IBM BlueGene/P Short introduction

Dr. Valentin Pavlov
Rila Solutions EAD
vpavlov@rila.bg

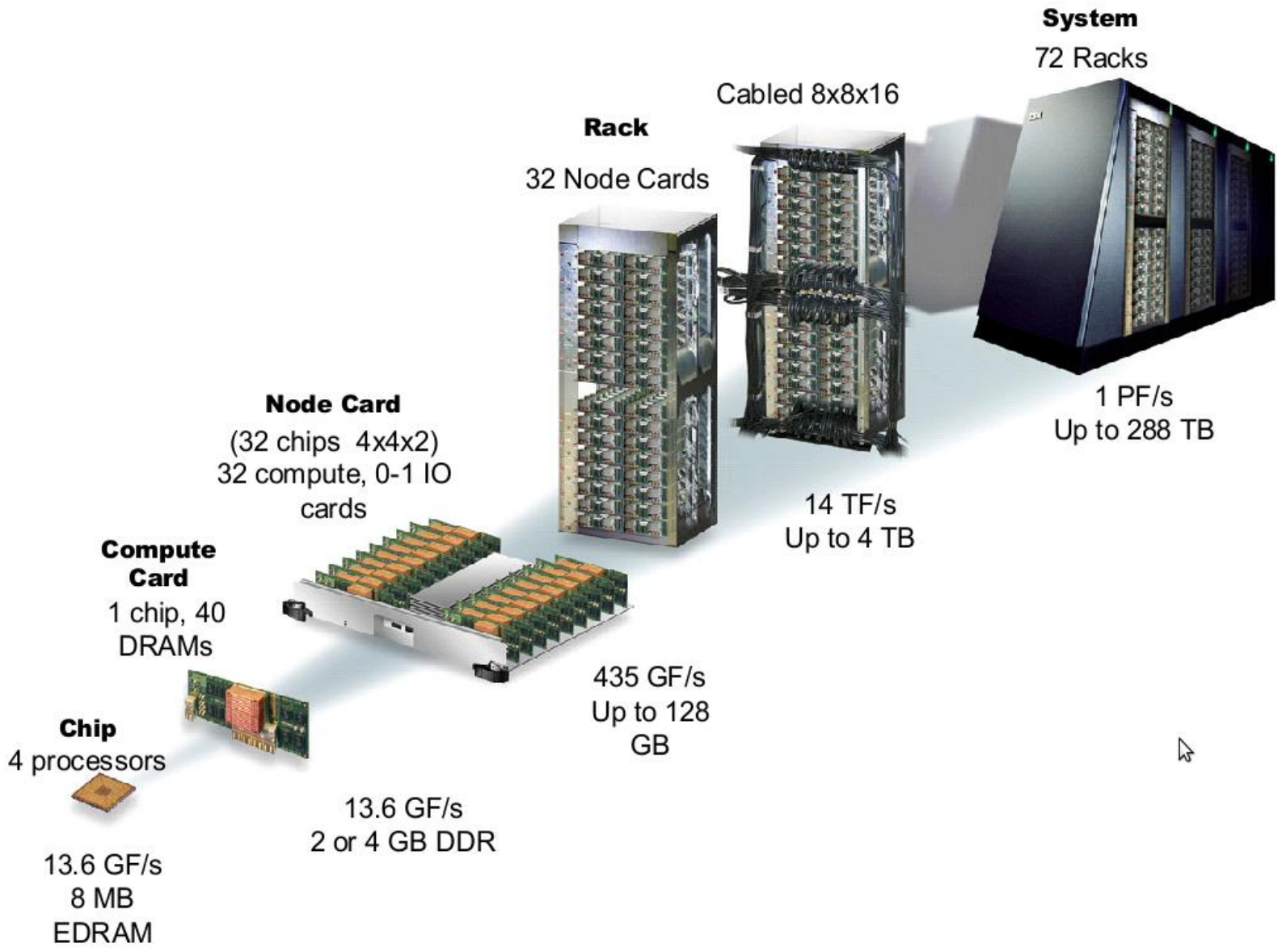
Specification

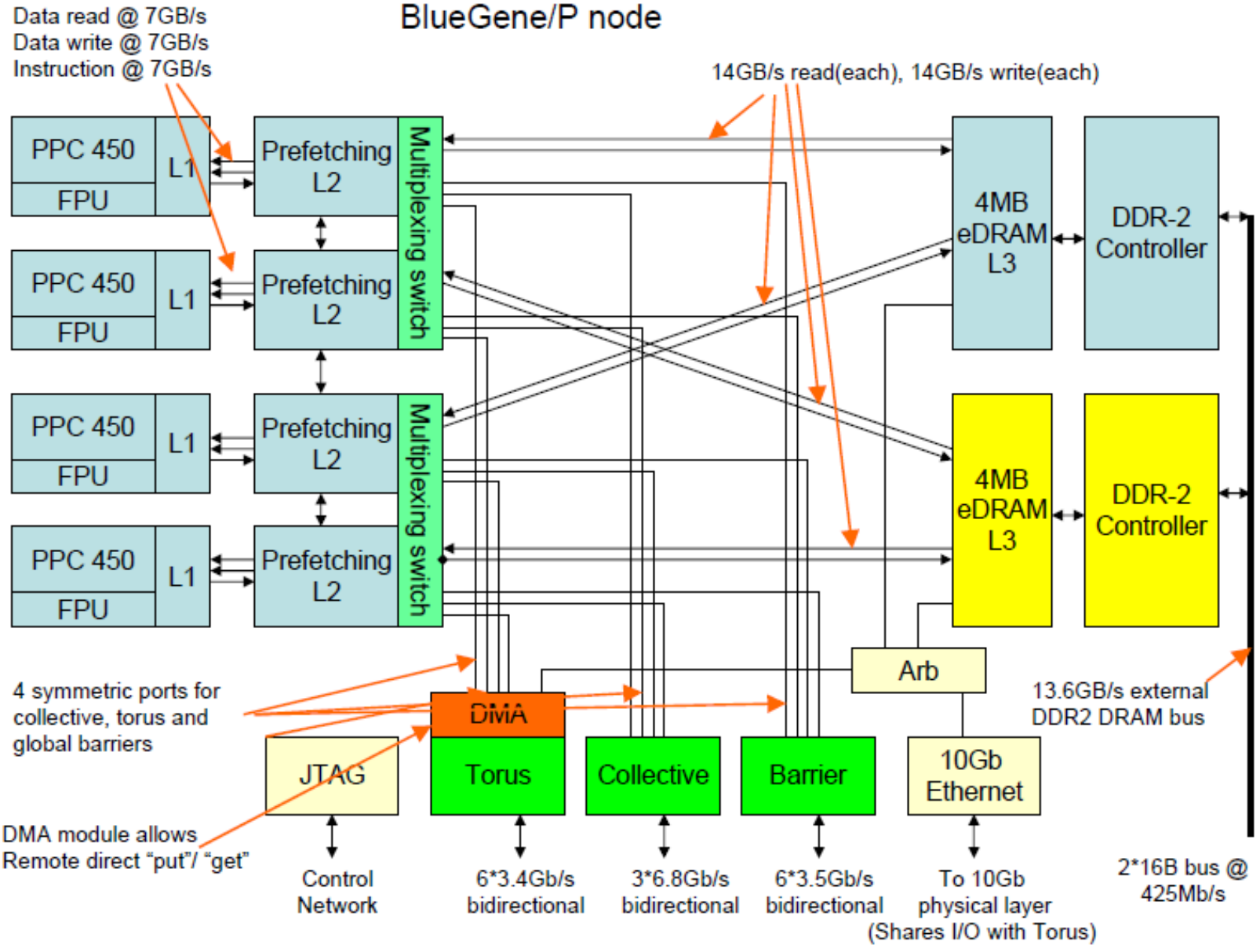
- Bulgarian Supercomputing Centre (BGSC) works with and provides access to a supercomputer IBM Blue Gene/P, consisting of 2048 computing nodes (8192 PowerPC cores @ 850 MHz, 4TB RAM)
- Connection between the computing nodes and the rest of the infrastructure: *16 channels x 10 Gb/s*
- Disk storage capacity: *12 TB*
- Front-end OS: *SLES10, externally accessible through SSH*
- Performance rating: *27.85 Tflops*
- Energy efficiency: *371.67 Mflops/W*

Benefits

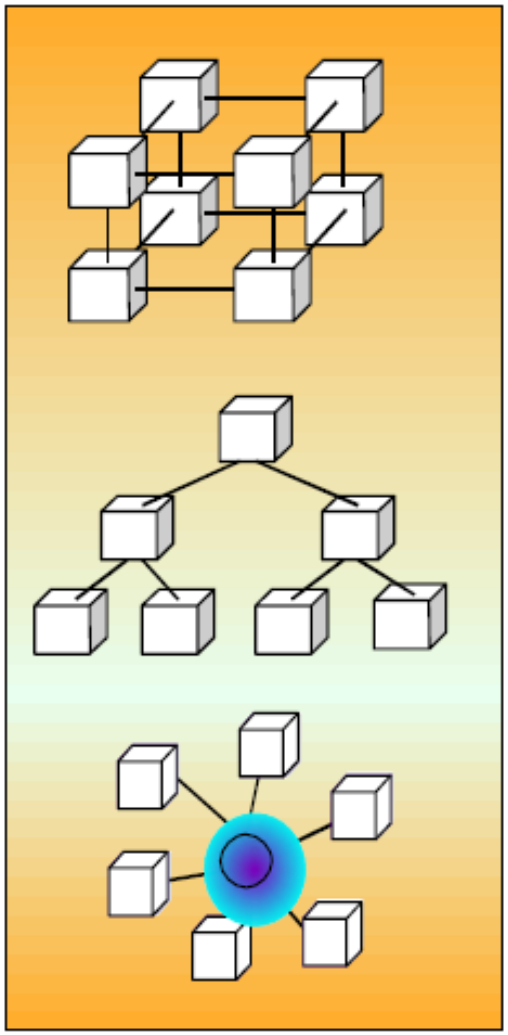
- Energy efficient
- Space saving
- Transparent, high availability, high speed network between the computing nodes
- Programming, based on standard API - MPI and OpenMP
- High scalability (thousands computing cores)
- High reliability

System organization




BlueGene/P node
Data read @ 7GB/s
Data write @ 7GB/s
Instruction @ 7GB/s
14GB/s read(each), 14GB/s write(each)
PPC 450
FPU
L1
Prefetching L2
Multiplexing switch
4MB eDRAM L3
DDR-2 Controller
Arb
4 symmetric ports for collective, torus and global barriers
DMA
JTAG
Torus
Collective
Barrier
10Gb Ethernet
13.6GB/s external DDR2 DRAM bus
DMA module allows Remote direct "put"/ "get"
Control Network
6*3.4Gb/s bidirectional
3*6.8Gb/s bidirectional
6*3.5Gb/s bidirectional
To 10Gb physical layer
(Shares I/O with Torus)
2*16B bus @ 425Mb/s

Blue Gene/P Interconnection Networks



3 Dimensional Torus

- Interconnects all compute nodes
  - Communications backbone for computations
- Adaptive cut-through hardware routing
- 3.4 Gb/s on all 12 node links (5.1 GB/s per node)
- 0.5 µs latency between nearest neighbors, 5 µs to the farthest
  - MPI: 3 µs latency for one hop, 10 µs to the farthest
- 1.7/2.6 TB/s bisection bandwidth, 188TB/s total bandwidth (72k machine)

Collective Network

- Interconnects all compute and I/O nodes (1152)
- One-to-all broadcast functionality
- Reduction operations functionality
- 6.8 Gb/s of bandwidth per link
- Latency of one way tree traversal 2 µs, MPI 5 µs
- ~62TB/s total binary tree bandwidth (72k machine)

Low Latency Global Barrier and Interrupt

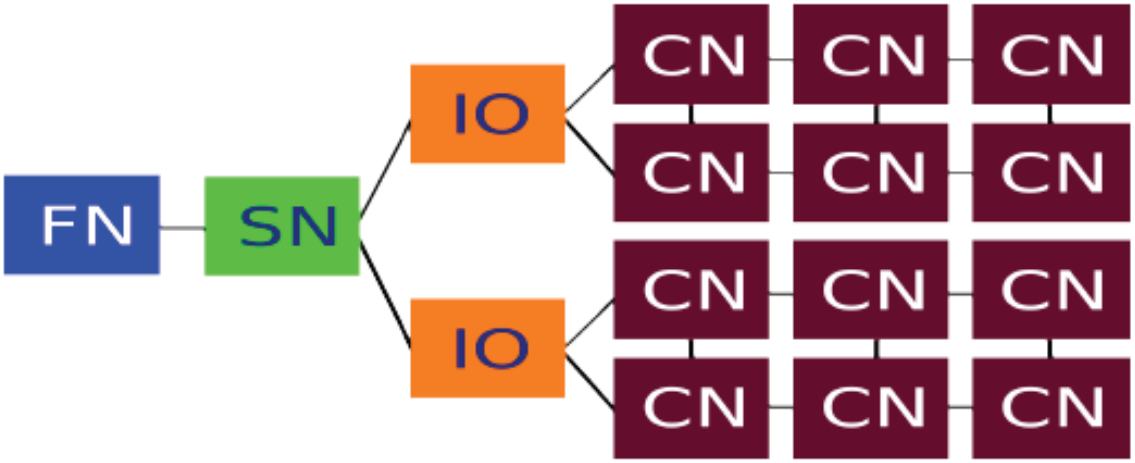
- Latency of one way to reach all 72K nodes 0.65 µs, MPI 1.6 µs

Other networks

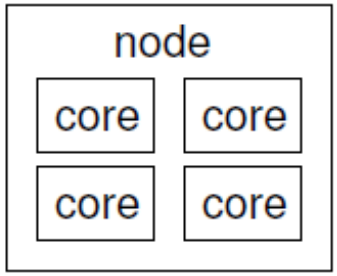
- 10Gb Functional Ethernet
  - I/O nodes only
- 1Gb Private Control Ethernet
  - Provides JTAG access to hardware. Accessible only from Service Node system

Blue Gene Hierarchical Organization

- Front-end nodes - dedicated for user's to login, compile programs, submit jobs, query job status, debug applications
- Compute nodes – run user applications, use simple compute node kernel (CNK) operating system, ship I/O-related system calls to I/O nodes
- I/O nodes – provide a number of Linux/Unix typical services, such as files, sockets, process launching, signals, debugging; run Linux
- Service nodes - perform partitioning, monitoring, synchronization and other system management services. Users do not run on service nodes directly.

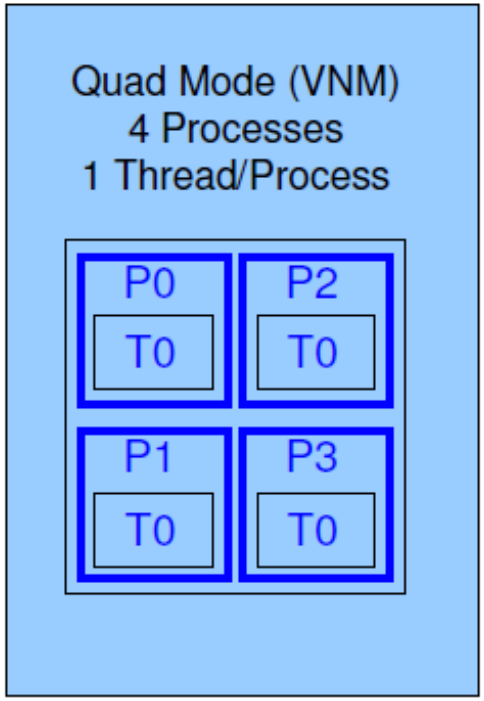


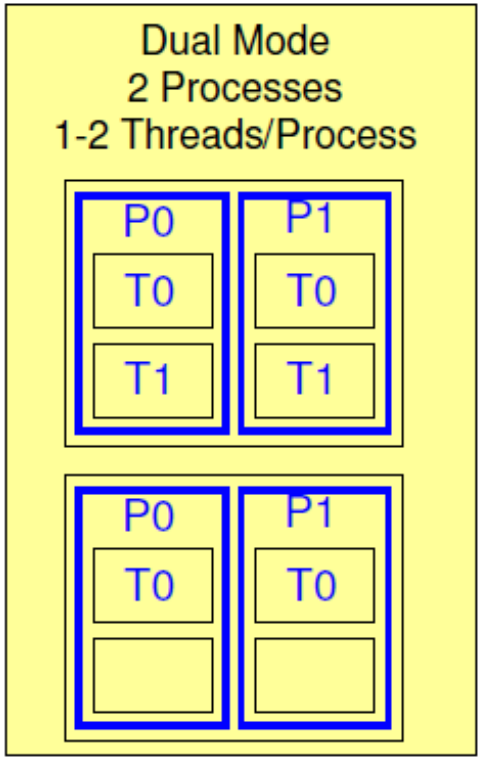
Execution Modes in BG/P per Node

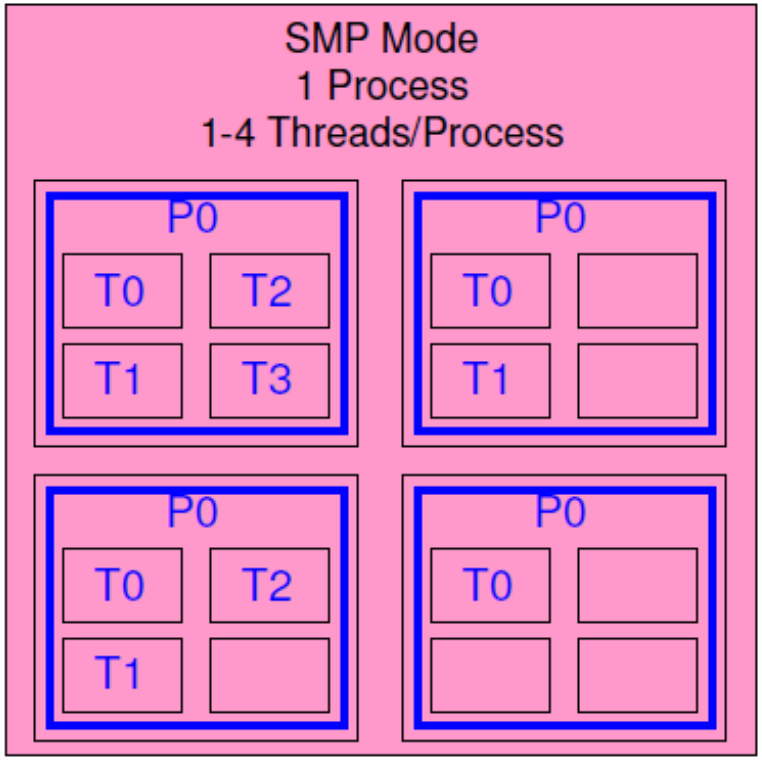


Software Abstractions Blue

- Next Generation HPC
  - **Many Core**
  - **Expensive Memory**
  - **Two-Tiered Programming Model**







File Systems

- Several places of interest for users (except the standard Linux dirs (/etc, /root, /usr, /dev and so on) are:
  - /shared1 – the fundamental user file system; its size is 4.4TB. It is visible from everywhere and this is where the jobs are run from.
  - /bgsys – BlueGene/P's system directory. This is where the system software, libraries, compilers, etc. is placed.

How to get access

- In order to get access to the machine you need:
  - Detailed description of your research project
  - The application must be shown to be scalable to at least 512 cores in order to benefit from the use of the supercomputer
  - Signed and written agreement with the usage policies. http://www.scc.acad.bg/index.php?option=com_content&view=article&id=102&Itemid=125&lang=bg
  - Send a request to bgteam@scc.acad.bg and we will send you the forms which you need to fill.
  - Your request will be duly reviewed by the management of the National Supercomputing Applications Consortium and accepted/rejected.

How to perform access

- Upon approval, an account will be created for you;
- The access is remote and performed via SSH (so it is secure);
- The name of the machine is: **bg-fen.scc.acad.bg**
- Example Linux command:
  - `ssh bg-fen.scc.acad.bg -l <username>`
- Be aware that SSH works on TCP port 22; If your organization has firewall, it has to be configured to let input and output trafic to this port for the specified machine;

Password change

- Upon your very first entering, the system will require you to change your password. Thus, please mind the messages that are written on the screen:

```
vpavlov@linux-hoe5:~> ssh bg-fen.scc.acad.bg -l vpavlov
Password:
Password change requested. Choose a new password.
Old Password:
New Password:
Reenter New Password:
Password changed.
Last login: Wed Oct 14 21:31:46 2009 from XXX.XXX.XXX.XXX


                                   ** BlueGene/P **

** This internal systems must only be used for conducting **
** IBM business or for purposes authorized by IBM management **
** Use is subject to audit at any time by IBM management. **
vpavlov@bgpfen:~>
```

How to copy your data

- Data to and from the machine can be copied by using a secure shell copy client, e.g. Scp:
  - `scp somefile username@bg-fen.scc.acad.bg:`
  - `scp -r somedir username@bg-fen.scc.acad.bg:`
- In the second example, the -r switch is used to specify recursive action – copy all files and sub-directories from the source path

Compiling your sofware

- Compiling applications (and libaries) to be used on the Computing Nodes is done via cross-compiling: the compiler works on the Front-end Node (FEN), but generates code targeted at the Computing Node (CN);
- The systam has 2 sets of C, C++ and FORTRAN compilers: GNU Toolchaing and IBM XL compilers;
- They are at:

```
/bgsys/drivers/ppcfloor/comm/default/bin/
```

- This directory can be added to the PATH env. Variable in order to be found by the shell. This can be done in your `~/.profile` file:

```
export PATH=/bgsys/drivers/ppcfloor/comm/default/bin/:$PATH
```

GNU Toolchain

- C compiler:

```
/bgsys/drivers/ppcfloor/comm/default/bin/mpicc
```

- C++ compiler:

```
/bgsys/drivers/ppcfloor/comm/default/bin/mpicxx
```

- FORTRAN-77 compiler:

```
/bgsys/drivers/ppcfloor/comm/default/bin/mpif77
```

- FORTRAN-90 compiler:

```
/bgsys/drivers/ppcfloor/comm/default/bin/mpif90
```

IBM XL compilers

- C/C++ compiler:

```
/bgsys/drivers/ppcfloor/comm/default/bin/mpixlc

/bgsys/drivers/ppcfloor/comm/default/bin/mpixlc_r
```

- FORTRAN-77 compiler:

```
/bgsys/drivers/ppcfloor/comm/default/bin/mpixlf77

/bgsys/drivers/ppcfloor/comm/default/bin/mpixlf77_r
```

- FORTRAN-90 compiler:

```
/bgsys/drivers/ppcfloor/comm/default/bin/mpixlf90

/bgsys/drivers/ppcfloor/comm/default/bin/mpixlf90_r
```

- The `_r` versions generate thread-safe code

How to perform compilation

- In order to perform a compilation, the Makefile or configure script (or whatever build system is used) must be instructed to use the cross-compiler instead of the standard gcc (or other).
- Usually, this is done by setting the env. Variables CC, FC, F77, CXX, etc. But in any case – the installation and configuration documentation of the package should be consulted.
- For example:

```
CC=mpixlc ./configure
```

IBM XL optimizations

- IBM XL compilers can produce code, specifically optimized for the PPC 450 double-hummer processor of the computing nodes. This is achieved by specifying the flags:

```
CFLAGS="-O3 -qarch=450d -qtune=450"
```

- These are also valid for C++ and FORTRAN and so CXXFLAGS and FCFLAGS are usually set also

Example

- There is a sample program in /bgsys/local/samples/helloworld. It is equipped with a Makefile and a LoadLeveler JCF file.
- These can be used as skeleton examples for your own projects.
- In order to make the application you have to copy it to your own directory and then type 'make':

```
cp -r /bgsys/local/samples/helloworld ~/your-name

cd ~/your-name

make
```

- These commands create a file named `hello`, which can be executed on the BlueGene/P

Job execution

- Jobs are scheduled for execution by a system called LoadLeveler
- The prepared jobs are submitted for execution via the command `llsubmit`, which receives something called **Job Control File**, which describes the executing program and its environment
- This puts the job into a queue of waiting jobs. There are scheduling strategies via which the jobs are prioritized. When suitable resource is available, the next task in the queue will execute.
- You can see the contents of the queue with `llq`. If the status of the job is **R**, it is running, if **I** – it waits, and if **H**, there was a problem and you need to look at the error output and remove your job via `llcancel`.

Contents of a JCF

- /bgusr/local/samples/helloworld.jcf is an example Job Control File:

```
# @ job_name = hello
# @ comment = "This is a HelloWorld program"
# @ error = $(jobid).err
# @ output = $(jobid).out
# @ environment = COPY_ALL;
# @ wall_clock_limit = 01:00:00
# @ notification = never
# @ job_type = bluegene
# @ bg_size = 128
# @ class = n0128
# @ queue
/bgsys/drivers/ppcfloor/bin/mpirun -exe hello -verbose 1 -mode VN -np 512
```

This is how it is sent for execution:

```
cd /bgsys/local/samples/helloworld

llsubmit hello.jcf
```

JCF Parameters

- `# @ job_name = hello` The name for the job, could be anything
- `# @ comment = "This is a HelloWorld program"` Some comment
- `# @ error = $(jobid).err` Where to send the stderr. Writing to file descriptor 1 (in C) writes to this file. Note how $(jobid) is used to make this file unique. For example, if LoadLeveler gives the job an ID of 4242, then he name of the file will be 4242.err in the current working directory.
- `# @ output = $(jobid).out` Same here, but for the stdout (file descriptor 0 in C).
- `# @ environment = COPY_ALL;` This instructs LoadLeveler to copy the whole user environment when running the job. Thus, the job has the same environment as the user that executes llsubmit.

JCF Parameters (cont.)

- `# @ wall_clock_limit = 01:00:00` Time limit; after this time, the job is cancelled automatically. This cannot be more than a certain time limit imposed by the class of the job.
- `# @ notification = never` There is no infrastructure for notifications, so 'never' is a good value for this parameter.
- `# @ job_type = bluegene` This MUST be bluegene.
- `# @ bg_size = 128` This must be an integer, divisable by 128, but not larger than 2048. This gives the number of computing nodes that will be used in order to execute the job.This must correspond to the class of the job.

JCF parameters (cont.)

- `# @ class = n0128` This is the class of the job. The most important parameter. Different classes have different priorities.
- `# @ queue` This instructs LoadLeveler to put the job in the queue.
- `/bgsys/drivers/ppcfloor/bin/mpirun -exe hello -verbose 1 -mode VN -np 512` This is the actual command that sends the job to the BlueGene/P's computing nodes.
- Some parameters are:
- `-exe <executable_file>` – the executable file itself
- `-args "<arguments>"` – arguments to the executable file
- `-verbose 1` – write information about job startup/finalizing in the stderr file

JCF parameters (прод.)

- `-mode VN|SMP|DUAL` – This provides the mode of execution
- `-np N` – the number of processes on which the job will execute
- `- env BG_MAXALIGNEXP=-1` – **very important** (and not documented!) parameter, which instructs the CN kernel to ignore alignment traps. Most of the software is not intended to work on systems with alignment and if this parameter is lacking, many systems will crash.

Modes, processes and bg_size

- In order for the job to be correctly stated, the following must be true:

bg_size >= np / km

- **bg_size** is the value in the JCF file `# @ bg_size`
- **np** is the value of the -np argument to mpirun
- **km** is a coefficient of the mode: 1 за SMP, 2 за DUAL, 4 за VN
- bg_size must be divisable by 128 and correspond to the class of the job

bg_size examples

- `-mode VN -np 400` : bg_size = 128
- `-mode VN -np 600` : bg_size = 256
- `-mode DUAL -np 400` : bg_size = 256
- `-mode DUAL -np 600` : bg_size = 512
- `-mode SMP -np 400` : bg_size = 512
- `-mode SMP -np 600` : bg_size = 1024

Geometry

- A rack has two parts – midplanes (upper and lower).
- Each midplane has 512 CN
- **When bg_size >= 512** you can specify the geometry, which may improve performance with some packages (e.g. GROMACS)

Geometry (cont.)

- # @ bg_connection = MESH | TORUS | PREFER_TORUS
  - MESH is the normal mode. The nodes are connected in cube (each with its 6 neighbours).
  - TORUS – toroidal connection in which the last in line is connected with the first in line. This halves the mean path and thus improves latency.
  - PREFER_TORUS – If possible – TORUS, if not – MESH

Geometry (cont.)

- # @ bg_shape = XxYxZ – X, Y and Z midplanes in each direction(see also bg_rotate)
- # @ bg_rotate = True | False – can rotate the geometry
- For our system this is only meaningful for bg_size = 1024, without permutations (bg_rotate = False). Then, bg_shape = 1x2x1 means that the two midplanes will be in 1 rack and bg_shape = 2x1x1 means they will be in different racks.

Job Classes

- The class of the job defines:
  - Priority – larger jobs take precedence
  - The maximum number of nodes that can be used
  - The maximum time that a job can run
- There is a limit how much jobs of each class can run simultaneously
- There are several classes:

Classes (cont.)

Class	Number of jobs	Max CNs	Time Limit
n0128	16	128	24 часа
n0128long	16	128	7 дни
n0256	6	256	24 часа
n0512	3	512	24 часа
n1024	1	1024	24 часа
n2048	1	2048	24 часа

THE END
Thank you for your attention!