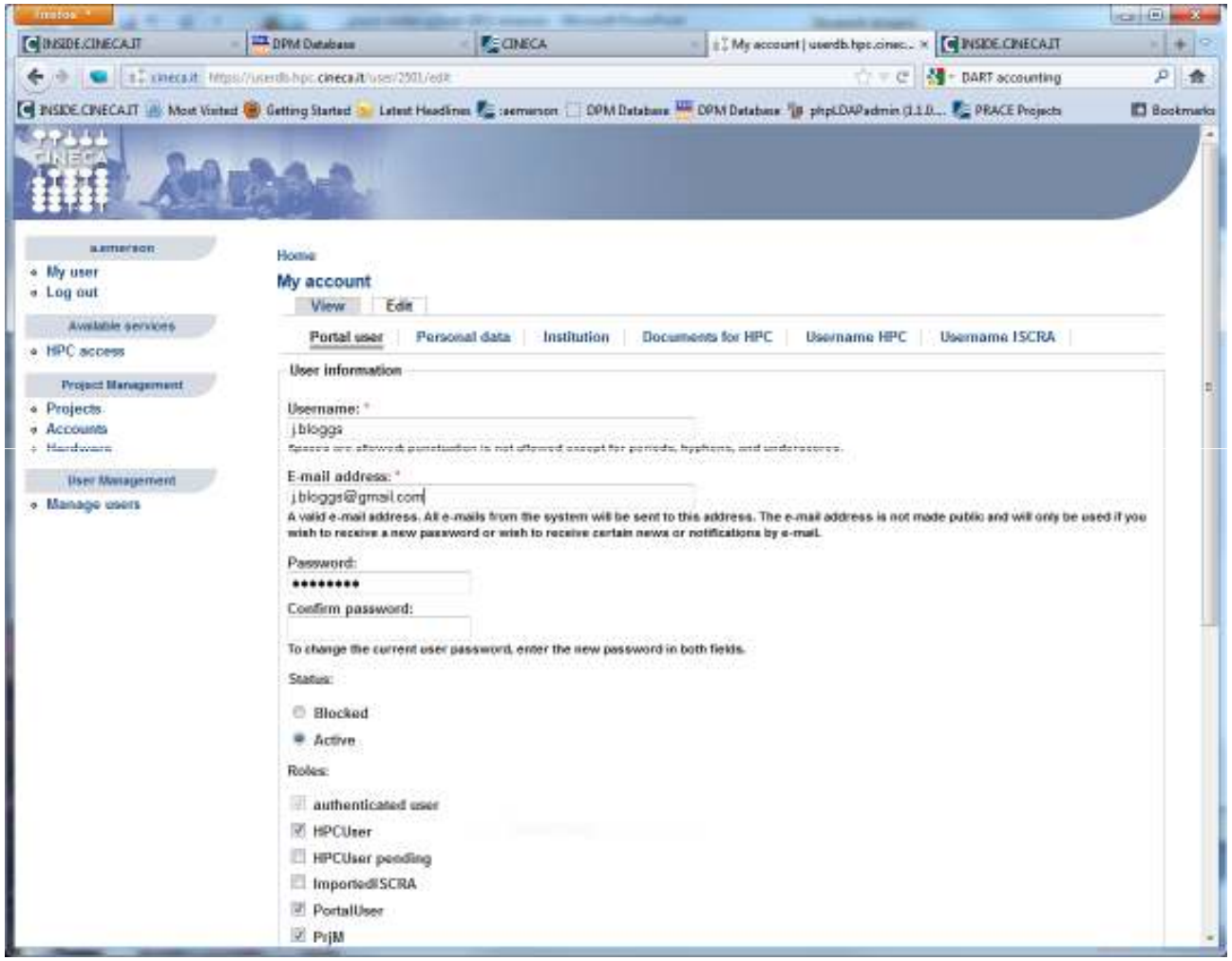
Academic users include:

- National Italian projects (ISCRA C – max. 50000 cpu/hours)
- PRACE Tier-1 (2-3 projects/call, 2.5M core hours/yr)
- Try accounts (to ask superc@cineca.it for)
- Specific agreements (e.g. SISSA, OSG, ...)

ISCRA projects are also available to non-Italian nationals providing the PI is based in an Italian Institution (collaborators can be based anywhere).

PRACE Tier-1 projects by application through Tier-1 project access. Preference given to projects capable of exploiting GPUs (http://www.prace-project.eu)

Requesting an Account



If you are a **new user** asking for an account on PLX, you may be asked to fill in your personal info on our user db

https://userdb.hpc.cineca.it

Support and documentation

Help desk system

- Email address superc@cineca.it
- Uses a trouble-ticket system to keep track of your request
- Please don't use private email addresses of CINECA staff.

HPC Portal - http://www.hpc.cineca.it/

FOR USERS section contains:

•Status of clusters
•Documentation
•Managing accounts (work in progress)

PLX USER GUIDE

http://www.hpc.cineca.it/content/ibm-plx-gpu-user-guide

How to login

- For ISCRA and other local users simple SSH

```
ssh username@login.plx.cineca.it
```

- For PRACE users preferred method is via gsissh (Globus) and X.509 certificates:

```
grid-proxy-init
gsissh gssh.plx.cineca.it -p 2222
```

If your setup is correct it won't ask for a password. The procedure assumes that the DN (Distinguished Name) of your certificate is in the PRACE LDAP (user account system). Also that your certificate in .pem format is in `$HOME/.globus`.

For more information: `http://www.prace-ri.eu/Interactive-Access-to-HPC`
Access also available via UNICORE.

USERNAMES ARE STRICTLY PERSONAL AND MUST NOT BE SHARED !

Accounting

CINECA Accounting

- **saldo** command

```
bash-3.2$ saldo -b

--------------------------------------------------------------------------------------------------------
account              start         end          total        localCluster   totConsumed     totConsumed
                                               (local h)   Consumed(local h)   (local h)              %
--------------------------------------------------------------------------------------------------------
ISCRA_PROJB          20120119    20120930      20000                  0            0                0
EXA_NUCMI            20110120    20130930    1500000            1000000      1000000               67
```

Accounting philosophy is based on the resources requested for the time of the batch job:

cost = no. of cores requested x job duration

In the CINECA system it is possible to have more than 1 budget (“account”) from which you can use time. The accounts available to your UNIX username can be found from the **saldo** command.

Accounting

CINECA training accounting

- **`repi`** command

```
bash-3.2$ repi -c

     ------------------------------------------------------------------------------
     account              start        end         total          Consumed          %
                                          ( local h)
     ------------------------------------------------------------------------------
     aco1se01          20120119    20120930      1000:00:00        0:00:00         0.0
```

Accounting philosophy is based on the resources requested for the time of the batch job:

cost = no. of cores requested x job duration

The accounts available to your UNIX username can be found from the **`repi`** command.

PLX module environment

PLX uses the module system to provide access to applications, compilers, libraries and so on. Modules typically set shell variables such as **`$PATH`** (for commands) or **`$LD_LIBRARY_PATH`** (for libs). Avoids “cluttering up” the default user shell settings which the user may never use.

Example module commands:

```
module load gromacs     # load default version of gromacs
module load namd/2.8    # load version 2.8 of NAMD
module unload lammps    # remove lammps module from environment
module show cmake       # show what the cmake module does (but
                        # don't load it)
module list             # list modules loaded by user
module avail            # list modules available
module purge            # remove all modules from the user environment
module load profile/advanced # load advanced module profile
module help fftw        # get help on the module
```

PLX module environment - continued

Some modules have dependencies, i.e. require other modules before loading:

```
module load amber/11
WARNING: amber/11 cannot be loaded due to missing prereq.
HINT: the following modules must be loaded first: IntelMPI/4.0-binary
```

```
module load IntelMPI/4.0--binary  cuda/4.0 amber/11
```

or

```
module load autoload amber/11
```

The PRACE CPE (Common Production Environment) specifies a common set of modules as well as defining variables for data storage:

```
module load prace
```

PLX compilation

Three compilers are provided:

- gnu
- pgi
- intel

and two flavours of MPI

- openmpi
- intelMPI

With the exception of GNU version 4.1.2 you need to load the appropriate compiler module:

```
module load intel
```

Often multiple versions are maintained to allow backwards compatibility:

```
module av pgi
pgi/11.1(default) pgi/11.6          pgi/11.7
```

PLX compilation

Typical compilation with external libraries:

```
module load intel fftw
module list
ifort -I$FFTW_INC -o myprog myprog.f -L$FFTW_LIB  -lfftw3f
```

The fftw module defines the `FFTW_INC, FFTW_LIB` variables which can then be used in the compilation.

MPI programs usually compiled with the “wrapped” versions of the compilers:

```
module load gnu openmpi/1.3.3--gnu--4.1.2
man mpif90
mpif90 -o myexec myprof.f90 (uses the gfortran compiler)
```

PLX compilation - GPU

GPU-enabled programs written with CUDA can be compiled after loading the cuda module.

```
module load gnu
module load cuda
nvcc -arch sm_20 -I$CUDA_INC -L$CUDA_LIB -lcublas -o
myprog myprog.c
```

Batch submission with PBS

The login nodes are meant for simple operations such as editing or short compilations.

Anything else must be done on the compute nodes via the PBS batch system. In particular, do not run parallel MPI programs on the login nodes.

UNIX commands requiring >10 min CPU also need to be done on the compute nodes.

The batch system on PLX is called PBS Pro.

You can access the compute nodes via a PBS interactive session:

```
qsub -A <account_no> -I -l select=1:ncpus=12:mpiprocs=12 -q debug
```

(ssh access to the compute nodes is disabled for non-staff users)

Once PBS finds a compute node with the requested resources it will log you on the node. Press Ctrl+D to close the session.

Batch submission with PBS

Some submission options:

```
qsub
-l <resources>          # resource request
-I                      # interactive mode
-q <queue>              #  queue name
-A <account number>     # necessary for some projects (not PRACE)
```

An important option is the resource request option `-l`

- for the wall time limit: `-l walltime=hh:mm:ss`
- to reserve nodes, cpus, mpi processes:

  `-l select=1:ncpus=12:mpiprocs=12`

  This reserves 1 “chunk” with 12 cores where we can run 12 MPI processes.
- to reserve memory:

  `-l mem=47GB` (the user can specify the requested memory up to 47 GB, on the node, default is 1GB)

Batch submission with PBS

Often more convenient to write a batch script with PBS directives:

```
#PBS -A MMM_myproj
#PBS -N job_name
#PBS -l walltime=1:00:00
#PBS -l select=16:mpiprocs=8:ncpus=8
#PBS -o job.out
#PBS -q parallel

# cd to submission directory
cd $PBS_O_WORKDIR
# load namd module
module load autoload namd
# use mpirun to run MPI prog namd
mpirun -np 128 namd2 md.namd
```

If these lines are in a file job.pbs, submit job with `qsub job.pbs`

Notice:

- We have asked for 16 chunks, each with 8 cores and 8 MPI processes
- `mpirun` to run an MPI program
- the option `-o` to rename the standard output

PBS examples

Variable chunk sizes

```
#PBS -l select=2:ncpus=8:mpiprocs=8+1:ncpus=5:mpiprocs=5
```

2 chunks of 8 cpus + 1 chunk 5 of cores.

Hybrid MPI/openmp

```
#PBS -l select=2:ncpus=8:mpiprocs=1
```

2 chunks: for each chunk 1 MPI/task + 8 threads.

Job dependencies (“chaining”)

```
qsub -W depend=afterok:JOBID.node351.plx.cineca.it job.sh
```

Wait for job JOBID to finish before starting.

Batch submission with PBS - GPUs

You should request explicitly the need for gpus:

```
-l select=1:ncpus=12:ngpus=2
```

Example:

```
#PBS -N namd_cuda
#PBS -l walltime=20:00
#PBS -l select=1:ncpus=2:mpiprocs=24:ngpus=2
#PBS -A cinstaff
#PBS -q parallel

module load autoload namd/2.8/cuda
module load cuda

# the following line is needed due to bug with CUDA 4.0 and Infiniband
export CUDA_NIC_INTEROP=1

cd $PBS_O_WORKDIR
cmd="namd2 +idlepoll dhfr.namd"
mpirun -np 2 $cmd
```

it is possible to use a GPU without asking PBS, but since nodes are not exclusive in our configuration may risk conflicts with other users => always explicitly request the GPUs.

Batch submission with PBS - qstat

```
# job information
qstat                       # shows all jobs on the system
qstat -u username           # shows jobs belonging to username
qstat -f <job_id>           # detailed information on job <job_id>

# queue information
qstat -q                    # list queues and definition
qstat -q queue_name         # definition and status of queue queue_name
qstat -Q                    # current status of queues (long format)
```

PLX queues

queue	Max node	Max/default cpus	Max/default GPUs	Max wall time
parallel	44	528/12	88/0	6h
longpar	22	264/12	44/0	24h
debug	2	24/1	4/0	30m
archive	1	1	0	4h

archive is defined on login nodes and used when you have to transfer a lot of data (interactive sessions last at most 10 minutes)

all these nodes are shared