# The Blue Gene/P at Jülich

## Introduction

W.Frings, Forschungszentrum Jülich, 26.08.2008

CAPACITIES

e-infrastructure

## Overview

- Introduction
- System overview
- Using the Blue Gene/P system Jugene
- Compiling
- Running a program
- Libraries, Overview
- Debugging, …
- Appendix
  – Description of Libraries

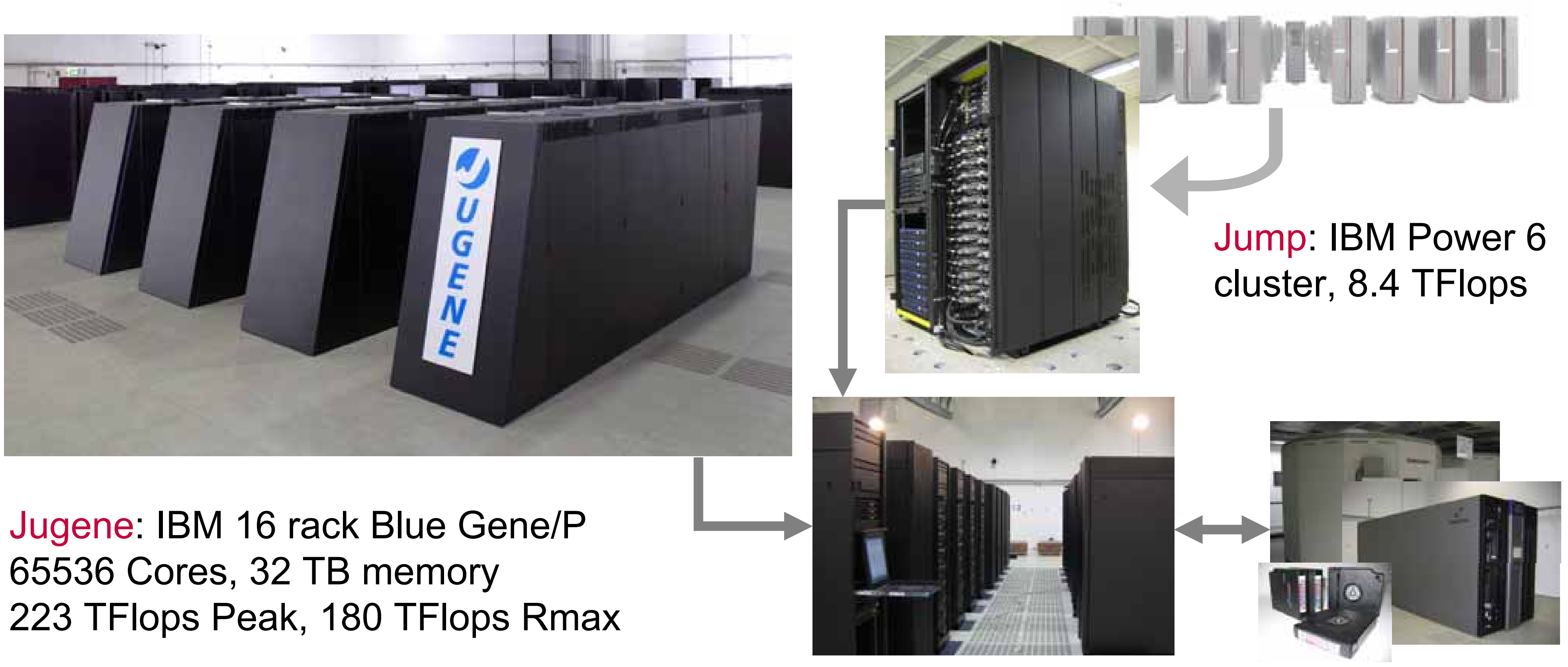
Partnership for Advanced Computing in Europe

# FZJ, Jülich Supercomputing Centre

**FZJ**, **Research Centre Jülich**
- one of the 15 Helmholtz Research Centres in Germany
- Europe's largest multi-disciplinary research center
- Area 2.2 km$^2$, 4400 Employees, 1300 scientists

**JSC, Jülich Supercomputing Centre**
- operation of the supercomputers, user support, R&D work in the field of computer and computational science, education and training
- peer-reviewed provision of computer time to national and European computational science projects (NIC, John von Neumann Institute for Computing)

## Introduction, Systems at JSC



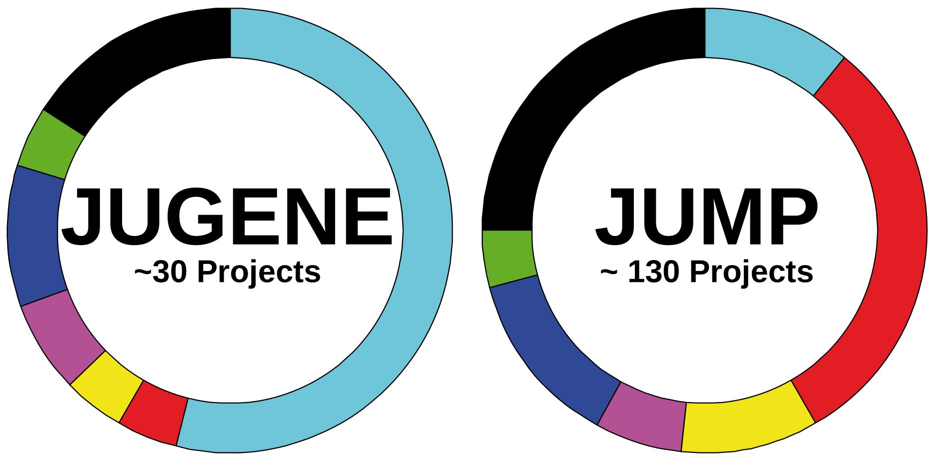
Jugene: IBM 16 rack Blue Gene/P
65536 Cores, 32 TB memory
223 TFlops Peak, 180 TFlops Rmax

Jump: IBM Power 6
cluster, 8.4 TFlops

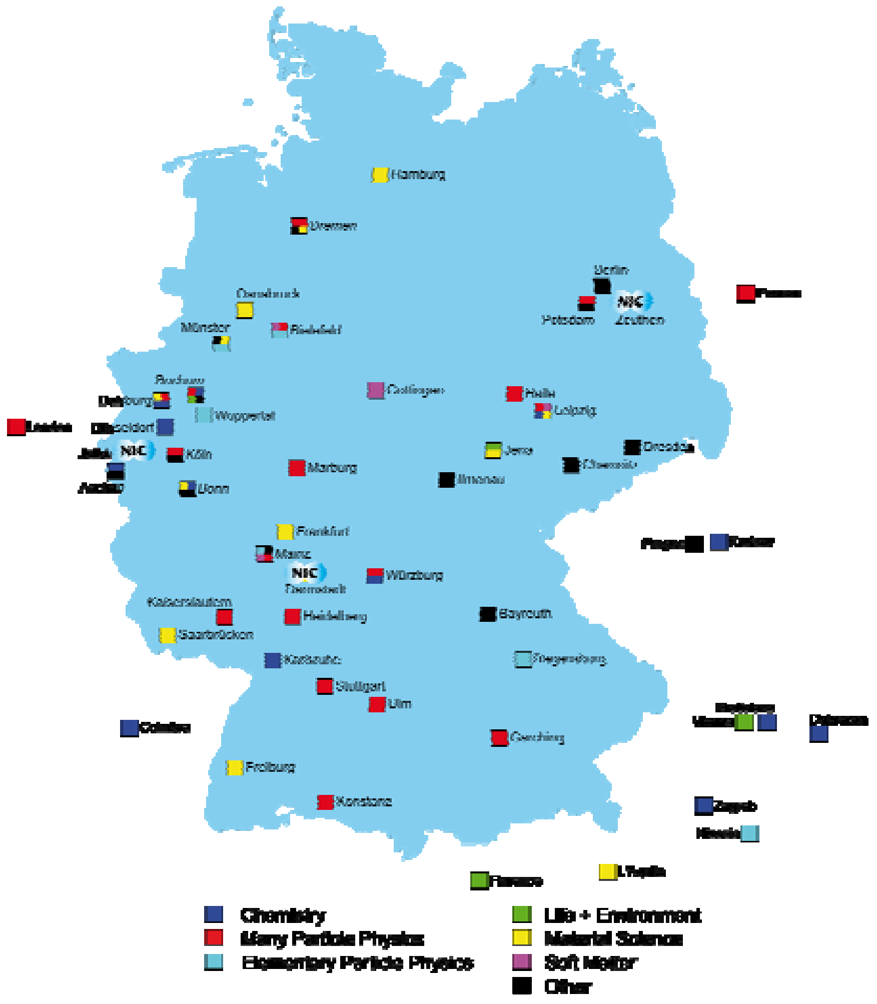
Just: GPFS-Fileserver,
32x IBM P5-55A,
~ 1PB disk space

Tape-Archives, 3 PB

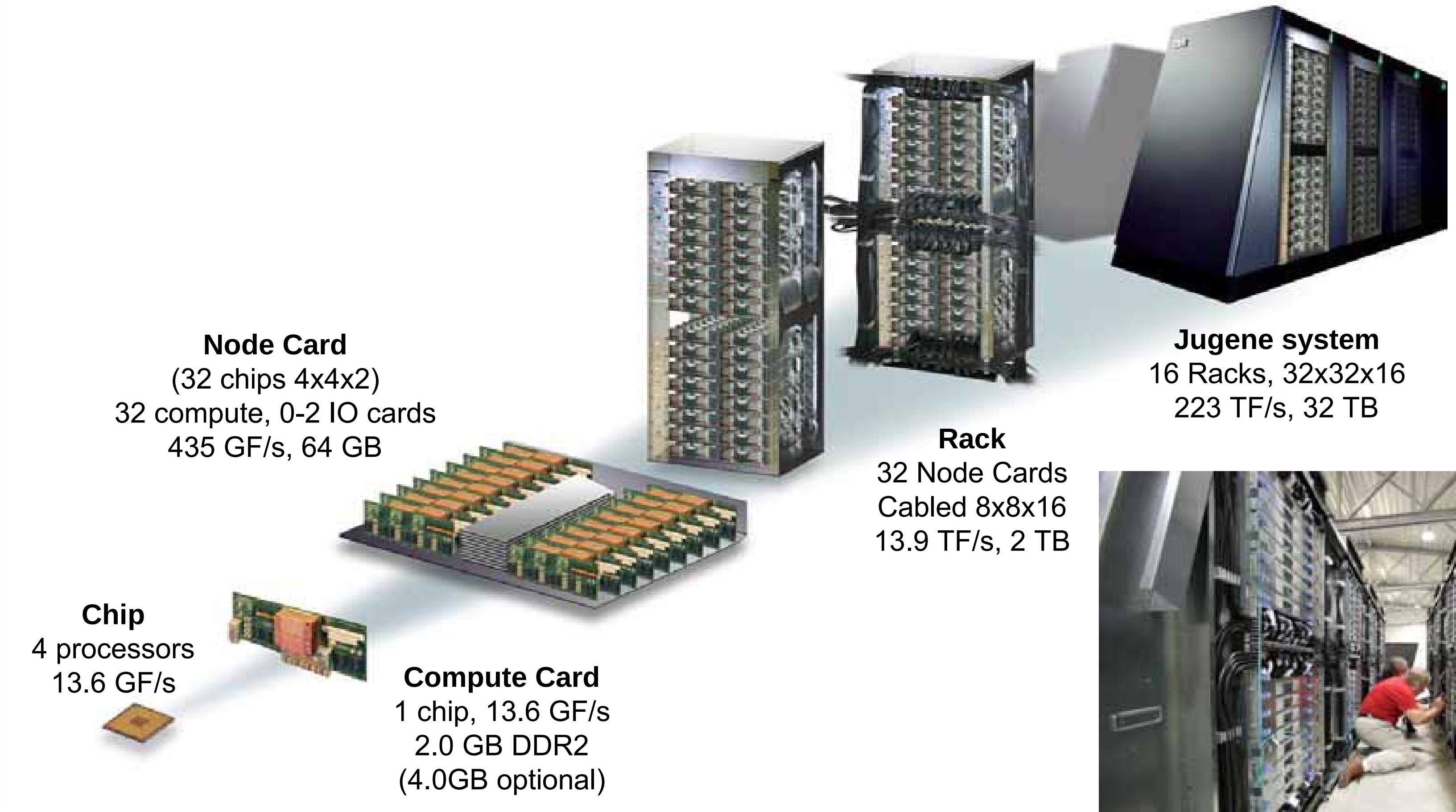
## Introduction, Users (NIC)



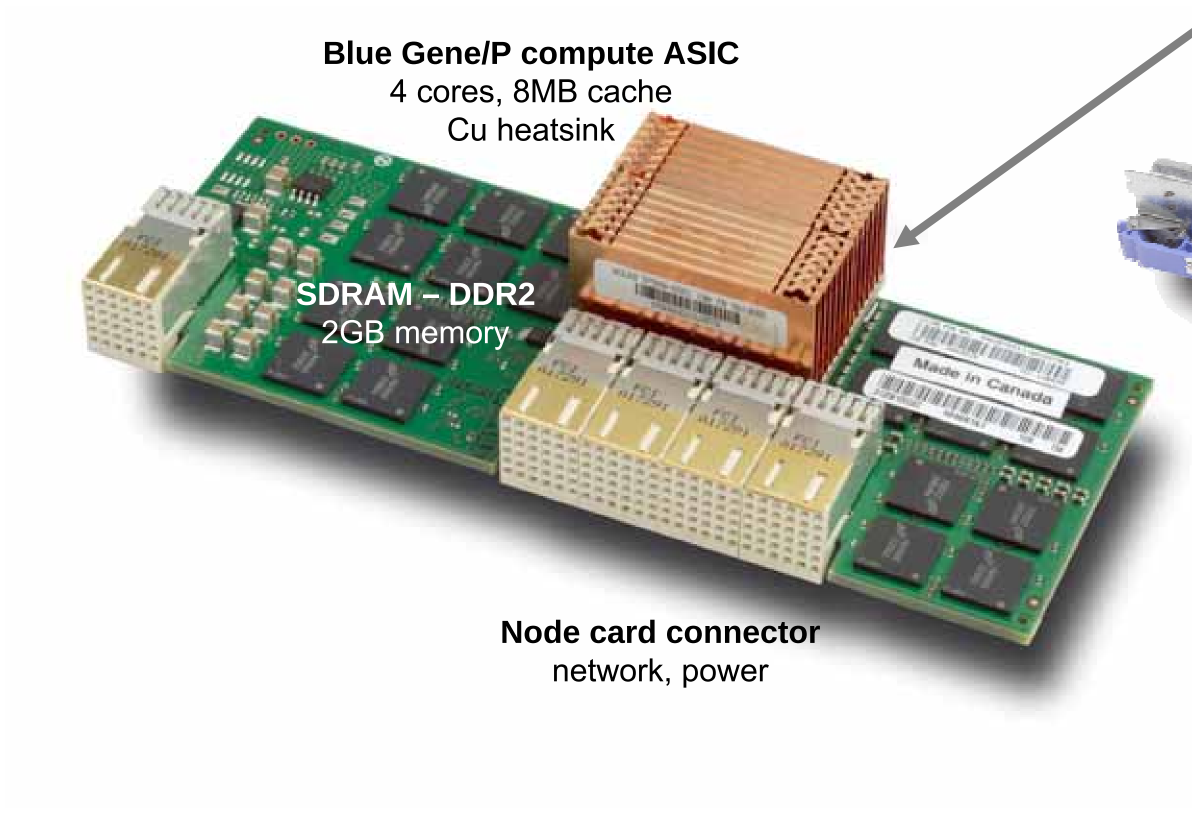
🟦 Chemistry
🟥 Many Particle Physics
🟦 Elementary Particle Physics
🟩 Life + Environment
🟨 Material Science
🟪 Soft Matter
⬛ Other

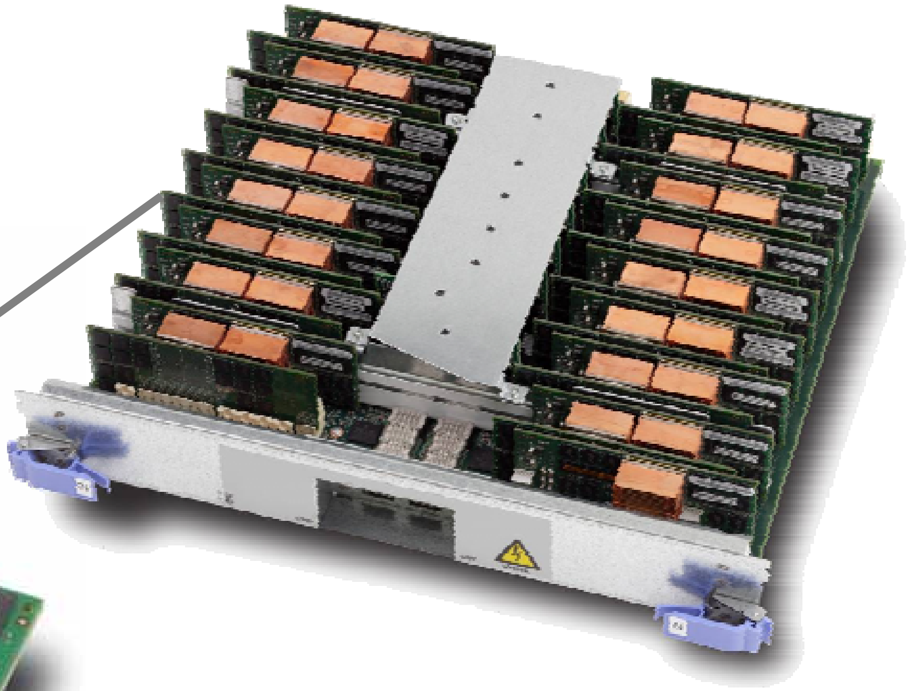


# Jugene: Blue Gene/P Building Blocks

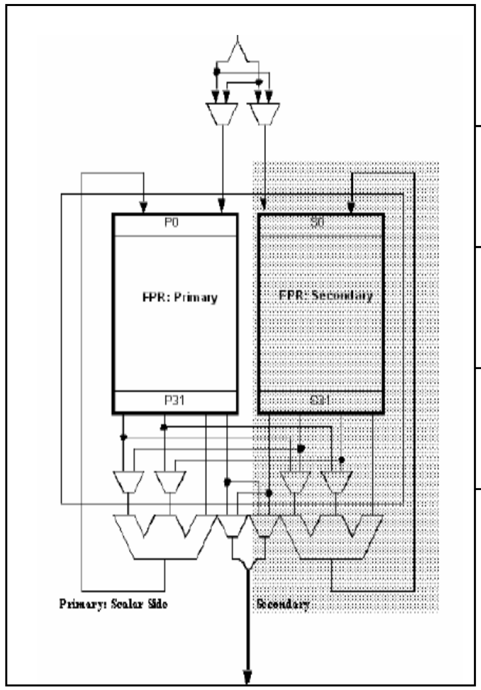


## Jugene: Blue Gene/P CPU card and node card

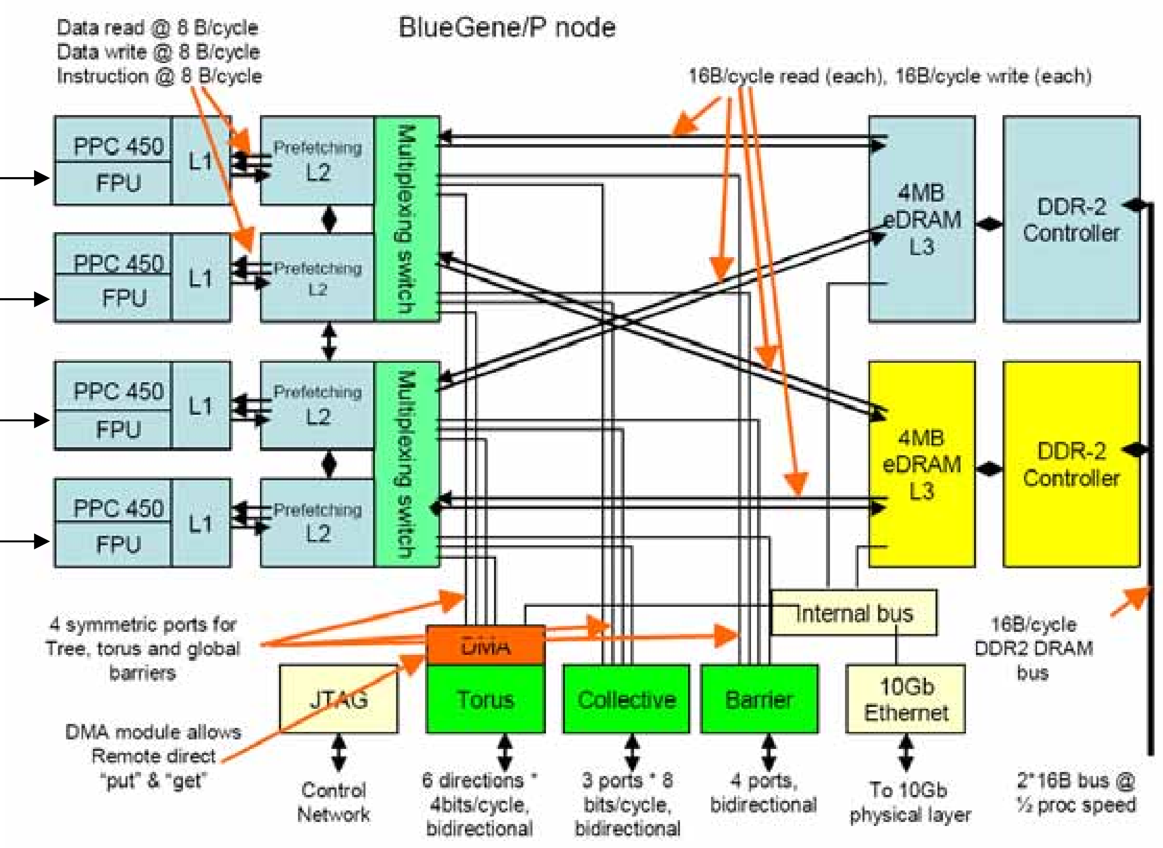




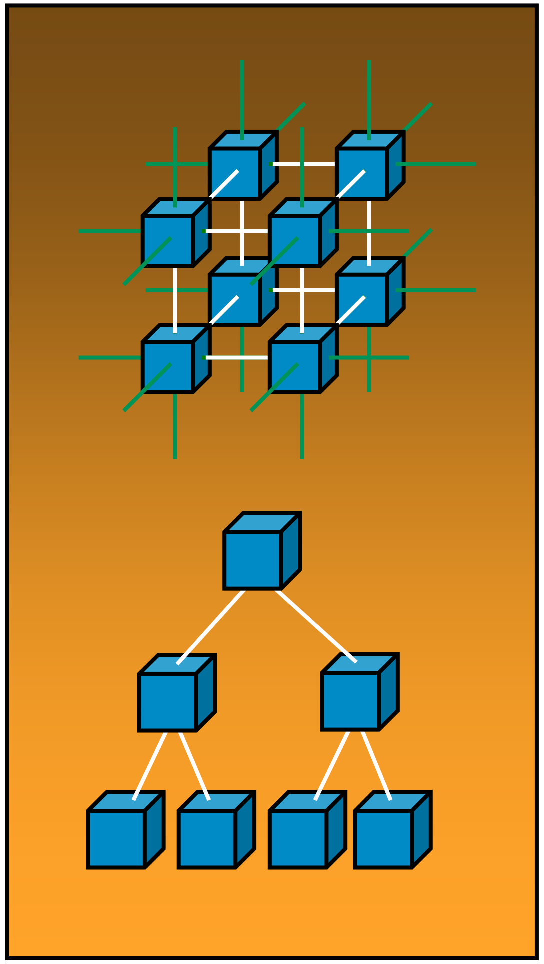
## Jugene: CPU card (4 x PowerPC 450 cores)



Dual FPU Architecture

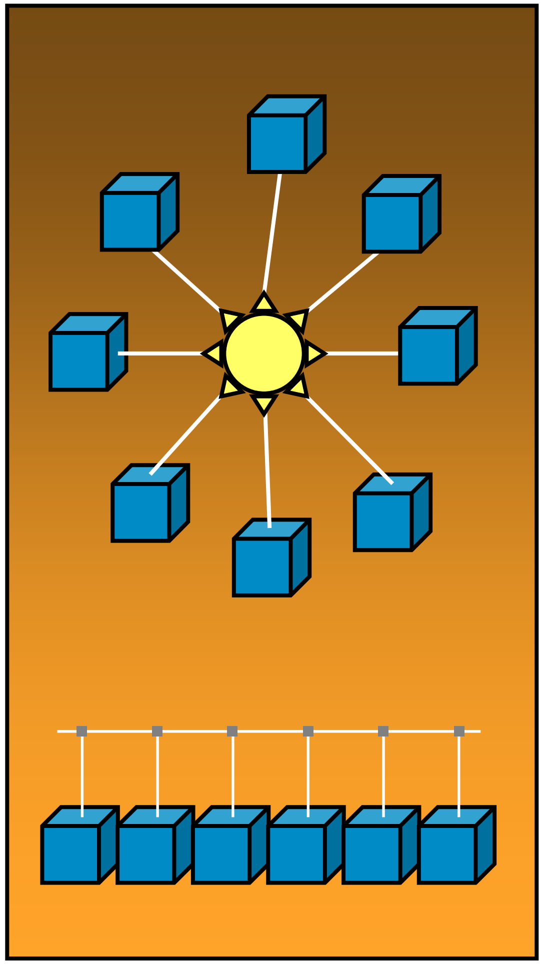


## Jugene: Blue Gene/P networks



- **3 Dimensional Torus**
  - Interconnects all compute nodes (73,728)
  - Virtual cut-through hardware routing
  - 425 MB/s on all 12 node links (5.1 GB/s per node)
  - Communications backbone for computations
  - 188TB/s total bandwidth
- **Collective Network**
  - One-to-all broadcast functionality
  - Reduction operations functionality
  - 850 MB/s of bandwidth per link
  - Interconnects all compute and I/O nodes

## Jugene: Blue Gene/P networks

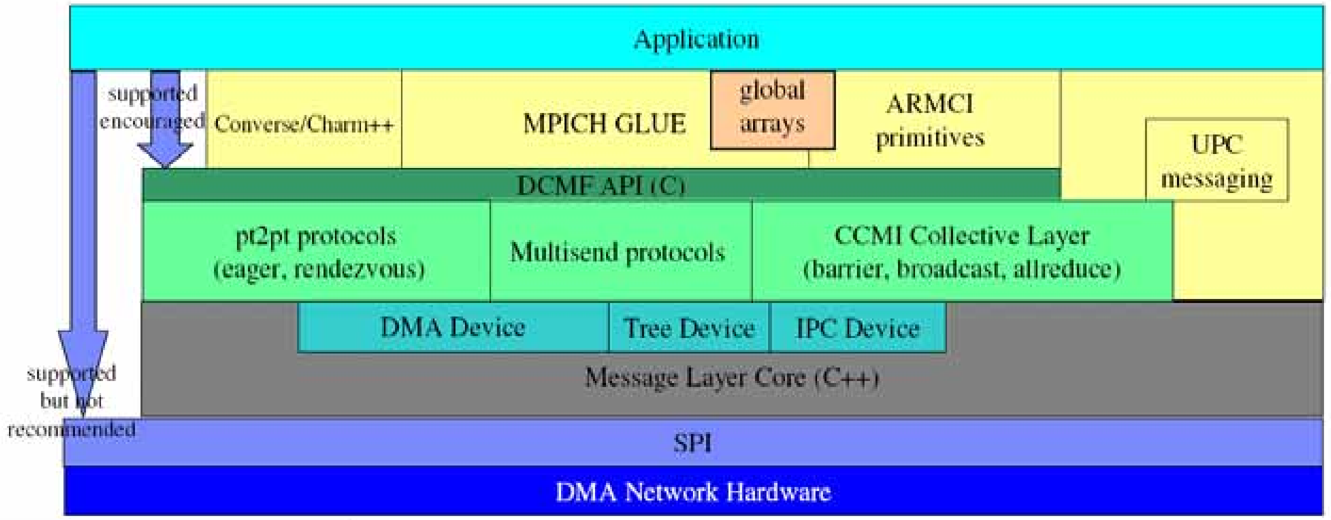


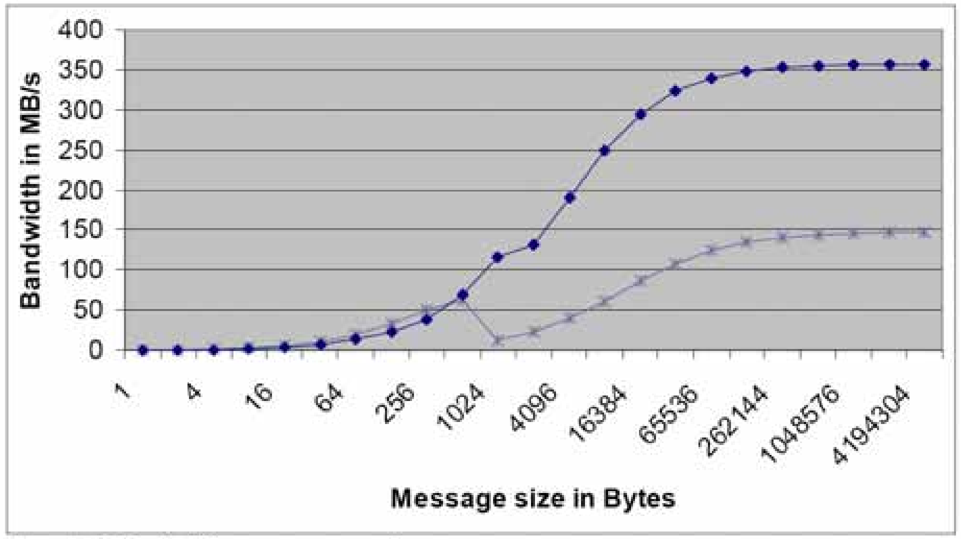
- **Low Latency Global Barrier and Interrupt**
  - Latency of one way to reach all 72K nodes
  - 0.65 µs, MPI 1.6 µs
- **External I/O Network**
  - 10 GBit Ethernet
  - Active in the I/O nodes
  - All external comm. (file I/O, control, user interaction, etc.)
- **Control Network**
  - 1 GBit Ethernet, Boot, monitoring and diagnostics

## Jugene: Blue Gene/L ←→ Blue Gene/P

| **Property** | | **Blue Gene/L** | **Blue Gene/P** |
|---|---|---|---|
| Node Properties | Node Processors<br>Processor Frequency<br>Coherency<br>L3 Cache size (shared)<br>Main Store<br>Main Store Bandwidth (1:2 pclk)<br>Peak Performance | 2* 440 PowerPC®<br>0.7GHz<br>Software managed<br>4MB<br>512MB/1GB<br>5.6GB/s<br>5.6GF/node | 4* 450 PowerPC®<br>0.85GHz<br>SMP<br>8MB<br>2GB/4GB<br>13.6 GB/s<br>13.9 GF/node |
| Torus Network | Bandwidth<br>Hardware Latency (Nearest Neighbour)<br>Hardware Latency (Worst Case) | 6*2*175MB/s=2.1GB/s<br>200ns (32B packet)<br>1.6µs (256B packet)<br>6.4µs (64 hops) | 6*2*425MB/s=5.1GB/s<br>100ns (32B packet)<br>800ns (256B packet)<br>3.2µs(64 hops) |
| Tree Network | Bandwidth<br>Hardware Latency (worst case) | 2*350MB/s=700MB/s<br>5.0µs | 2*0.85GB/s=1.7GB/s<br>3.5µs |
| System Properties | Area (72k nodes)<br>Peak Performance (72k nodes)<br>Total Power | 114m$^2$<br>410TF<br>1.7MW | 160m$^2$<br>~ 1PF<br>~2.3MW |

## Jugene: MPI





PingPong ,Torus network, BG/P vs. BG/L

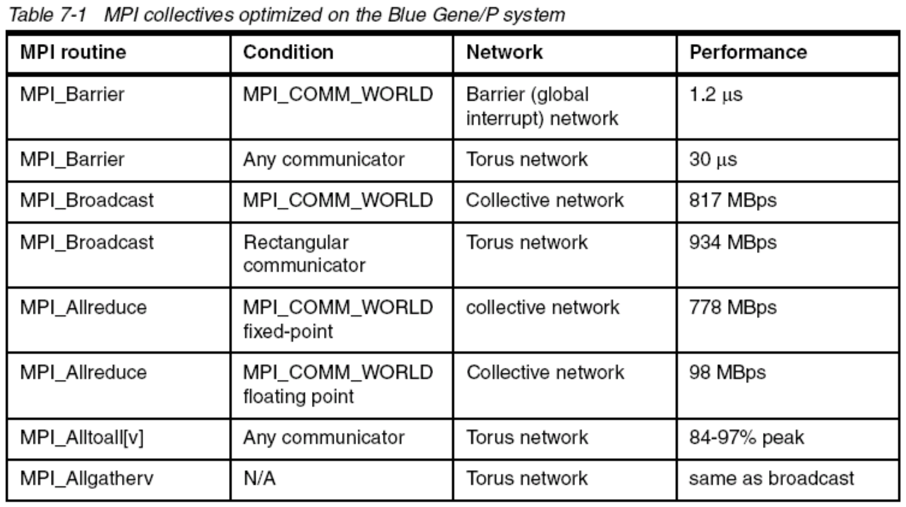
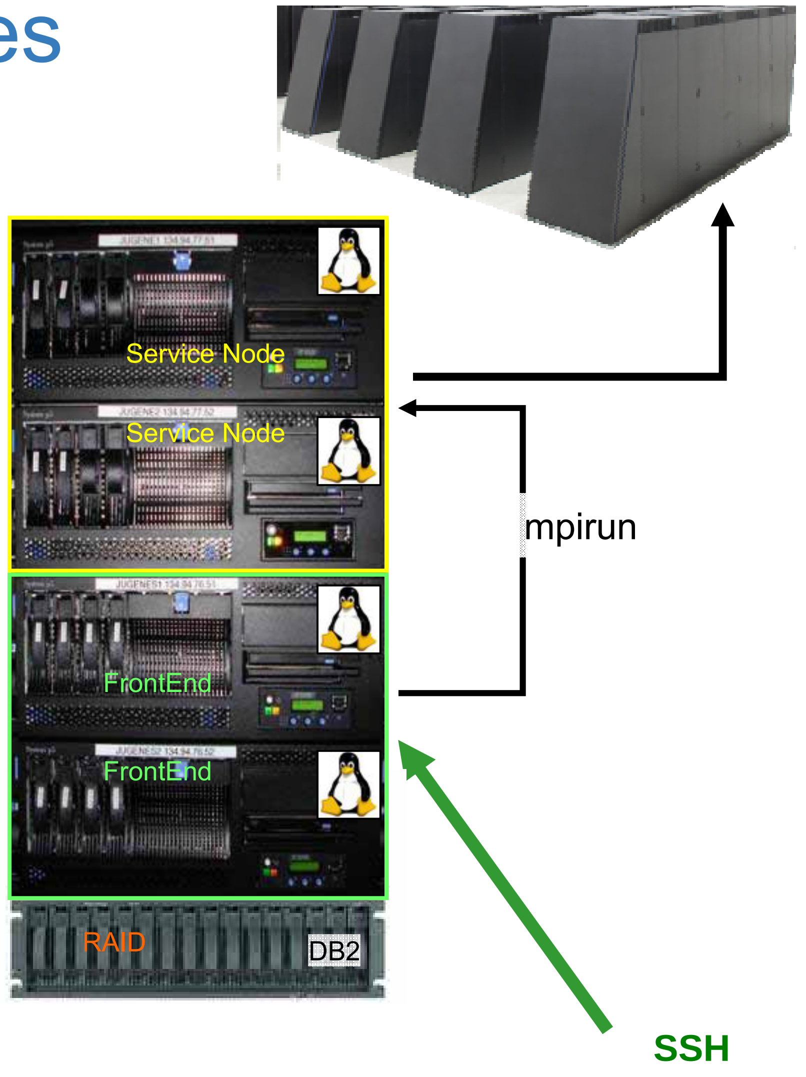


*Table 7-1 MPI collectives optimized on the Blue Gene/P system*

| MPI routine | Condition | Network | Performance |
|---|---|---|---|
| MPI_Barrier | MPI_COMM_WORLD | Barrier (global interrupt) network | 1.2 µs |
| MPI_Barrier | Any communicator | Torus network | 30 µs |
| MPI_Broadcast | MPI_COMM_WORLD | Collective network | 817 MBps |
| MPI_Broadcast | Rectangular communicator | Torus network | 934 MBps |
| MPI_Allreduce | MPI_COMM_WORLD fixed-point | collective network | 778 MBps |
| MPI_Allreduce | MPI_COMM_WORLD floating point | Collective network | 98 MBps |
| MPI_Alltoall[v] | Any communicator | Torus network | 84-97% peak |
| MPI_Allgatherv | N/A | Torus network | same as broadcast |

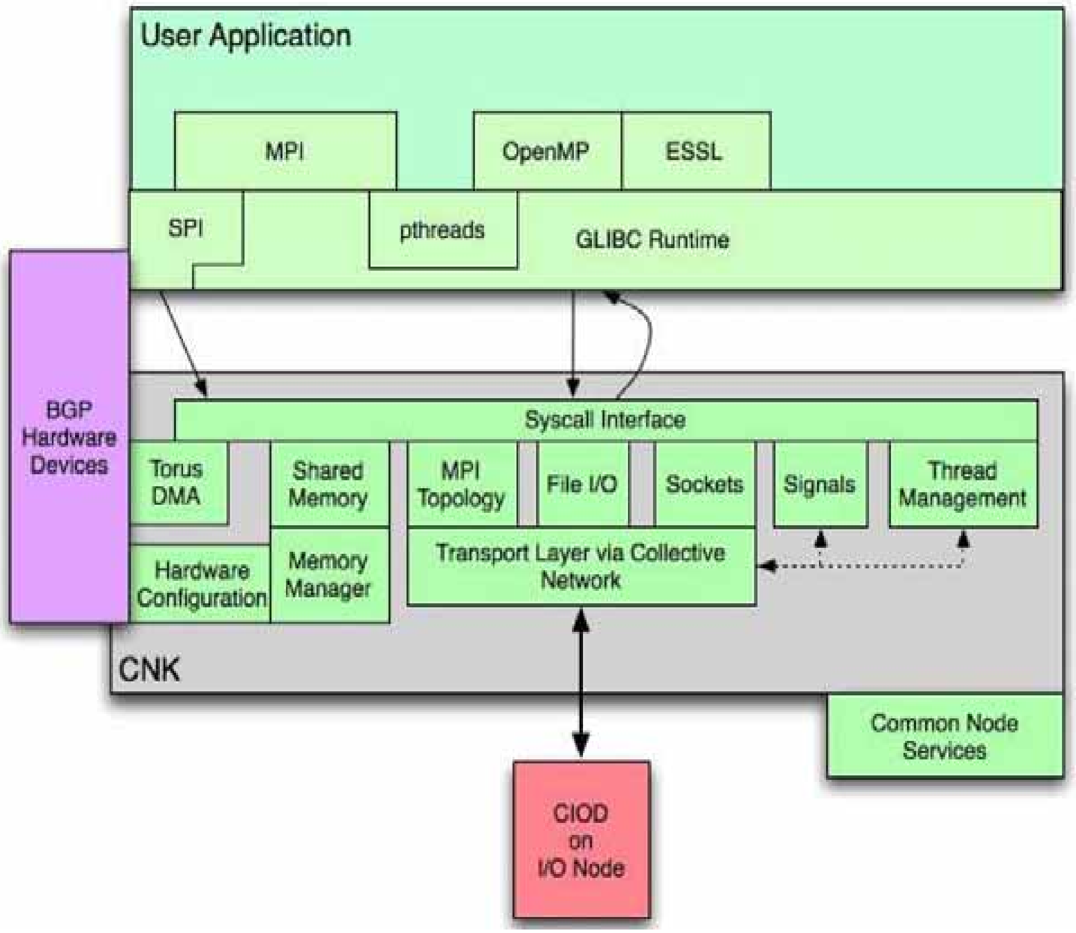
## Jugene: Login and Service nodes

- 2 Frontend Nodes
  - jugene.fz-juelich.de alias for jugene1 and jugene2
  - Linux (SLES10)
  - login nodes for compiling, editing, pre- and postprocessing
  - job submission (LoadLeveler)
  - IBM p5-55A, 8 x Power5+, 32GB memory
  - different processor → **cross compiling**
- 2 Service Nodes
  - IBM p5-55A, 8 x Power5+, 32GB memory
  - DB2 database managing BG/P system
  - local storage device DS4700 (1 TB)

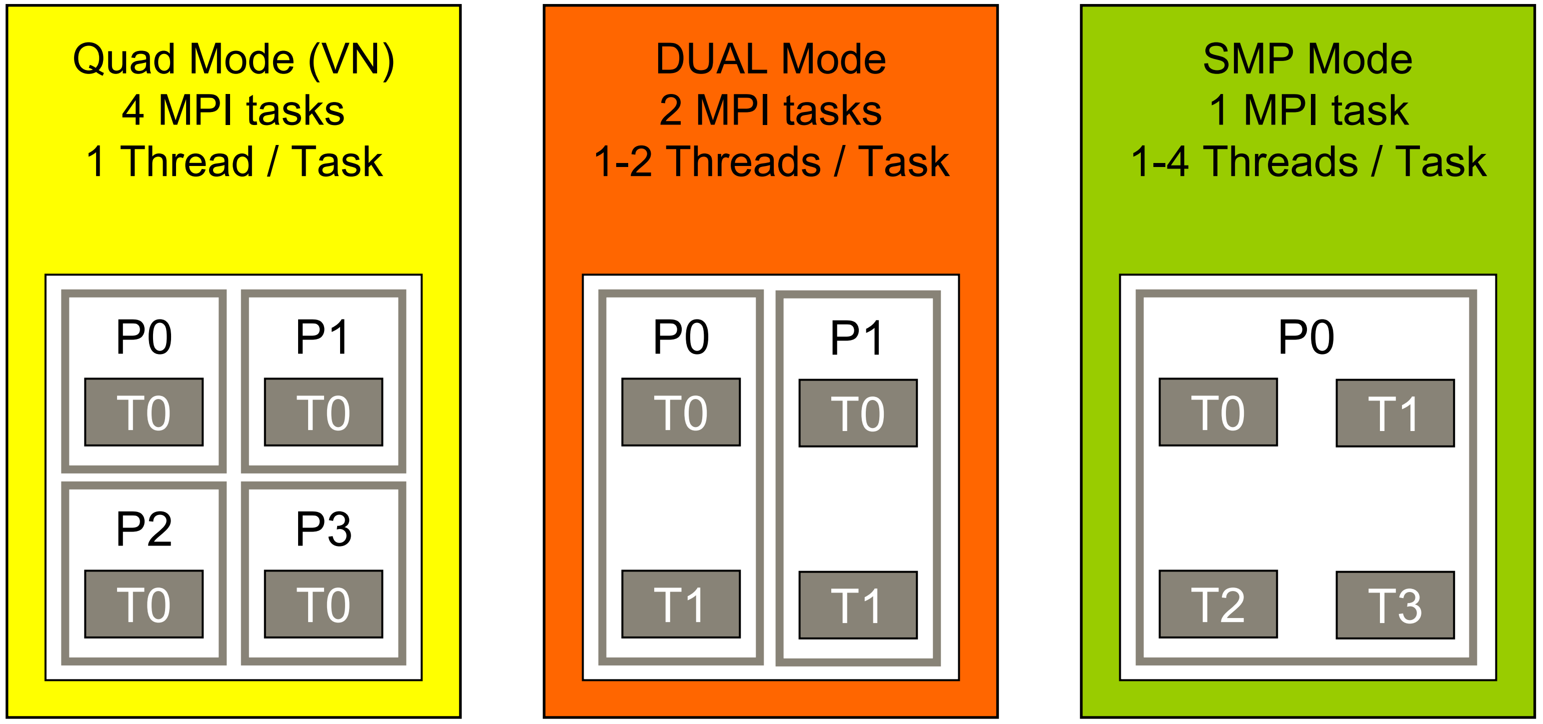


## Compute node kernel

- **CNK is a lightweight kernel and is NOT Linux**
- **Goals**
  - be as Linux compatible as possible
  - provide entire node's resources to the application…and get out of the way!
- **OS noise is minimal by design**
  - TLBs are by default statically mapped – no page faults
  - Only the user application is running – no system daemons
  - No source of *normal* interrupts except for:
    - timer interrupts as requested by the application
    - Torus DMA completion interrupts



## Jugene: Execution modes



`-mode vn` `-mode dual` `-mode smp`

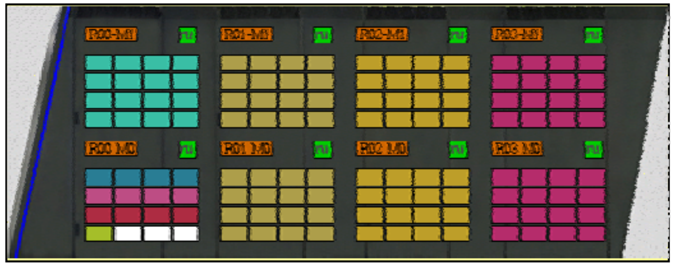
- Option of mpirun or llrun
- can be changed after booting partition (*different to BG/L*)

## Jugene: Partitions

- Subdivision of a Blue Gene/P system,
- Partitions are software defined and booted for each job
- Torus, Collective and Barrier networks are completely isolated
- A single job runs on a partition – i.e. jobs never share resources
- Small partitions:

| | | | | |
|---|---|---|---|---|
| – 1 x Node card | → | 32 cpu nodes | → | 128 cores |
| – 2 x Node card | → | 64 cpu nodes | → | 256 cores |
| – 4 x Node card | → | 128 cpu nodes | → | 512 cores |
| – 8 x Node card | → | 256 cpu nodes | → | 1024 cores |

→ only for testing on Jugene (walltime: 0.5h)



- Large partitions (production)

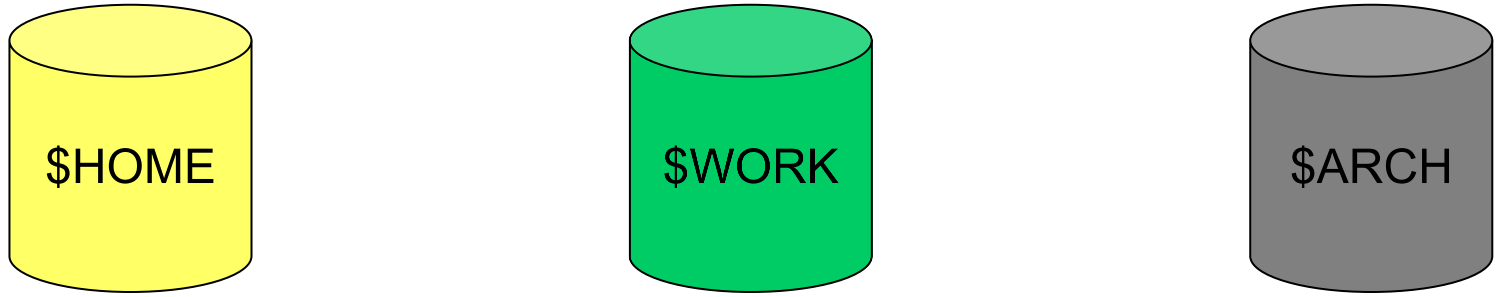
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| – Midplane | | | → | 16 Node cards | → | 512 cpu nodes | → | 2048 cores |
| – Rack | | | → | 32 Node cards | → | 1024 cpu nodes | → | 4096 cores |
| – Row/Column | → | 4 Racks | → | 128 Node cards | → | 4096 cpu nodes | → | 16384 cores |
| – Half system | → | 8 Racks | → | 256 Node cards | → | 8192 cpu nodes | → | 32768 cores |
| – Full system | → | 16 Racks | → | 512 Node cards | → | 16384 cpu nodes | → | 65536 cores |

## Jugene: Access

- SSH access to Jump → with password
- SSH access to Jugene → only with ssh keys
- Procedure:
  - **ssh-keygen -t [dsa|rsa]** → id_rsa.pub or id_dsa.pub
  - Login to Jump (class user id with password)
  - Append the contents of id_rsa.pub or id_dsa.pub to **$HOME/.ssh/authorized_keys** on **JUMP**
  - *Make sure there is no write access for group or world on the $HOME directory, otherwise .ssh does not work.*
  - ssh –x to Jugene
- Shell: bash
  - can be changed at the end of `$HOME/.profile`
- Common $HOME on Jump and Jugene

→ AFS and access to .ssh

## Filesystems



- permanent file system
- backup
- quota limits/group

- fast scratch file system (6 GB/s)
- no backup
- automatic file deletion after 4 weeks

- automatic migration of files to tape
- backup
- quota limits/group

Just: GPFS-Fileserver

## Compiling

- Two compiler available:
  - IBM XL compiler, Fortran 11.1, VACPP 9.0
    → `/opt/ibmcmp/xlf/11.1/bin`, `/opt/ibmcmp/vacpp/bg/9.0/bin`
  - GNU compiler suite 4.1.2
    → `/bgsys/drivers/ppcfloor/gnu-linux`
- Login node has different hardware architecture (Power 5)
  → **Cross Compiling**
- Standard compiler commands (xlc, xlC, gcc, …) generate executables for login node only!!!
- use Compiler Wrapper to compile for Blue Gene/P nodes
- IBM compiler
  - supports optimization for Blue Gene processor PPC 450
  - OpenMP
  - hybrid MPI+OpenMP

## Compiling: Wrappers for Cross-Compiling

- IBM VACPP

| | |
|---|---|
| standard | → **bgcc, bgc89, bgc99, bgxlc, bgxlC**, ... |
| MPI | → **mpixlc, mpixlcxx** |
| OpenMP | → **###_r –qsmp=omp –qthreaded** |

- IBM XL Fortran

| | |
|---|---|
| standard | → **bgf77/bgxlf, bgf90/bgxlf90, bgf2003/bgxlf2003**... |
| MPI | → **mpixlf77, mpixlf90, mpixlf2003**, … |
| OpenMP | → **###_r –qsmp=omp –qnosave –qthreaded** |

- GNU compiler

| | |
|---|---|
| MPI | → mpif77 |
| MPI | → mpicc, mpicxx |

# Compiling: key options for IBM compiler

- Debugging
  -g -qfullpath → store path to source files in executables
- Architecture
  - -qarch=**450** -qtune=450 → generates tuned code for PPC450 (BlueGene/P)
  - -qarch=**450d** -qtune=450 → generates also instruction for 2 floating point pipes (SIMD)
- Optimization

| | |
|---|---|
| -O2 | → basic optimization, instruction scheduling |
| -O3 -qstrict | → more aggressive optimization, that do not impact accuracy |
| -O3 –qhot | → aggressive optimization, that may impact the accuracy (high order transformations of loops) |
| -O3 –qhot=simd | → generate SIMD instructions |

  IPA (interprocedural optimizer)

| | |
|---|---|
| -O4 | → at compile time, limited scope analysis, SIMD instructions |
| -O5 | →at link time, whole-program analysis, SIMD instructions |

- OpenMP -qsmp=omp -qthreaded
- Optimization of codes: start with –qarch=450, then increase to –qarch=450d

## Compiling: Example Makefile

```
#      (c) COPYRIGHT IBM CORP. 2001, 2007 ALL RIGHTS RESERVED.     #
                LICENSED MATERIALS-PROPERTY OF IBM.                #
BGP_SYS = /bgsys/drivers/ppcfloor/comm

CC = $(BGP_SYS)/bin/mpixlc_r
FC = $(BGP_SYS)/bin/mpixlf95_r

# try -qarch=450 first, then use -qarch=450d for 2nd FPU later on
#  (SIMDization requires at least -O3)
# use -qlist -qsource with 450d and look for Parallel ASM instructions.

CFLAGS= -O3 -g -qmaxmem=-1 -I$(BGP_SYS)/include -L$(BGP_SYS)/lib -qarch=450 -qtune=450
FFLAGS= -O3 -g -qmaxmem=-1 -I$(BGP_SYS)/include -L$(BGP_SYS)/lib -qarch=450 -qtune=450
LDFLAGS = -g -Wl,-allow-multiple-definition

# LIBS_MPI = -lmpich.rts -lmsglayer.rts -lrts.rts -ldevices.rts
# LIBSF_MPI = -lmpich.rts -lfmpich.rts -lmsglayer.rts -lrts.rts -ldevices.rts
#
# ESSL    = -L/opt/ibmmath/essl/4.3/lib -lesslbg
# MASS    = -L/opt/ibmcmp/xlmass/bg/4.4/lib  -lmass -lmassv

LIBS =

default: helloworld.rts
helloworld.rts:   helloworld.o
          $(CC) $(CFLAGS) -o helloworld.rts helloworld.o $(LIBS)
helloworld.o:   helloworld.c Makefile
clean::
        rm -rf *.o  *~ *core*
        rm -rf *.rts
```

/bgsys/local/samples/helloworld

## module: Load additional packages

- run on login node to extent $PATH, …
- module

| | |
|---|---|
| avail | # show all available products |
| list | # list loaded products |
| load product(s) | # setup access to product |
| unload product(s) | # release access |
| swap product1 product2 | # replace product with older or newer version |
| whatis product(s) | # print short description |
| help product(s) | # print longer description |
| show product(s) | # show what “settings” are<br># performed for product |

## Jugene: Running interactive Jobs (FZJ utility)

- Special command **llrun** (wrapper of mpirun under LoadLeveler control)

```
    llrun [options]
    or
    llrun [options] binary [arg1 arg2 ... argn]
```

- Important options:

| | |
|---|---|
| -np <tasks> | → number of mpi tasks |
| -mode <SMP\|DUAL\|VN> | → execution mode |
| -exec <binary> | → executable |
| -args <"<arguments>"> | → arguments of executable |
| -o <filename> | → do not run/submit job but save to file |

- further help: `llrun -?`
- only for small jobs <= 256 cpu nodes, 0.5h walltime
- Example: `llrun -np 128 /bgsys/local/samples/helloworld/hello.rts`

## Jugene: Submitting Batch jobs with LoadLeveler (I)

- Required Keywords (Type & Size)

```
     #@ job_type = bluegene
     #@ bg_size  = <number of nodes>
     or
     #@ bg_shape = (XxYxZ) [midplanes in X,Y,Z direction or permutation]
```

- Optional keywords

```
#@ bg_connection = MESH | TORUS | PREFER_TORUS
#@ bg_rotate = True | False                              [to disable permutations]
```

- Submission

```
llsubmit  <cmdfile>
→ llsubmit: Batch class is chosen automatically at FZJ
  llsubmit: Processed command file through Submit Filter: …….
  llsubmit: The job ” jugene1.23497”has been submitted.
```

  – Job is processed by a FZJ submit filter, which associates a class name (and checks the cpu quota)

## Jugene: Submitting Batch jobs with LoadLeveler (II)

- General LL keywords:

```
      #@ wall_clock_limit = <HH:MM:SS>
      #@ notification = error | start | complete | always
      #@ notify_user  = <valid-email-address>
      #@ input = <some-filename>
      #@ output = <some-filename>
      #@ error  = <some-filename>
      #@ initialdir = <some-pathname>
      #@ environment = COPY_ALL
```

- Sample job script

```
# @ error = $(job_name).$(jobid).out
# @ output = $(job_name).$(jobid).out
# @ wall_clock_limit = 01:50:00
# @ job_type = bluegene
# @ bg_size = 512
# @ queue
mpirun -mode DUAL -verbose 1 -exe myprogram.rts  -args "456 99"
```

- Job examples in **/bgsys/local/samples/LoadL**

## LoadLeveler: Batch jobs handling

- Query status of a job/nodes/classes

| | |
|---|---|
| `llq -l [-x] jugene1.2347` | Detailed status of one job |
| `llq -s jugene1.2347` | Reason for wait status of job (Idle) → *……Not enough resources to start now…..* |
| `llstatus` | Status of nodes (included in llqall) |
| `llclass` | Active classes and free slots |

- Cancel jobs

| | |
|---|---|
| `llcancel jugene1.2347` | Kill this job |
| `llcancelall` | Kill all jobs of this user (be careful) |

- Full system view (Usage of Partition, nodes and job queues)

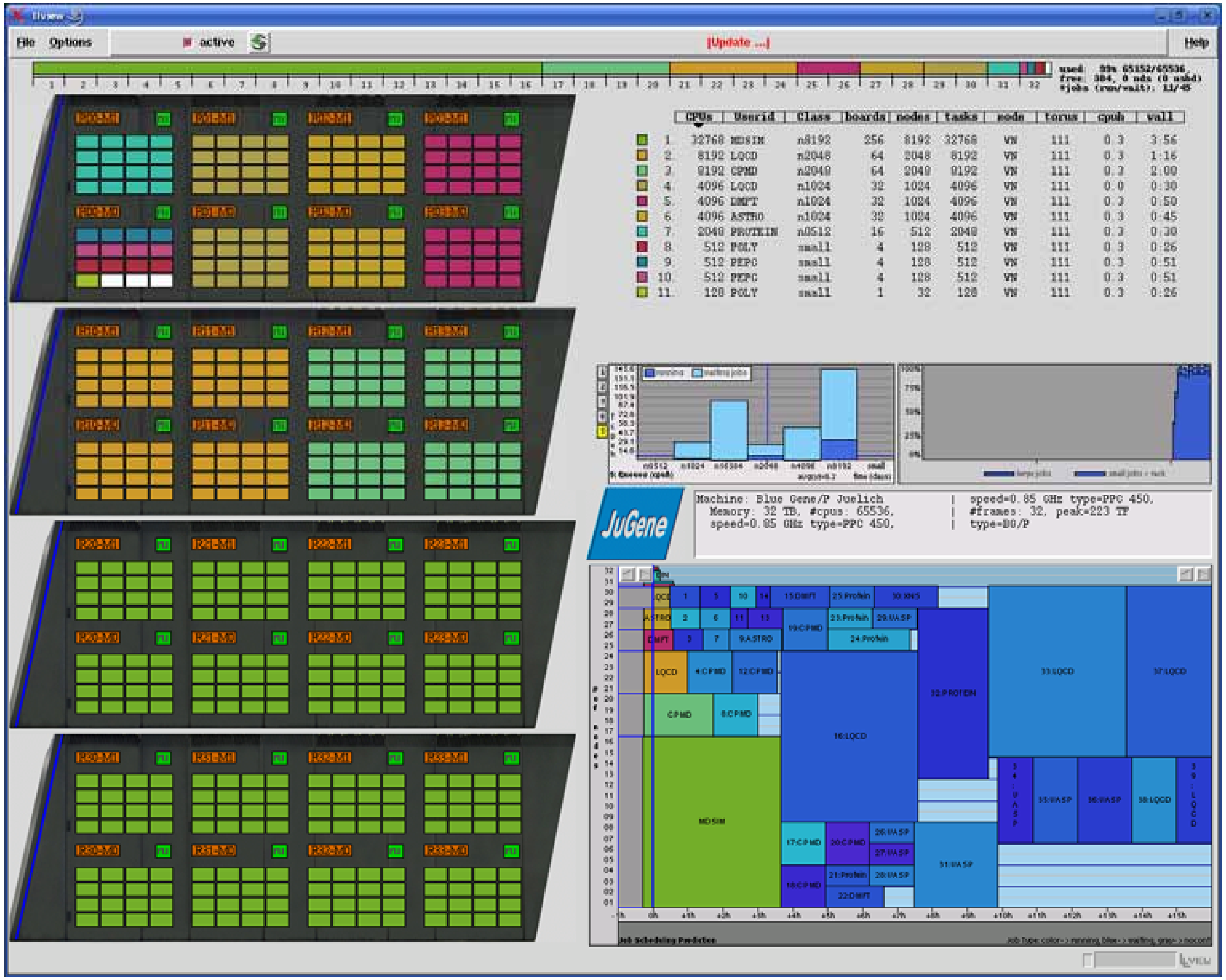
| | |
|---|---|
| `llstat` | text based monitoring of system status, including job and node usage |
| `llview` | graphical monitoring tool |

## LoadLeveler: Jobs Classes

| Class/Queue | Max. nodes | Wall Clock Limit | Priority |
|---|---|---|---|
| **n16384** | 16384 | 24h | On demand only |
| **n8192** | 8192 | 24h | ↑ |
| **n4096** | 4096 | 24h | |
| **n2048** | 2048 | 24h | |
| **n1024** | 1024 | 24h | |
| **n0512** | 512 | 24h | |
| **small** | 256 | 30 min | |
| **nocsmall** | 128 | 30 min | |
| **serial** | 0 | 60 min | jugeneX |

*class will be automatically set by LoadLeveler filter*

# LLview: monitoring batch system usage



```
used:  99% 65152/65536
free:  384, 0 nds (0 unhd)
#jobs (run/wait): 11/45
```

|  |  | CPUs | Userid | Class | boards | nodes | tasks | mode | torus | cpuh | wall |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 🟩 | 1 | 32768 | MDSIM | n8192 | 256 | 8192 | 32768 | VN | 111 | 0.3 | 3:56 |
| 🟧 | 2 | 8192 | LQCD | n2048 | 64 | 2048 | 8192 | VN | 111 | 0.3 | 1:16 |
| 🟩 | 3 | 8192 | CPMD | n2048 | 64 | 2048 | 8192 | VN | 111 | 0.3 | 2:00 |
| 🟨 | 4 | 4096 | LQCD | n1024 | 32 | 1024 | 4096 | VN | 111 | 0.0 | 0:30 |
| 🟥 | 5 | 4096 | DMFT | n1024 | 32 | 1024 | 4096 | VN | 111 | 0.3 | 0:50 |
| 🟧 | 6 | 4096 | ASTRO | n1024 | 32 | 1024 | 4096 | VN | 111 | 0.3 | 0:45 |
| 🟦 | 7 | 2048 | PROTEIN | n0512 | 16 | 512 | 2048 | VN | 111 | 0.3 | 0:30 |
| 🟥 | 8 | 512 | POLY | small | 4 | 128 | 512 | VN | 111 | 0.3 | 0:26 |
| 🟦 | 9 | 512 | PEPC | small | 4 | 128 | 512 | VN | 111 | 0.3 | 0:51 |
| 🟪 | 10 | 512 | PEPC | small | 4 | 128 | 512 | VN | 111 | 0.3 | 0:51 |
| 🟩 | 11 | 128 | POLY | small | 1 | 32 | 128 | VN | 111 | 0.3 | 0:26 |

```
Machine: Blue Gene/P Juelich              |   speed=0.85 GHz type=PPC 450,
  Memory: 32 TB, #cpus: 65536,            |   #frames: 32, peak=223 TF
  speed=0.85 GHz type=PPC 450,            |   type=BG/P
```

## Libraries on Blue Gene/P (I)

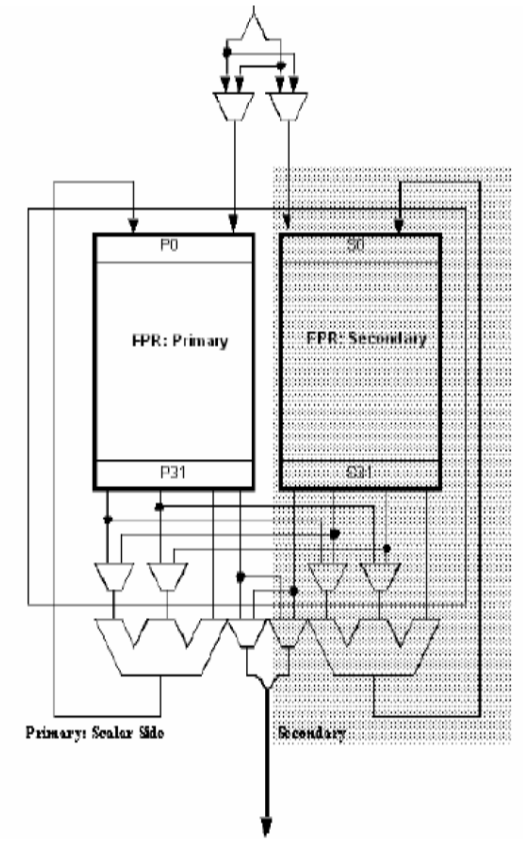
- Important: Use libraries, they are optimized for the hardware
  → the development work is already done
- Sequential Libraries:
  - **ESSL, LAPACK, ARPACK, GSL**
- Parallel Libraries and Application Systems:
  - Threaded Libraries
    - **ESSLsmp, FFTW**
  - MPI parallel Libraries:
    - **ScaLAPACK, FFTW, MUMPS, ParMetis, hypre**
    - **PARPACK, SPRNG, sundials, PETSc**
  - Chemistry Software
    - **CPMD, VASP** (special license needed)
    - **Gromacs**

## Libraries on Blue Gene/P (II)

- are installed in `/bgsys/local/`*`name`*
- most are also available with the module command
  - *module load name* sets environment variables for `–L$(*_LIB)` and `–I$(*_DIR)` to include in makefile
- Link sequence important, .o always before the libraries, sometimes double linking necessary
- libraries  are compiled with –qarch=450, to avoid misaligned variables calling programs must be compiled in the same way
- see also: `www.fz-juelich.de/sc/jugene/software/`

# SIMD Optimization

- Using the two FPU in parallel → compiler has to generate special Instructions
- SIMD: `-qarch=450d –qtune=450 –O2` (or up)
- Information: `-qsource –qreport –qlist`
- Data has to be aligned on 16 bytes
- special directive: `__alignx` (C), `ALIGNX` (Fortran)
  → tells compiler that is aligned correctly
- SIMD of loops only if memory access with stride 1
- Intrinsic built-in function of compiler
- Further information:
  - Red Book: BG/P Appl. Development, p. 105ff
  - Exploiting the Dual Floating Point Units in Blue Gene/L, Whitepaper, March 2006
  - Appendix B “Built-in Functions in XL C/C++ Compiler Reference
  - “Intrinsic Procedures” in XL Fortran Language Reference
    → Jugene documentation WWW page



## MPI: Mapping tasks to torus, BG/P MPI extensions

**Mapping:**

- mpirun Option `–env “BG_MAPING=XYZT”` (default)
- X,Y,Z are torus coordinates, T is the core number in a cpu node
- possible values: XYZT, XZYT, YZXT, YXZT, ZXYT, ZYXT, TXYZ, TXZY, TYZX, TYXZ, TZXY, TZYX
- optimal mapping depends on application and communication pattern, e.g.
- for 1-dimensional problems with nearest-neighbor comm. → TXYZ
- see Red Book: BG/P Appl. Development, p. 303ff

**BG/P MPI communicators:**

- MPIX_Cart_comm_create(…)
  → 4D Cartesian communicator (= hardware torus)
- MPIX_Pset_same_comm_create(…)
  → set of comm., all tasks of a I/O-node are in the same communicator
- MPIX_Pset_diff_comm_create(…)
  → all tasks of one I/O-node are in different communicators
- see Red Book: BG/P Appl. Development, p. 75ff

## Debugging: Core files

- on Blue Gene core files are ASCII files containing few information:
  - segmentation fault number (reason)
  - function call chain, …
  - registers, …
- Command addr2line to analyze core files:

  addr2line [options] –e *executable* < core.##
  - shows function call chain in human readable format (file,line)

## Debugging: TotalView

- Compile your program with common debug options

  ```
  … -g -qfullpath myprog.f
  … -g -qfullpath myprog.c
  … -g -qfullpath myprog.cpp
  ```

- Start your program under the control of totalview

  ```
  llrun … -tv a.out arg1 …
  ```

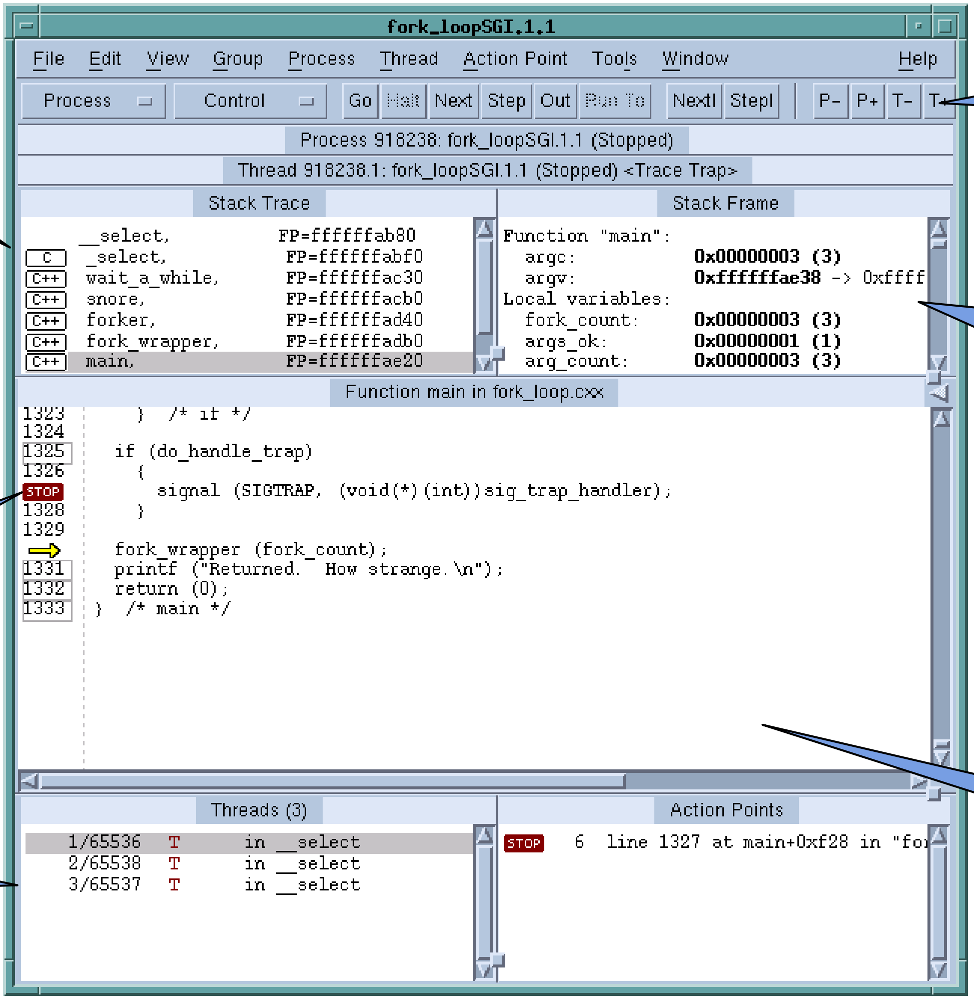
- Starting totalview with a core file

  ```
  totalview a.out core
  ```

- Availability on JuGene:

  Part of the *Scalability Research Program of* TotalView Technologies

# Debugging: TotalView


Stack trace
Toolbar for common options
Local variables for selected Stack frame1
Break points
Active threads
Source code window
fork_loopSGI.1.1
File Edit View Group Process Thread Action Point Tools Window Help
Process Control Go Halt Next Step Out Run To NextI StepI P- P+ T- T+
Stack Trace
Stack Frame
Threads (3)
Action Points

```
Process 918238: fork_loopSGI.1.1 (Stopped)
Thread 918238.1: fork_loopSGI.1.1 (Stopped) <Trace Trap>
```

```
     __select,             FP=ffffffab80
C    _select,              FP=ffffffabf0
C++  wait_a_while,         FP=ffffffac30
C++  snore,                FP=ffffffacb0
C++  forker,               FP=ffffffad40
C++  fork_wrapper,         FP=ffffffadb0
C++  main,                 FP=ffffffae20
```

```
Function "main":
  argc:            0x00000003 (3)
  argv:            0xffffffae38 -> 0xffff
Local variables:
  fork_count:      0x00000003 (3)
  args_ok:         0x00000001 (1)
  arg_count:       0x00000003 (3)
```

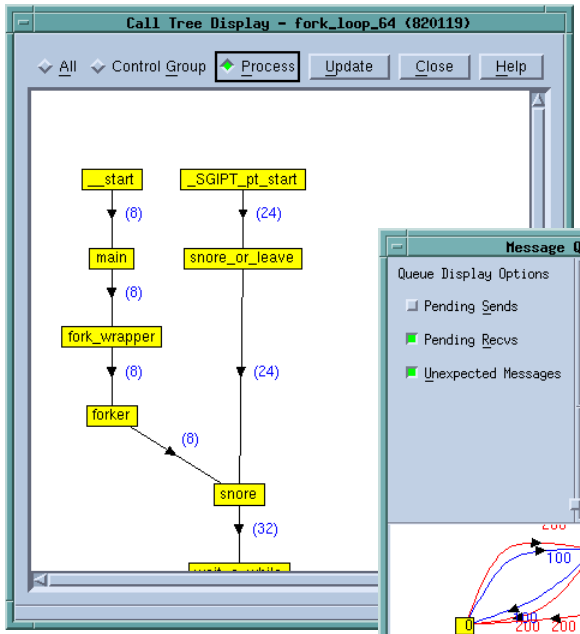
```
Function main in fork_loop.cxx
      } /* if */

   if (do_handle_trap)
      {
        signal (SIGTRAP, (void(*)(int))sig_trap_handler);
      }

   fork_wrapper (fork_count);
   printf ("Returned.  How strange.\n");
   return (0);
 } /* main */
```

```
1/65536  T    in __select
2/65538  T    in __select
3/65537  T    in __select
```

```
STOP   6   line 1327 at main+0xf28 in "fo
```

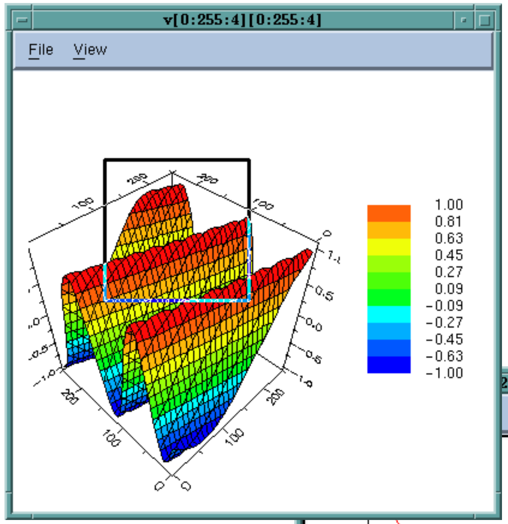
## Debugging: TotalView

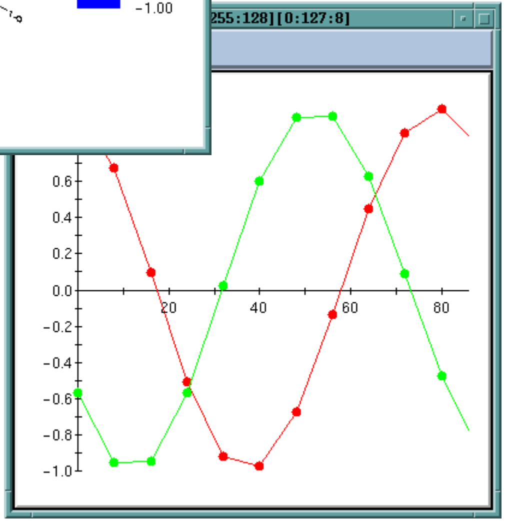
TOTALVIEW TECHNOLOGIES

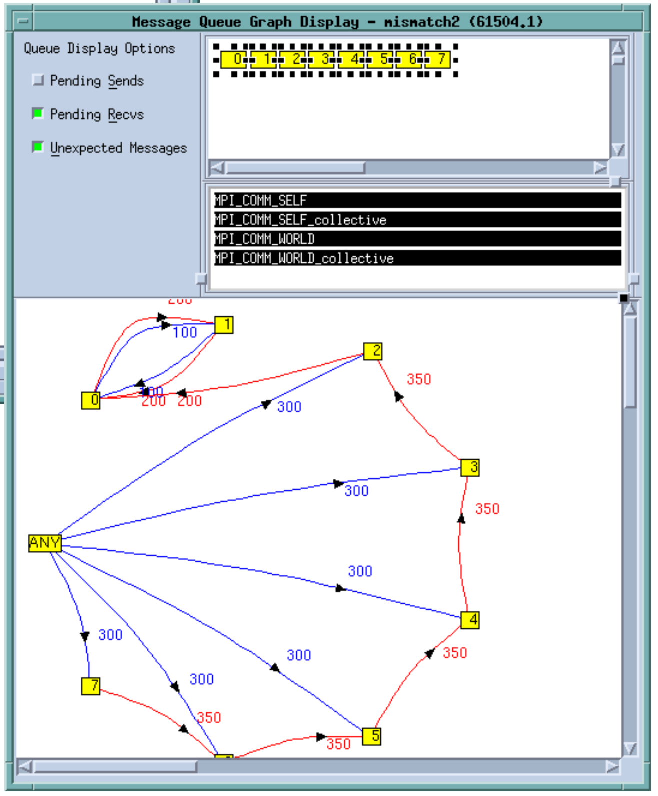


Call graph

Data visualization





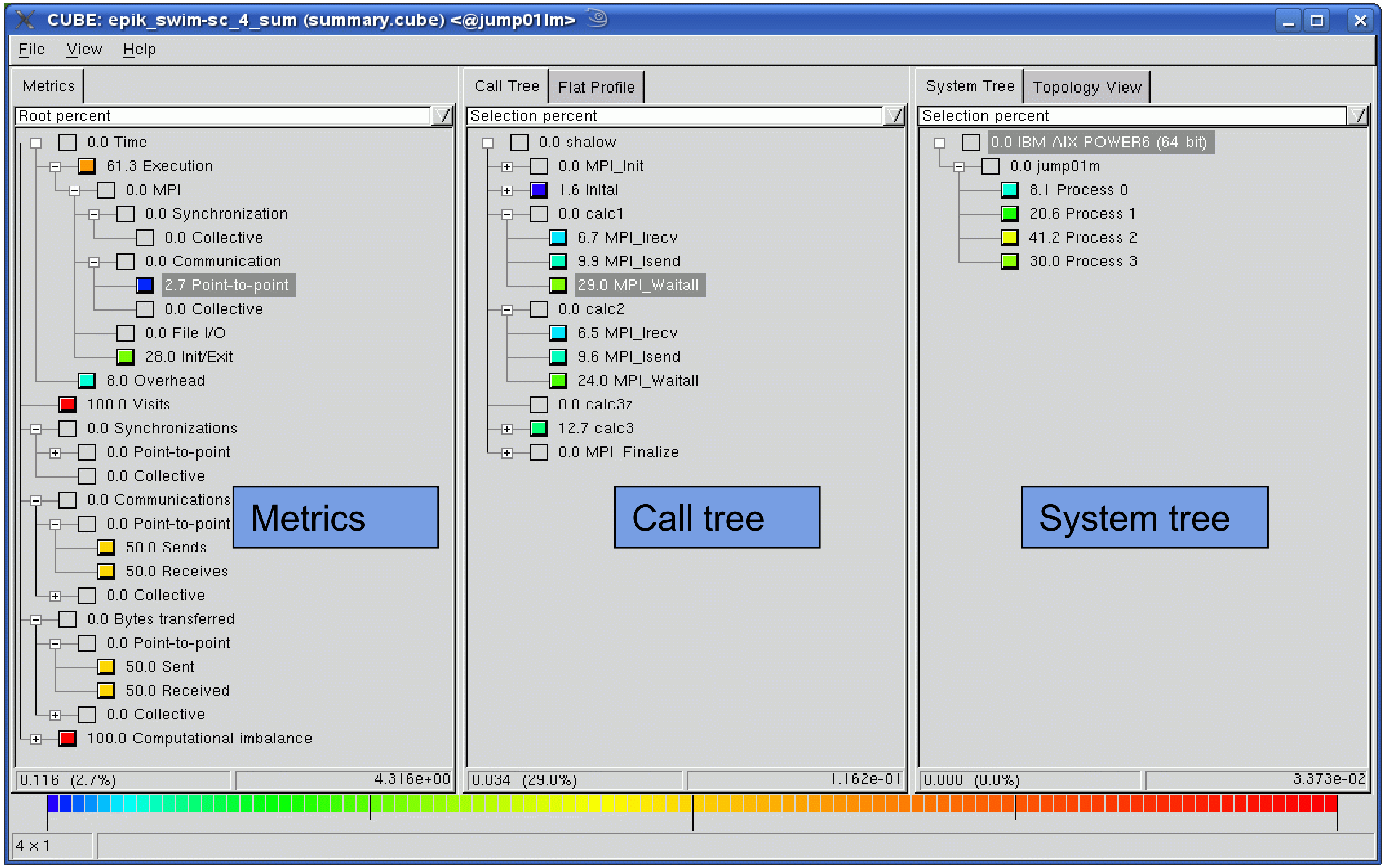


Message queue graph

## Tools: Performance Analysis with scalasca

- open source toolset for scalable performance analysis of large-scale parallel applications
- WWW: http://www.scalasca.org
- Scalasca Quick Reference Guide:
  $SCALASCA_ROOT/doc/manuals/quickref.pdf
- Usage:
  - `module load scalasca` #just once
  - Prefix compile and link commands with Scalasca instrumenter
    - `skin mpcc/mpxlf/...`
  - Prefix execute command with Scalasca analyzer
    - `scan llrun` ...
  - Browse results with
    - `square epik_...`

# Scalasca Result Browser



```
CUBE: epik_swim-sc_4_sum (summary.cube) <@jump01lm>
File  View  Help

Metrics                          | Call Tree  Flat Profile        | System Tree  Topology View
Root percent                     | Selection percent              | Selection percent
0.0 Time                         | 0.0 shallow                    | 0.0 IBM AIX POWER6 (64-bit)
  61.3 Execution                 |   0.0 MPI_Init                 |   0.0 jump01m
    0.0 MPI                      |   1.6 inital                   |     8.1 Process 0
      0.0 Synchronization        |   0.0 calc1                    |     20.6 Process 1
        0.0 Collective           |     6.7 MPI_Irecv              |     41.2 Process 2
      0.0 Communication          |     9.9 MPI_Isend              |     30.0 Process 3
        2.7 Point-to-point       |     29.0 MPI_Waitall           |
        0.0 Collective           |   0.0 calc2                    |
      0.0 File I/O               |     6.5 MPI_Irecv              |
      28.0 Init/Exit             |     9.6 MPI_Isend              |
  8.0 Overhead                   |     24.0 MPI_Waitall           |
100.0 Visits                     |   0.0 calc3z                   |
0.0 Synchronizations             |   12.7 calc3                   |
  0.0 Point-to-point             |   0.0 MPI_Finalize             |
  0.0 Collective                 |                                |
0.0 Communications               |                                |
  0.0 Point-to-point             |                                |
    50.0 Sends                   |                                |
    50.0 Receives                |                                |
  0.0 Collective                 |                                |
0.0 Bytes transferred            |                                |
  0.0 Point-to-point             |                                |
    50.0 Sent                    |                                |
    50.0 Received                |                                |
  0.0 Collective                 |                                |
100.0 Computational imbalance    |                                |
0.116  (2.7%)        4.316e+00   | 0.034  (29.0%)     1.162e-01   | 0.000  (0.0%)      3.373e-02
4 x 1
```

## Jugene: Further Information

- WWW: `http://www.fz-juelich.de/jsc/jugene`
- Documentation: `http://www.fz-juelich.de/jsc/jugene/documentation`

  - Articles
    - Overview of the IBM Blue Gene/P project
    - Exploiting the Dual Floating Point Units in Blue Gene/L, Whitepaper, March 2006

  - Red Books
    - Blue Gene/P Application Development, Redbook, July 2008

## Appendix

→ Exercises

→ Available Libraries on Jugene

## Exercises on Tuesday, Wednesday

- Port, run and optimize your own program
  1. → Single core performance
  2. → Speedup

- Exercises
  1. Single core Optimization, SIMD & Compiler-Flags
  2. Performance measurement with HPM and Scalasca (swim3d)
  3. Running a MPI Benchmark
  4. Running an application (PEPC)

```
Location: /home1/zam/kurs55/EXERCISE_i/ ...
                                          → Exercisei.txt
```

PARTNERSHIP
FOR ADVANCED COMPUTING
IN EUROPE

## Blue Gene/P: Sequential Libraries, ESSL

- *IBM Engineering and Scientific Subroutine Library (ESSL) is a state-of-the-art collection of high-performance subroutines providing a wide range of mathematical functions for many different scientific and engineering applications. Its primary characteristics are performance, functional capability, and usability*
- library function can be called from FORTRAN, C , and C++
- Arrays are FORTRAN → column-first
- Header file `essl.h` for C and C++

  Installed in `/bgsys/local/lib` on JUGENE (not as module)
- -qessl → automatic replacement of vector operations with ESSL calls

```
mpixlf90_r name.f –L/bgsys/local/lib –lesslbg [-qessl]
```

```
mpixlc_r   name.c –L/bgsys/local/lib –lesslbg –lm [-qessl]
           -L/opt/ibmcmp/xlf/bg/11.1/lib –lxl -lxlopt –lxlf90_r
           –lxlfmath -L/opt/ibmcmp/xlsmp/bg/1.7/lib –lxlomp_ser
           -lpthread
```

## Blue Gene/P: Sequential Libraries, LAPACK

- *Library of subroutines for solving dense linear algebra problems efficiently on high-performance computers. Performance issues are addressed by implementing a large number of algorithms in terms of the level 2 and 3 BLAS and by incorporating recent algorithmic improvements for linear algebra computation. BLAS routines have been optimized for single and multiple-processor environments, these algorithms give nearly optimal performance.*
- Public domain version 3.1
- Must be used together with ESSL, Some routines are already in ESSL, but attention: some calling sequences are different!

```
module load lapack
mpixlf77_r –qarch=450 –qtune=450 name.f
           -L/bgsys/local/lib
          [-lesslbg] -L$(LAPACK_LIB) –llapack –lesslbg
```

- All routines in ESSL are taken from ESSL
- ESSL must be linked after LAPACK to resolve references

## Blue Gene/P: Sequential Libraries, ARPACK

- *ARnoldi PACKage, Version 2.1*
- *ARPACK is a collection of Fortran77 subroutines designed to solve large scale sparse eigenvalue problems, Important Features: Reverse communication interface, Single and double precision real/complex arithmetic versions for symmetric, non-symmetric, standard or generalized problems, Routines for banded matrices, …*
- FORTRAN 77 library
- calls LAPACK and BLAS routines

```
module load arpack
mpixlf77_r  –qarch=450 –qtune=450 name.f
            -L$(ARPACK_LIB) –larpack –L$(LAPACK_LIB) –llapack
            –L/bgsys/local/lib -lesslbg
```

## Blue Gene/P: Sequential Libraries, GSL

- GNU Scientific Library, Version 1.1
- *The GNU Scientific Library (GSL) is a numerical library for C and C++ programmers. It is free software under the GNU General Public License.*
- *The library provides a wide range of mathematical routines such as random number generators, special functions and least-squares fitting. There are over 1000 functions in total with an extensive test suite*
- Not recommended for performance reasons, on JUGENE use esslbg wherever possible

```
module load gsl
      → GSL_DIR=bgsys/local/gsl
```

## Blue Gene/P: Parallel Libraries, ScaLAPACK

- *The ScaLAPACK library includes a subset of LAPACK routines redesigned for distributed memory MIMD parallel computers. It is currently written in a Single-Program-Multiple-Data style using explicit message passing for interprocessor communication. It assumes matrices are laid out in a two-dimensional block cyclic decomposition.*
- *Contents: Parallel BLAS 1-3, PBLAS Version 2, Dense linear system solvers, Banded linear system solvers, Solvers for Linear Least Squares Problem, Singular value decomposition, Eigenvalues and eigenvectors of dense symmetric/hermitian matrices*
- Release 1.8 public domain library, together with BLACS v1.1

```
module load scalapack

mpixlf77 name.f -o name -L$(SCALAPACK_LIB) -lscalapack
                        -L$(BLACS_LIB) –lblacsF77init
                        –lblacs –lblacsF77init
                        -L$(LAPACK_LIB) -llapack
                        -L/bgsys/local/lib –lesslbg
```

## Blue Gene/P: Parallel Libraries, FFTW

- *FFTW (Fastest Fourier Transform in the West) is a efficient, multi-threaded C subroutine library with fortran interface for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST).*
- FFTW version 2.1.5 also includes parallel transforms for both shared- and distributed-memory systems.
- FFTW version 3.1.2 doesn't include parallel transforms for distributed-memory systems.
- threaded version  FFTW 3.1.2 has been installed under /bgsys/local/fftw3
- old MPI version    FFTW 2.1.5 has been installed under /bgsys/local/fftw2

## Blue Gene/P: Parallel Libraries, PARPACK

- *PARPACK, a part of the ScaLAPACK project, is an extension of the ARPACK software package used for solving large scale eigenvalue problems on distributed memory parallel architectures. The message passing layers currently supported are BLACS and MPI.*
- Reverse communication interface, user has to supply parallel matrix-vector multiplication
- ARPACK Version 2.1

```
module load arpack

mpixlf90_r name.f -lparpack –larpack –llapack –lesslbg
```

## Blue Gene/P: Parallel Libraries, MUMPS

- *MUMPS (MUltifrontal Massively Parallel Solver) is a package for solving linear systems of equations Ax=b, where the matrix A is sparse and can be either unsymmetric, symmetric positive definite, or general symmetric. MUMPS uses a multifrontal technique which is a direct method based on either the L U or the L D transpose(L) factorization of the matrix. The software requires MPI for message passing and makes use of BLAS, BLACS, and ScaLAPACK subroutines.*
- Real or Complex supported
- Version 4.7.3 on Jugene
- installed in /bgsys/local/mumps →

| | |
|---|---|
| – libsmumps.a | single precision real |
| – libdmumps.a | double precision real |
| – libcmumps.a | single precision complex |
| – libzmumps.a | double precision complex |

## Blue Gene/P: Parallel Libraries, SPRNG, Sundials

- SPRNG: The Scalable Parallel Random Number Generators Library for ASCI Monte Carlo Computations Version 2.0: various random number generators in one Library
- SPRNG on JUGENE is installed in /bgsys/local/sprng
- available in an optimised (-O3 -qstrict) and a non-optimised (-O0 -g) version
- sequential and parallel routines, C and FORTRAN interface

- Sundials: Package for the solution of ordinary differential equations
  - CVODE solves initial value problems for ordinary differential equation (ODE) systems
  - CVODES, solves ODE systems and includes sensitivity analysis capabilities (forward and adjoint)
  - IDA, solves initial value problems for differential-algebraic equation (DAE) systems
  - KINSOL, solves nonlinear algebraic systems
- Installed in /usr/local/beta/sundials-2.3.0

## Blue Gene/P: Parallel Libraries, PETSc

- *PETSc is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It employs the MPI standard for all message-passing communication.*
- It is configured to be used together with hypre, ParMetis, sundials, and ScaLAPACK.
- Version 2.3.3 installed on JUGENE
- Examples in: `/bgsys/local/PETSc/src/vec/vec/examples/tutorials/`

```
module load petc
Makefile: include  ${PETSC_DIR}/bmake/common/base
```

## Blue Gene/P: Parallel Libraries, ParMETIS, Hypre

- *ParMETIS (3.1) is an MPI-based parallel library that implements a variety of algorithms for partitioning unstructured graphs, meshes, and for computing fill-reducing orderings of sparse matrices. ParMETIS extends the functionality provided by METIS and includes routines that are especially suited for parallel AMR computations and large scale numerical simulations. The algorithms implemented in ParMETIS are based on the parallel multilevel k-way graph-partitioning, adaptive repartitioning, and parallel multi-constrained partitioning schemes developed in Karypis Lab*.

```
mpixlc_r   -c -I/bgsys/local/include   main.c
mpixlc_r   main.o   -L/bgsys/local/lib   -lparmetis  -lmetis  -lm
```

- hypre: library of high performance preconditioners that features parallel multigrid methods for both structured and unstructured grid problems.
- Version 2.0.0 on JUGENE

```
mpixlc_r -c -I/bgsys/local/hypre/include   main.c
mpixlf90_r main.o -L/bgsys/local/lib -L/bgsys/local/hypre/lib
                  -lHYPRE -lg2c -lm-llapack -lesslbg  -lm
```