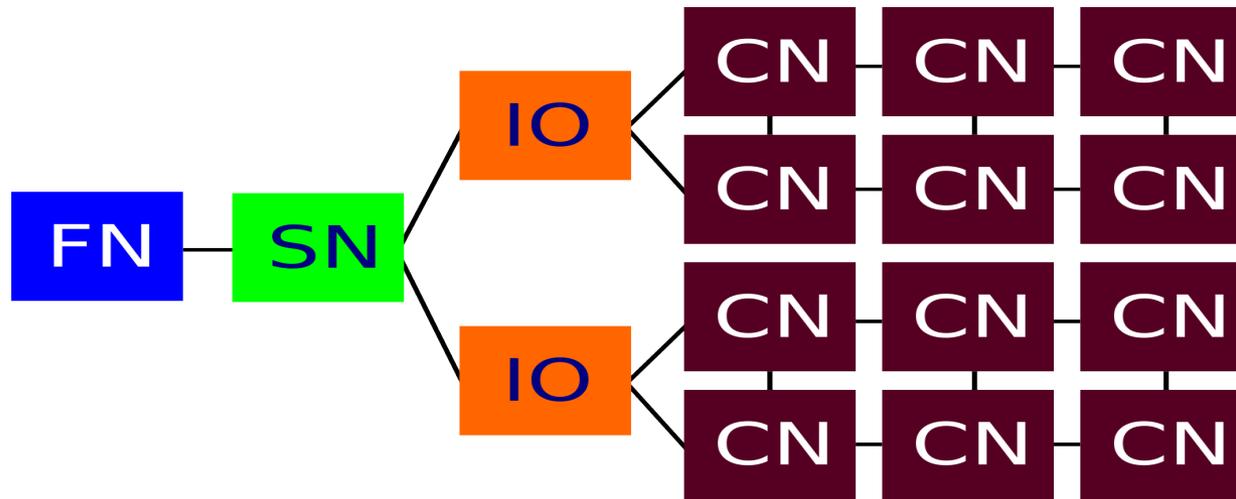


Jeff Hammond, Charles Bacon

Argonne Leadership Computing Facility

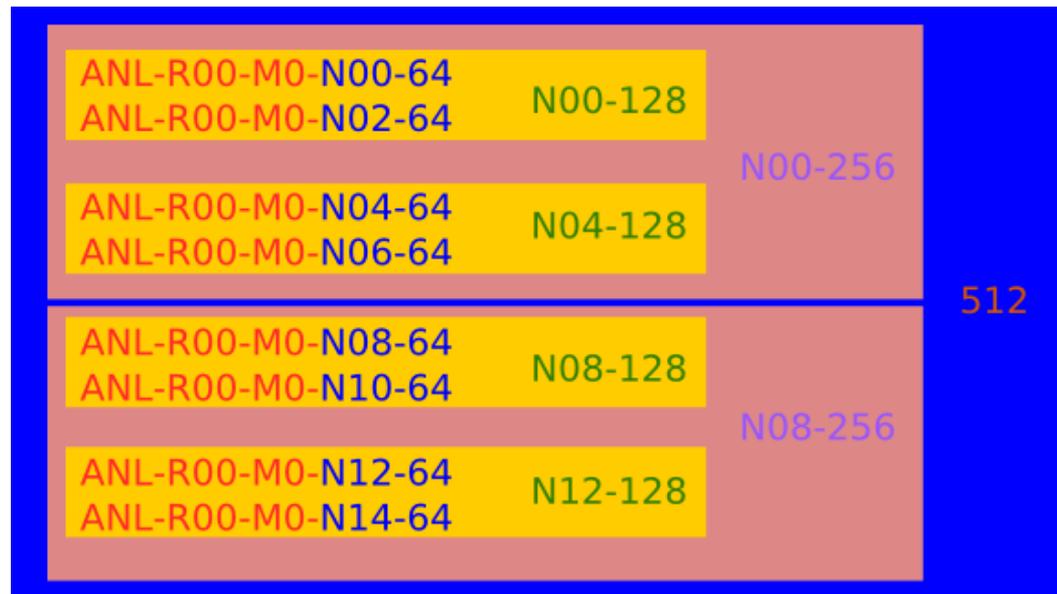
Blue Gene/P System Overview



- **Front-end nodes (FN):** dedicated for user's to login, compile programs, submit jobs, query job status, debug applications, 2.5 GHz PowerPC 970, Linux OS
- **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, 1 I/O node per 64 compute nodes
- **Compute nodes (CN):** run user application, accessed through qsub, no shell, limited OS services, quad core 850 MHz PowerPC 450, CNK OS

Blue Gene/P Compute Node Partitions

- Blue Gene compute nodes are grouped into partitions:
 - Each partition is isolated from the others, partitions only share the file system
 - Partition sizes on Argonne systems:
 - 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 40960
 - Minimal partition size is 64 nodes due to one I/O 64 compute node ratio
 - Jobs run in the smallest partition into which they fit
 - Partitions are nested- larger partitions contain smaller ones
 - If a job is running on a partition, no other job can run on the enclosing larger partitions
 - `bg-listblocks --all` lists all defined partitions

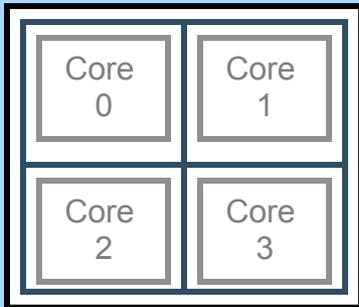


Execution Modes in BG/P

VN Mode

0-4 Processes

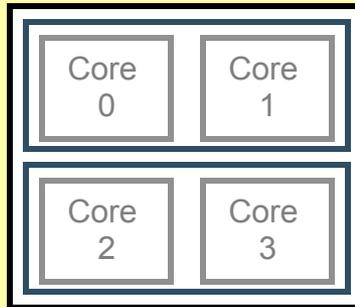
1 Thread/Process



Dual Mode

0-2 Processes

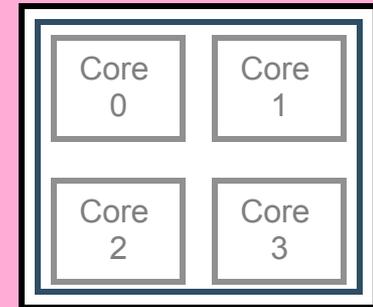
1-2 Threads/Process



SMP Mode

0-1 Process

1-4 Threads/Process



Blue Gene/P Programming Environment

- **Linux cross-compilation environment:**
 - users login to FEN for compilation, job submission, debugging
- **Space sharing:**
 - exactly one job per partition
 - smp-mode, one MPI task/node, 4 threads/task, 2GB of RAM
 - dual-mode, two MPI tasks/node, 2 threads/task, 1GB of RAM
 - vn-mode, 4 single-threaded MPI tasks/node, 512MB of RAM
- **SPMD model:**
 - compute nodes run the same executable
- **Applications:**
 - memory limited to physical memory
 - statically and dynamically linked libraries
 - restricted set of POSIX routines (no fork, system, ...)
 - threading support
 - message passing using MPI based on ANL's mpich2



Compilers

- Supported Languages:
 - Fortran (77, 90, 95, 2003)
 - C (C89, C90)
 - C++
 - Python
- Compiler Families:
 - IBM XL
 - GNU
- Target Architectures:
 - Back End: BG/P-PowerPC with CNK OS
 - Front End: Standard-PowerPC with SUSE Linux OS
- There are four compiler flavors on the BG/P
 - IBM for compute nodes [PowerPC450, CNK] (mpixlc, bgxlf95, ...)
 - IBM for login nodes [PowerPC970, SUSE] (xlc, xlf90, ...)
 - GNU for compute nodes [PowerPC450, CNK] (mpicc, powerpc-bgp-linux-gcc, ...)
 - GNU for login nodes [PowerPC970, SUSE] (gcc, ...)
- Tips:
 - IBM compilers link with thread safe libraries when invoked with appended “_r” (ex: mpixlc_r)
 - Compilers for the back-end compute nodes have “bg” or “mpi” in their name
 - It is strongly recommended to compile using the “mpixl...” wrappers

Many Compiler Names

bgcc, bgcc_r	IBM BG compiler for pre-ANSI C non-standard source file
bgc89, bgc89_r	IBM BG compiler for C89-conformed C source file
bgc99, bgc99_r	IBM BG compiler for C99-conformed C source file
bgxlc, bgxlc_r	IBM BG compiler for C source file
bgxlc++, bgxlc++_r, bgxlc, bgxlc_r	IBM BG compiler for C++ source file
bgxlf, bgxlf_r, bgf77, bgfort77	IBM BG compiler for Fortran 77 source file
bgxlf90, bgxlf90_r, bgf90	IBM BG compiler for Fortran 90 source file
bgxlf95, bgxlf95_r, bgf95	IBM BG compiler for Fortran 95 source file
bgxlf2003, bgxlf2003_r, bgf2003	IBM BG compiler for Fortran 2003 source file
powerpc-bgp-linux-gcc	GNU BG compiler for C (in /bgsys/drivers/ppcfloor/gnu-linux/bin)
powerpc-bgp-linux-g++	GNU BG compiler for C++ (in /bgsys/drivers/ppcfloor/gnu-linux/bin)
powerpc-bgp-linux-gfortran	GNU BG compiler for Fortran (in /bgsys/drivers/ppcfloor/gnu-linux/bin)
cc, cc_r	IBM <i>non</i> -BG compiler for pre-ANSI C non-standard source file
c89, c89_r	IBM <i>non</i> -BG compiler for C89-conformed C source file
c99, c99_r	IBM <i>non</i> -BG compiler for C99-conformed C source file
xlc, xlc_r	IBM <i>non</i> -BG compiler for C source file
xlc++, xlc++_r, xlc, xlc_r	IBM <i>non</i> -BG compiler for C++ source file
xlf, xlf_r, f77, fort77	IBM <i>non</i> -BG compiler for Fortran 77 source file
xlf90, xlf90_r, f90	IBM <i>non</i> -BG compiler for Fortran 90 source file
xlf95, xlf95_r, f95	IBM <i>non</i> -BG compiler for Fortran 95 source file
xlf2003, xlf2003_r, f2003	IBM <i>non</i> -BG compiler for Fortran 2003 source file
gcc	GNU <i>non</i> -BG compiler for C
g++	GNU <i>non</i> -BG compiler for C++
gfortran	GNU <i>non</i> -BG compiler for Fortran



MPI Compiler Wrappers

■ MPI wrappers for IBM compilers:

Wrapper	Thread-Safe	Compiler	Description
mpixlc	mpixlc_r	bgxlc	IBM BG C Compiler
mpixlcxx	mpixlcxx_r	bgxlc	IBM BG C++ Compiler
mpixlf77	mpixlf77_r	bgxlf	IBM BG Fortran 77 Compiler
mpixlf90	mpixlf90_r	bgxlf90	IBM BG Fortran 90 Compiler
mpixlf95	mpixlf95_r	bgxlf95	IBM BG Fortran 95 Compiler
mpixlf2003	mpixlf2003_r	bgxlf2003	IBM BG Fortran 2003 Compiler

■ MPI wrappers for GNU compilers:

Wrapper	Compiler	Description
mpicc	powerpc-bgp-linux-gcc	GNU BG C Compiler
mpicxx	powerpc-bgp-linux-g++	GNU BG C++ Compiler
mpif77	powerpc-bgp-linux-gfortran	GNU BG Fortran 77 Compiler
mpif90	powerpc-bgp-linux-gfortran	GNU BG Fortran 90 Compiler

■ “-show” option: shows complete command used to invoke compiler

ex: mpixlc -show sum.c

```
/opt/ibmcmp/vacpp/bg/9.0/bin/bgxlc sum.c -I/bgsys/drivers/V1R3M0_460_2008-081112P/ppc/comm/default/include -I/bgsys/drivers/V1R3M0_460_2008-081112P/ppc/comm/sys/include -L/bgsys/drivers/V1R3M0_460_2008-081112P/ppc/comm/default/lib -Wl,-rpath,/bgsys/drivers/V1R3M0_460_2008-081112P/ppc/comm/default/lib -lmpich.cnk -L/bgsys/drivers/V1R3M0_460_2008-081112P/ppc/comm/sys/lib -Wl,-rpath,/bgsys/drivers/V1R3M0_460_2008-081112P/ppc/comm/sys/lib -ldcmfcoll.cnk -ldcmf.cnk -lpthread -L/bgsys/drivers/V1R3M0_460_2008-081112P/ppc/runtime/SPI -Wl,-rpath,/bgsys/drivers/V1R3M0_460_2008-081112P/ppc/runtime/SPI -ISPI.cna -lrt
```



Compiling Without Wrappers

- Wrappers like mpixlc are scripts that invoke the underlying compiler with the needed libraries specified
- Compilers can be invoked directly without wrapper but libraries must be explicitly specified:

```
$DP= /bgsys/drivers/ppcfloor
```

```
bgxlc prog.c -I/$DP/comm/default/include -I/$DP/comm/sys/include -L/$DP/default/lib -lmpich.cnk  
-L/$DP/comm/sys/lib -ldcmfcoll.cnk -ldcmf.cnk -lpthread -L/$DP/runtime/SPI -ISPI.cna -lrt
```

- All system libraries are located in /bgsys/drivers/ppcfloor
- All application libraries are located in /soft/apps



IBM XL BG/P Default Compiler Options

■ Default options for Fortran, C, C++ Compilers:

- `-O0` Lowest optimization setting
- `-qstaticlink` Link statically
- `-qnoautoconfig` Tune for specified architecture (no auto tuning)
- `-qarch=450d` Target BG/P Instruction Set
- `-qtune=450` Tune for BG/P architecture
- `-qcache=level=1:type=i:size=32:line=32:assoc=64:cost=8` Tuning information for Level 1 Instruction Cache
- `-qcache=level=1:type=d:size=32:line=32:assoc=64:cost=8` Tuning information for Level 1 Data Cache
- `-qcache=level=2:type=c:size=4096:line=128:assoc=8:cost=40` Tuning information for Level 2 Cache

Example:

```
bgxlc test.c
```



```
bgxlc -O0 -qstaticlink -qnoautoconfig -qarch=450d -qtune=450 -qcache=... test.c
```



IBM XL Optimization Settings Options

Level	Implies	Description
-O0	-qstrict -qfloat=noftint:norsqrt:rngchk -qstrict_induction	Minimal optimization, preserves program semantics, best for debugging
-O2 (or -O)	-qstrict -qfloat=noftint:norsqrt:rngchk -qnostrict_induction -qmaxmem=8192	Preserves program semantics, eliminates redundant code, basic loop optimization
-O3	-qnostrict -qfloat=fltint:rsqrt:norngchk -qnostrict_induction -qmaxmem=-1 -qhot=level=0	High order loop analysis and transformations, better loop scheduling, inlining, in depth memory access analysis, <i>can alter program semantics</i>
-O4	<i>All -O3 options plus</i> -qhot=level=1 -qhot=vector -qipa=level=1	Additional loop analysis, basic interprocedural optimization, <i>can alter program semantics</i>
-O5	<i>All -O4 options plus</i> -qipa=level=2	Advanced interprocedural analysis, <i>can alter program semantics</i>



Additional IBM XL Optimization Controls

- qarch Specifies processor architecture (default=450d)
- qtune Specifies system architecture for optimization (default=450)
- qcache Describes the cache configuration (default= BG/P cache)
- q[no]strict_induction Turn on/off loop induction variable optimizations
- q[no]strict Turn on/off optimizations that can alter semantics
- qhot Range of options controlling higher order optimizations
- qipa Range of options controlling interprocedural optimizations
- qfloat Range of options controlling handling of floating points
- qmaxmem Specifies memory limit for compiler while optimizing
- qalias Assertions about aliasing in program, effects optimizations
- qunroll Enable additional search for loop unrolling
- q[no]unwind Specifies whether stack can be unwound
- qsmp=[noopt,omp,auto ..] Enables/disables thread level parallelization using OpenMP
or compiler automatic parallelization



Hierarchy of Optimization Levels

- A suggested set of optimization levels from least to most optimization:
 - -O0
 - -O2
 - -O2 -qmaxmem=-1 -qhot=level=0
 - -O3 -qstrict
 - -O3
 - -O3 -qhot=level=1
 - -O4
 - -O5
- Tips:
 - -O0 is best level for use with a debugger
 - -O2 is a good level for verifying correctness and establishing a performance baseline
 - Performance can decrease at higher levels of optimization, especially at -O4 or -O5
 - May specify different optimization levels for different routines/files



Optimization Steps w/o Code Changes

- Start from original MPI program, make it run
 - The least aggressive compiler options (-O2)
 - Default libraries
- Increase compiler optimization level
- Verify different running modes: smp vs. dual vs. vn
- Use highly optimized libraries (BLAS-LibGOTO, MASSV, ESSL)
- Optimize communication performance:
 - Fast MPI libraries: use alternate compiler wrappers in `/bgsys/drivers/ppcfloor/gnu-linux/bin`
 - MPI Environment Variable (ex: `DCMF_EAGER`)
- Optimize mapping (logical MPI-task to core) using `BG_MAPPING`



Other IBM XL Compiler Options

- *Input:*

- W<program>,<option>: pass options to a specified compiler program (ex: -WF,-D<macro>)

- *Language Control:*

- qlanglvl: sets language level and options for compilation

- qautodbl: convert single precision to double precision (Fortran only)

- *Portability:*

- qalign: specifies alignment of data objects

- *Error Checking & Debugging:*

- g: generates debugging information used by debugging tools

- qcheck: enables runtime checks of items like array bounds

- qfltrap: controls trapping of runtime floating point exception (default it no trapping)

- pg: produce profiling output for use with tools like gprof

- qoptdebug: produces file containing pseudo-code for use by debugger at high optimization levels



Other IBM XL Compiler Options (cont.)

- *Output:*

- o: specify output path and/or filename

- c: produce object code only and do not link

- S: produce assembly code

- *Information:*

- v or -V: provide detail information on each step of the compilation process

- qreport: produce a transformation report showing how the code was optimized

- qlist: produce a listing including an object listing

- qsource: produce a listing including source code

- qlistopt: produce a listing showing options in effect for compiler

- qipa=list: produce a listing containing interprocedural optimization information



Sample Blue Gene/P makefile

```
CC = mpixlc
```

```
CXX = mpixlcxx
```

```
FC = mpixlf90
```

```
OPTFLAGS = -O3
```

```
CFLAGS = $(OPTFLAGS) -qlist -qsource -qreport -g
```

```
FFLAGS = $(OPTFLAGS) -qlist -qsource -qreport -g
```

```
myprog: myprog.c
```

```
    $(CC) $(CFLAGS) -o myprog myprog.c
```



Common Compiling Problems:

- **A program does not compile or link; there are undefined symbols or definitions that seem to be from system libraries**
 - Make sure you are:
 - Cross-compiling with correct mpi-wrapper script or a bg-prefixed compiler
 - Not using /usr/include include files
 - Not using and /usr/lib library files
 - Remember:
 - » System libraries are located in /bgsys/drivers/ppcfloor.
 - » Application libraries and tools are located in /soft/apps
- **Undefined references at link time: undefined reference to ``_xlf_create_threadlocal'`**
 - Use a thread-safe version of a compiler, which has an `_r` suffix (ex: `mpixlf90_r` instead of `mpixlf90` in)



More Information on the XL Compilers

■ Man pages:

- `man xlf`
- `man xlc`
- `man xlc++`

■ Links to compiler documentation:

■ <https://wiki.alcf.anl.gov/index.php/References>

- Using the IBM XL Compiler for Blue Gene
- IBM XL C/C++ Advanced Edition for Linux, V9.0 Getting Started
- IBM XL C/C++ Advanced Edition for Linux, V9.0 Compiler Reference
- IBM XL C/C++ Advanced Edition for Linux, V9.0 Language Reference
- IBM XL C/C++ Advanced Edition for Linux, V9.0 Programming Guide
- IBM XL Fortran Advanced Edition for Linux, V11.1 Getting Started
- IBM XL Fortran Advanced Edition for Linux, V11.1 Compiler Reference
- IBM XL Fortran Advanced Edition for Linux, V11.1 Language Reference
- IBM XL Fortran Advanced Edition for Linux, V11.1 Optimization and Programming Guide

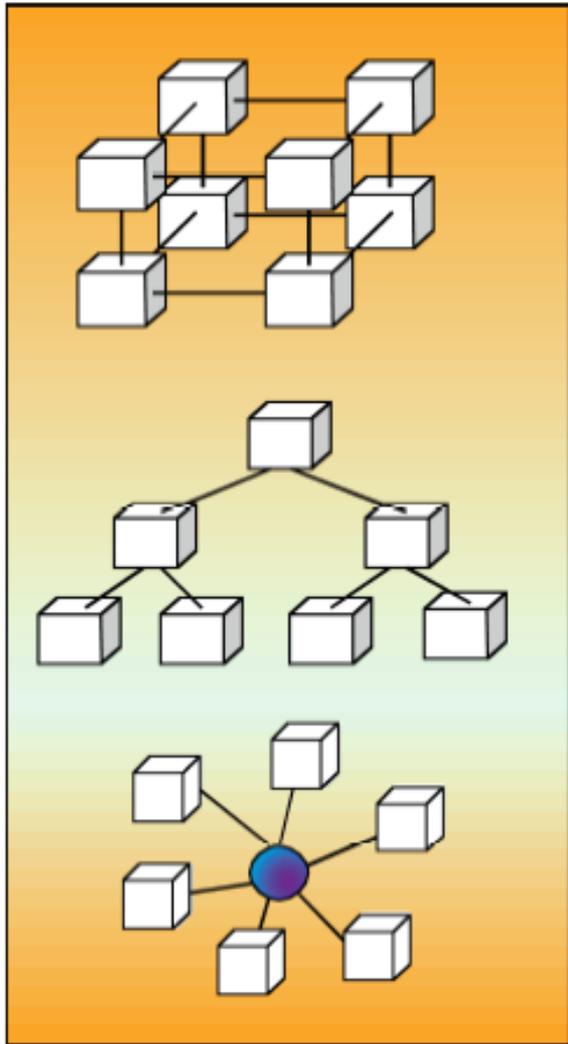


MPI on BG/P - MPI

- The primary function of BG/P: running your MPI-based applications
- BG/P MPI looks suspiciously like MPICH2 1.0.x
 - Almost MPI standard 2.0
 - No process management (MPI_Spawn(), MPI_Connect(), etc)
 - MPI-IO supported
 - One-sided communication supported
 - Based on MPICH2 1.0.7 base code
 - Attempting to to re-integrate all BGP changes into MPI main tree
- Utilizes the 3 different BG/P networks for different MPI functions



Blue Gene/P Interconnection Networks



3 Dimensional Torus

- Interconnects all compute nodes
- Virtual cut-through hardware routing
- 3.4 Gb/s on all 12 node links (5.1 GB/s per node)
- 0.5 μ s latency between nearest neighbors, 5 μ s to the farthest
- MPI: 3 μ s latency for one hop, 10 μ s to the farthest
- Communications backbone for point-to-point
- *Requires half-rack or larger partition*

Collective Network

- One-to-all broadcast functionality
- Reduction operations for integers and doubles
- 6.8 Gb/s of bandwidth per link per direction
- Latency of one way tree traversal 1.3 μ s, MPI 5 μ s
- Interconnects all compute nodes and I/O nodes

Low Latency Global Barrier and Interrupt

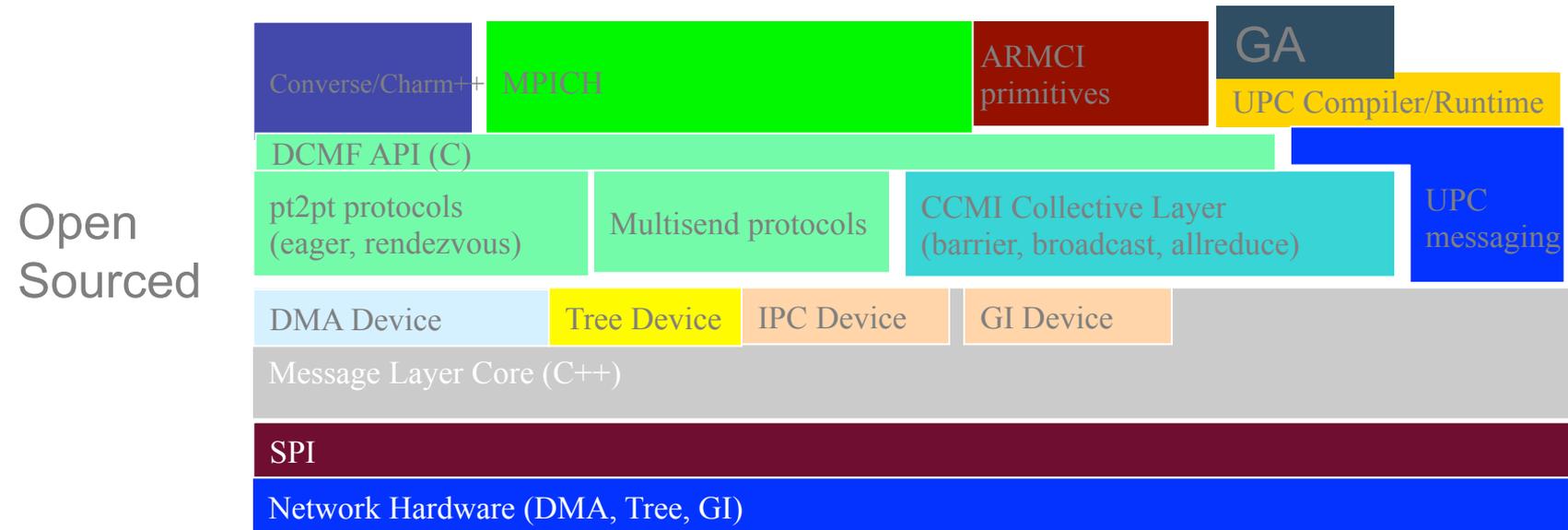
- Latency of one way to reach 72K nodes 0.65 μ s, MPI 1.6 μ s

Torus Network

- Cabling of the ANL BG/P
 - enables large partition configurations
 - puts some restrictions on small configurations
- Places restrictions on available partitions
 - For single rack (1024 node) job, torus HW in adjacent rack is put in “passthrough” mode, looping back without its nodes participating
 - Prevents a 1024 node job in that adjacent rack
 - 4-rack (4096 node) job, prevents an adjacent 4096 node job
- Job scheduler (Cobalt) will prevent running conflicting jobs



BG/P Messaging Stack



- Multiple programming paradigms supported
 - MPI and ARMCI, Charm++ and UPC (as research initiatives)
- SPI : Low level systems programming interface
- DCMF : Portable active-message API

MPI Torus Mapping

- Arrangement of MPI processes on the Torus network is controlled by a mapping
- Appropriate mappings can reduce the cost of point-to-point communication
- Mappings associate an MPI rank with a location in the Torus specified in (X,Y,Z,T) coordinates
- (XYZ) are torus coordinates, T is a CPU number
- Default mapping is TXYZ (what you get if you don't specify anything else)
 - Each node is filled with processes before moving to the next
- Various XYZT permutations are pre-defined (YZXT, TZYX, ... - T is always 1st or last)



MPI Torus Mapping

- Non-default mappings may be specified in two ways:
 - `qsub --env BG_MAPPING=TXYZ --mode vn ...`
 - This puts MPI task 0,1,2,3 to Node 0 CPU0, CPU1, CPU2, CPU3; MPI tasks 4,5,6, and 7 to Node2 CPU0,CPU1,CPU2,CPU3
 - `qsub --BG_MAPPING=<FileName> --mode smp ...`
 - use high-performance toolkits to determine communication pattern
 - optimize mapping by custom mapfile
 - mapfile: each line contains 4 coordinates to place the task, first line for task 0, second line for task 1...
 - avoid conflict in mapfiles (no verification)



Partition Dimensions

Nodes	X	Y	Z	Torus
64	4	4	4	No
128	4	4	8	No
256	8	4	8	No
512	8	8	8	Yes
1024	8	8	16	Yes
2048	8	8	32	Yes
4096	8	16	32	Yes
8192	8	32	32	Yes
16384	16	32	32	Yes
24576	24	32	32	Yes
32768	32	32	32	Yes
40960	40	32	32	Yes



MPI on BG/P - Communicator Creation

- API extensions to map nodes to specific Torus/pset configurations
 - MPIX_Cart_comm_create()
 - Returns an MPI communicator that is exactly the same as the underlying hardware - the X, Y & Z dimensions match those of the partition
 - The coordinates of a node in the communicator match its coordinates in the partition
 - Reduces need for complex node-mapping files
 - This is a collective operation and must be run on all nodes
 - Check the return code when using this function! Look for MPI_SUCCESS
 - MPIX_Pset_same_comm_create()
 - Returns a communicator where all nodes belong to the same pset
 - MPIX_Pset_diff_comm_create()
 - Returns a communicator where all nodes have the same pset rank
- MPI_Cart_create()
 - MPI_Cart_create() with reorder true attempts to give communicators that mirror hardware
 - DCMF_TOPOLOGY=0 disables any attempt to give good communicators from MPI_Cart_create() call



MPI on BG/P - Collectives

- Currently optimized collectives:
 - Broadcast (COMM_WORLD, rectangle, arbitrary)
 - (All)reduce (COMM_WORLD, rectangle, arbitrary)
 - Alltoall(v|w) (all comms, single threaded only)
 - Barrier (COMM_WORLD, arbitrary)
 - Allgather(v) (uses (async)bcast, reduce, or alltoall)
 - Gather (uses reduce)
 - Reduce_scatter (uses reduce, then scatterv)
 - Scatter (uses bcast)
 - Scatterv (uses bcast or alltoallv)



MPI Environment Variables

- Behavior of the MPI implementation can be altered through environment variables:

DCMF_EAGER Sets cutoff for switch to rendezvous protocol

DCMF_TOPOLOGY Enable/Disable optimized topology for cart_create()

DCMF_BCAST Controls protocol used for broadcast

DCMF_COLLECTIVES Enable/Disable optimized collectives

Plus several dozen more ...

- See Redbook “*IBM System Blue Gene Solution: Blue Gene/P Application Development*” Appendix D for complete list



Threading Support

- OpenMP is supported
- pthreads is supported
 - NPTL pthreads implementation in glibc requires no modifications
- Compiler auto thread parallelization is available
 - use `-qsmp=auto`
 - not always effective
- Compute Node Kernel supports
 - execution of one quad-threaded process
(each of the CPUs is assigned to each of maximum 4 threads)
 - execution of two two-threaded processes
 - execution of four single-threaded processes
 - proper mode should be specified for `qsub`



OpenMP Implementation

- Shared-memory parallelism is supported on single node
- Interoperability with MPI as
 - MPI at outer level, across compute nodes
 - OpenMP at inner level, within a compute node
- Thread-safe compiler version should be used (mpixlc_r etc.)
 - with any threaded/OMP/SMP applications
- OpenMP 2.5 standard directives are supported:
 - parallel, for, parallel for, sections, parallel sections, critical, single
 - #pragma omp <rest of pragma> for C/C++
 - !\$OMP <rest of directive> for Fortran
- Compiler functions
 - omp_get_num_procs, omp_get_num_threads
 - omp_get_thread_num, omp_set_num_threads
- Number of OpenMP threads
 - set using environment variable OMP_NUM_THREADS
 - must be exported to the compute nodes using qsub –env flag



Software Libraries

- ALCF Supports two sets of libraries:
 - IBM system and provided libraries: [/bgsys/drivers/ppcfloor](#)
 - glibc
 - mpi
 - DCMF (Deep Computing Messaging Framework)
 - SPI (System Programming Interface)
 - UPC (Universal Performance Counters)
 - BG/P Personality
 - Site supported libraries and programs: [/soft/apps](#)
 - PETSc
 - FFTW
 - HDF5
- https://wiki.alcf.anl.gov/index.php/Applications_and_Libraries



Driver Versions

- The driver is the system software stack and includes:
 - Kernel
 - Compilers
 - Libraries and headers
- Driver directory:
/bgsys/drivers/ppcfloor (symlink to current driver directory)
- Recently upgraded to version V1R4M2
 - Should recompile code built with previous driver
- Each driver version may have many “efixes”
 - Need to relink to take advantage of



Supported Libraries and Programs

Library	Location	Description
BLAS, LAPACK	/soft/apps/blas-lapack-lib	Basic vector linear algebra subroutines.
ESSL	/soft/apps/ESSL-4.{3,4}	Mathematical subroutines designed to improve the performance of engineering and scientific applications on BlueGene
LIBGOTO	/soft/apps/LIBGOTO	Very efficient BLAS-1.2.3 implementation for BlueGene from Kazushige
SCALAPACK	/soft/apps/SCALAPACK	High-performance linear algebra routines for distributed-memory message-passing MIMD computers and networks of workstations.
PETSc	/soft/apps/petsc	A suite of data structures and routines that provide the building blocks for the implementation of large-scale application codes on serial and parallel computers.
fftw-2.1.5, fftw-3.1.2	/soft/apps/fftw-{2.1.5,3.1.2}	A library for computing the discrete many-dimensional Fourier transform
p3dffft	/soft/apps/p3dffft-2.2*	Highly scalable parallel 3D Fast Fourier Transforms library.
hypre-2.0.0	part of PETSc install	A library for solving large, sparse linear systems of equations on massively parallel computers.
MUMPs-4.7.3	part of PETSc install	A multifrontal massively parallel sparse direct solver.
spooles-2.2	part of PETSc install	A library for solving sparse real and complex linear systems of equations.



Supported Libraries and Programs

Program	Location	Description
TotalView	/soft/apps/totalview-8.5.0-0	Multithreaded, multiprocess source code debugger for high performance computing.
Coreprocessor	/soft/apps/coreprocessor.pl	A tool to debug and provide postmortem analysis of dead applications.
TAU-2.19	/soft/apps/tau/tau_latest	A portable profiling and tracing toolkit for performance analysis of parallel programs written in Fortran, C++, and C
HPCT	/soft/apps/hpct_bgp	MPI profiling and tracing library, which collects profiling and tracing data for MPI programs.

Program	Location	Description
armci	/bgsys/drivers/ppcfloor/comm	The Aggregate Remote Memory Copy (ARMCI) library
HDF5	/soft/apps/hdf5-1.6.6	The Hierarchical Data Format (HDF) is a model for managing and storing data.
NetCDF	/soft/apps/netcdf-3.6.2	A set of software libraries and machine-independent data formats that supports the creation, access, and sharing of array-oriented scientific data.
Parallel NetCDF	/soft/apps/parallel-netcdf-1.0.2	A library providing high-performance I/O while still maintaining file-format compatibility with Unidata's NetCDF.
mercurial-0.9.5	/soft/apps/mercurial-0.9.5	A distributed version-control system
Scons	/soft/apps/scons-0.97	A cross-platform substitute for the classic Make utility
tcl-8.4.14	/soft/apps/tcl-8.4.14	A dynamic programming language,

Optimized libraries

- Math libraries:

- BG-optimized BLAS Level 1,2,3 from Kazushige Goto, U. of Texas
- IBM ESSL library: BLAS1, 2, 3 in /soft/apps/ESSL
- Generic versions of BLAS/LAPACK/FFTW

- Performance comparisons for matrix multiply:

-O0	102.82 s	20.88 MFlop/s	
-O2	70.86 s	30.31 MFlop/s	
-O3	3.471 s	618.52 MFlop/s	
-O4	7.921 s	271.10 MFlop/s	
-O5	7.919 s	271.12 MFlop/s	
ESSL	0.836 s	2569.6 MFlop/s	(75.76 % of peak)
GOTO	0.828 s	2593.0 MFlop/s	(76.26 % of peak)



Personality API

- The Personality API provides a way of finding BG/P hardware and environment information for a process
- Example:

```
#include <common/bgp_personality.h>
#include <common/bgp_personality_inlines.h>

_BGP_Personality_t p;

Kernel_GetPersonality( &p, sizeof(p) );

p.DDR_Config.DDRSizeMB;          /* memory size */
p.Kernel_Config.ProcessConfig;  /* running mode */
p.Network_Config.Xnodes;        /* torus dimensions */
p.Network_Config.Ynodes;
p.Network_Config.Znodes;
```



Cobalt*: An ANL's Scheduler for HPC

- Research in nature - investigating advanced systems management for complex cutting-edge architectures
- Open source and uses open source components enabling rapid experimentation and exploration advanced features
- Fits to both computational needs and computer science research (most resource managers are not system software research environments)

* <http://www-unix.mcs.anl.gov/cobalt/index.xml>



Cobalt commands

■ Job management commands

qsub: submit a job

qstat: query a job status

qdel: delete a job

qalter: alter batched job parameters

qmove: move job to different queue

qhold: place queued (non-running) job on hold

qrls: release hold on job

cqwait: wait for job to finish

showres: show current and future reservations

qavail: show available backfill partitions



qsub options

- Syntax:

```
qsub [-d] [-v] -A <project name> -q <queue> --cwd <working directory>  
--env envvar1=value1:envvar2=value2 --kernel <kernel profile>  
-K <kernel options> -O <outputprefix> -t time <in minutes>  
-e <error file path> -o <output file path> -i <input file path>  
-n <number of nodes> -h --proccount <processor count>  
--mode <mode> -M <email> --dependencies <jobid1>:<jobid2> <command> <args>
```

- Standard options:

-A project	project to charge
-q queue	queue
-t <time_in_minutes>	required runtime
-n <number_of_nodes>	number of nodes
--proccount <number_of_cores>	number of CPUs
--mode <smp dual vn>	running mode
--env VAR1=1:VAR2=1	environment variables
<command> <args>	command with arguments



More qsub options

- Other options:

- O <outputprefix> name of output files (<outputprefix>.error ...)
- h submit in held state
- M <email_address> send mail at start and end of job
- dependencies <job1>:<job2> hold jobs until listed jobs are complete
- o <output_file_path> send executables standard out to specified file
- e <error_file_path> send executables standard error to specified file
- i <input_file_path> connect executables standard input to specified file



Simple qsub example

```
qsub -A <project> -q prod -t 30 -n 512 --proccount 2048  
--mode vn <executable> <executable-args>
```

-A <project> : if only on 1 project defaults to that

-q <queue> : usually “prod” or “prod-devel”

-t <time> : total job time in minutes, ***includes boot time***

-n <nodes> : number of nodes requested

--proccount <np> : number of MPI tasks to start

--mode <smp/dual/vn> : runtime mode



Job boot times

- Each time a job is submitted using a standard qsub command all of the nodes in a partition are rebooted
- Boot times depend on the size of the partition

Nodes in Partiion	Boot time (seconds)
512	80
1024	86
2048	105
4096	166
8192	256
16384	351
24576	532
32768	712



Runtime Mode

■ SMP mode

- `qsub --mode smp`
- Up to 1 MPI task on each node, 2 GB RAM/task (shared by 4 cores)

■ Dual mode

- `qsub --mode dual`
- Up to 2 MPI tasks on each node, 1GB RAM/task (shared by 2 cores)

■ Virtual Node mode

- `qsub --mode vn`
- Up to 4 MPI tasks on each node, 512 MB RAM/task (1 core each)



qsub: More Examples

- Despite being redundant, we recommend to always specify the number of nodes, the number of CPUs, and the mode of your run
- `qsub -q prod-devel -t 30 -n 32 --proccount 32 --mode smp myprog`
 - will use smp-mode with 32 MPI tasks on 64 nodes (smallest # available)
- `qsub -q prod -t 30 -n 256 --proccount 256 --mode smp myprog`
 - will never run (wants to run on 256 node partition, but none in prod)
 - qsub will actually reject this (and any number of nodes < 512)
 - can do: `qsub -q prod -t 30 -n 512 --proccount 256 --mode smp myprog`
- `qsub -q prod -t 60 -n 1024 --proccount 4000 --mode vn -O myprefix`
`--env BG_MAPPING=XYZ myprog myargs`
 - will use 4000 out of 4096 cores available
- `qsub -q prod -t 60 -n 512 --proccount 512 --mode smp`
`--dependencies 145:146:147 myprog myargs`
 - job won't start until jobs 145, 146 and 147 complete (without errors)



Queues

- “prod-devel” queue for testing and debugging
 - Partition sizes: 64 – 512 nodes (in powers of 2)
 - Time limit: 1 hour
 - Max of 20 submitted jobs and 5 running jobs
 - Priority is given to small, short jobs
- “prod” queue for production compute jobs
 - Partition sizes: 512 – 32768 nodes (in powers of 2)
 - Time limit: 12 hours
 - Max of 20 submitted jobs and 5 running jobs
 - Priority is given to large jobs
- Other special queues exist for mostly administrative purposes and are not generally available for running jobs (see all with `qstat -Q`)
- All of the machine available to capability jobs (8+ racks). Smaller availability for small (<8 rack) jobs that run 6+ hours



Cobalt files for a job

- Cobalt will create 3 files per job, the basename “<prefix>” defaults to the jobid, but can be set with the “qsub -O myprefix” option
- cobaltlog file: <prefix>.cobaltlog
 - first file created by Cobalt after a job is submitted
 - contains submission information from qsub command, mpirun, and environment variables
- error file: <prefix>.error
 - created at the start of a job
 - contains job startup information and any content sent to standard error while the user program is running
- output file: <prefix>.output
 - contains any content sent to standard output by user program



Methods of submitting a job

- Run *qsub* from the command line:

```
qsub -q prod-t 10 -n 64 --proccount 64 --mode smp Hello
```

- Runs only the compiled executable file, no pre-processing or post-processing

- Place the *qsub* command within a shell script:

```
#!/bin/bash
```

```
RUN=<program_executable>
```

```
NODES=64
```

```
...
```

```
qsub -q short -t 0:10:00 -n $NODES --proccount $CORES --mode $MODE -O $TASK --env BG_MAPPING=$MAPPING $RUN
```

- Script runs outside the scheduler on the login node as soon as it is invoked
- Allows pre-processing but ‘*qsub*’ is non-blocking so post-processing is harder

- Use Cobalt in script mode

```
qsub --mode script ... job.sh
```

- Job script run by the scheduler only after the job starts
- Job script runs on a dedicated node different from the login node
- Allows for pre-processing and post-processing at the end of the job
- Environment not the same as your current environment – `dumpenv.sh > my_environment.sh` first



qsub: A Script to Submit a Typical Job

```
#!/bin/bash
RUN=<program_executable>
NODES=64
CORES=256
MODE=vn
MAPPING=XYZT
TASK=$RUN-$NODES-$CORES-$MODE

rm -rf $TASK.error $TASK.output
echo Processors: nodes $NODES, cores $CORES, mode $MODE
qsub -q prod-devel -t 30 -n $NODES --proccount $CORES --mode $MODE -O $TASK \
    --env BG_MAPPING=$MAPPING $RUN
qstat -f
touch $TASK.error
tail -f $TASK.error
```



Cobalt Script Mode

- Sample Script:

```
#!/bin/sh

echo "Starting Cobalt job script"

# Do pre-processing work here

# set up environment - See the section on Script Environment
. my_environment.sh

# Run executable (important- do not use plain 'mpirun')
cobalt-mpirun -mode vn -np $NODES -cwd `pwd` -env "FOO=1 BAR=2" myprog1.exe args

# Do post-processing work here

# Run another executable on the same partition
cobalt-mpirun -mode vn -np $NODES -cwd `pwd` -env "FOO=1 BAR=2" myprog2.exe args

...
```

- Submit using script mode “qsub --mode script -q prod -t 10 ... job.sh”
- Use ‘cobalt-mpirun’ inside script, not ‘mpirun’ or ‘qsub’
- ‘cobalt-mpirun’ blocks until job is complete whereas ‘qsub’ does not



Waiting for a Job to Complete

- The ‘qsub’ command is non-blocking, returns immediately after job is submitted
- To block until job is started or completed use the ‘cqwait’ command
`cqwait [--start] <jobid>`
- Default behavior is to block until the specified job completes
- With option ‘--start’ blocks until job starts
- Useful when running qsub in a shell script



qstat: Show Status of a Batch Job(s)

■ qstat -f

- a full display is produced

```
JobID      JobName      User      WallTime  QueuedTime  RunTime  Nodes  State  Location      Mode  Procs  Queue  StartTime
=====
11543 fl-64-64-smp morozov 00:30:00 00:00:06 00:13:41 64  running ANL-R00-M1-N02-64 SMP 64 short 02/27/08
```

- JobID can be used to kill the job or alter the job parameters
- valid status: queued, running, user_hold, maxrun_hold
- check the mode of your job

■ qstat -lf <jobid>

- shows many details of a job

■ qstat -Q

- will show all available queues and their limits
- special queues, which we use to handle reservations



Altering Submitted Job Parameters

- qalter

- Allows alteration of the parameters of a queued (but not running) job

```
qalter -t <time> -A <project> -n <nodes> --mode <mode>  
--proccount <processes> -M <email> -e <stderr> -o <stdout> <jobid>
```

- qmove

- Move a job to a different queue

```
qmove <new_queue> <jobid1> <jobid2>
```

- qhold

- Hold a submitted job

```
qhold <jobid1> <jobid2>
```

- qrls

- Release a held job

```
qrls <jobid1> <jobid2>
```



Reservations

- Reservations allow exclusive use of a partition for a specified group of users for a specific period of time
- <https://wiki.alcf.anl.gov/index.php/Queuing#Reservations>
- A reservation blocks other users jobs from running on that partition
- Often used for system maintenance or debugging
- Reservations are sometimes idle, but still block other users jobs from running on a partition
- The ‘showres’ command can be used to determine if any reservations are in place

showres [-l]

Reservation Queue User

Start

Duration

Partitions

```
=====
```

pm	R.pm	acherry:wscullin:janderso:lueningh:loren:davidr:harms:morozov:buettner	Mon Feb 16 08:00:00 2009	12:00	ANL-R00-1024
----	------	--	--------------------------	-------	--------------



Partition related commands

- **partlist**

- shows status of partitions

Name	Queue	State
ANL-R00-1024	default:spruce	blocked (ANL-R00-M0-N00-256)
ANL-R00-M0-512	default:spruce	blocked (ANL-R00-M0-N00-256)
ANL-R00-M1-512	default:spruce	idle
ANL-R00-M0-N00-256	default:spruce	busy

- **bg-listblocks**

- shows allocated partitions
- useful if cobalt is restarted and “forgets” about jobs
- “bg-listblocks -all” shows all defined partitions

- **bg-listjobs**

- shows running jobs

- **qavail**

- shows partitions available for backfill



Why a job is not running in a queue

- there is a reservation, which interferes with your job
 - `showres` shows all reservations currently in place
- there is no available partitions
 - `partlist` shows all partitions marked as functional
 - `partlist` shows the assignment of each partition to a queue
- wrong queue
 - the job submitted to a queue, which is restricted to run at this time
- partitions are draining for a large job
 - to make space for a large job, the subpartitions of the large partition must not allow new jobs on that will last longer than the remaining drain time
 - the state is identified by a combination of `bg-listblocks -all, qavail`
- Your job cannot run next to an existing job because of wiring dependencies



Core Files

- Jobs experiencing fatal errors will general produce a core file for each process
- Examining core files:
 - Core files are in text format, readable with the “more” command
 - `bgp-stack` command provides call stack trace from a core file:
 - Ex: `bgp-stack <executable> <corefile>`
 - Command line interface
 - Can only examine one core file at a time
 - `coreprocessor` command provides call stack trace from multiple cores
 - GUI interface requires X11 forwarding (`ssh -X intrepid.alcf.anl.gov`)
 - Provides information from multiple core files
- Environment variables control core dump behavior:
 - `BG_COREDUMPONEXIT`: core dump when application exits
 - `BG_COREDUMPDISABLED`: disable core dumps
 - https://wiki.alcf.anl.gov/index.php/Debugging#Core_file_settings



Filesystems on Blue Gene/P

■ Home filesystem:

- GPFS
- /home/<username> -> /gpfs/home/<username>
- moderate performance, moderate capacity file system. Not for jobs.
- visible from login, compute, I/O, and service nodes
- limited in space, soft quota of ~200 GB
- daily snapshots in ~/.snapshots

■ Data filesystem:

- Directories:
 - /intrepid-fs0/users/<username> [GPFS]
 - /intrepid-fs1/users/<username> [PVFS]
- high performance, high capacity file system
- visible from login, I/O, and compute nodes
- scratch data space, no backups
- soft quota of 1TB



File systems on Blue Gene/P (cont.)

- Scratch directories:
 - No local scratch directory is available on the compute nodes
 - /scratch or /tmp on the login nodes cannot be seen by the compute nodes
 - fast place to perform compiles
- Tape archive
 - ftp-like interface ‘get’ and ‘put’ operations
 - Use “hsi” command
 - For help use “hsi help”



File transfers

- Two methods supported:
 - scp:
 - should work anywhere
 - can be slow
 - GridFTP:
 - supported authentication:
 - ssh
 - grid credentials
 - globus-url-copy command line client
 - May also try the new Globus Online service

Softenv

- A tool for managing a user's environment
- Settings:
 - Maintained in the file `~/.softenvrc`
 - Add/remove keywords from `~/.softenvrc` to change environment
- Commands:
 - `softenv`
 - a list of all keywords defined on the systems
 - `resoft`
 - reloads initial environment from `~/.softenvrc` file
 - `soft add|remove keyword`
 - Temporarily modify environment by adding/removing keywords



Clusterbank: Job Accounting

- cbank [projects]

Name	Jobs	Charged	Available
-----	-----	-----	-----
Catalyst	7814	929778.0	-869307.6
	-----	-----	-----
	7814	929778.0	-869307.6

- Other reports: allocations, users, charges

- Options:

- p <project> : show info for specific project(s)
- u <user> : show info for specific user(s)
- a <YYYY-MM-DD> : show info after date (inclusive)
- b <YYYY-MM-DD> : show info before date (exclusive)



Resources

■ Main support page

<http://www.alcf.anl.gov>

- Links on side panel:
 - Getting Started
 - Using ALCF Resources

■ ALCF Wiki

<http://wiki.alcf.anl.gov>

- Links to:
 - User Documentation
 - Applications & Performance

■ IBM RedBooks:

Compiler User Guides, Application Development Manuals

<http://www.redbooks.ibm.com/redbooks.nsf/redbooks/>



Help!

- Running Your Application Problems

- Porting, Compiling, and Running your jobs
- Performance issues, Tuning, Scaling, Debugging, Profiling
- Unexpected results, core dumps, apparently wrong result

Consult with your **Catalyst** or E-mail to support@alcf.anl.gov

- Login problems, access problems, environment problems, Job scheduler problems, system not responding

E-mail to support@alcf.anl.gov

