

Lawrence Livermore National Laboratory

The Dawn/Dawndev Build Environment

February 18, 2010



John Gyllenhaal

Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94551
This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344

LLNL-PRES-424544

Cross-compilation required on BG/P

- Use /usr/local/bin/mpi* compilers (.e.g., mpixlc, mpicc)
 - Serial bg* and powerpc-bgp* compilers also provided
 - OK to use mpi compilers for serial codes (unlike AIX)
- 32-bit static link default (limited shared library support)
- Few third-party libraries (no X11)
 - All LC provided backend libraries are in /usr/local/tools
- Function shipping: I/O, gethostname() returns I/O node name
- No fork(), system(), and usleep() call support (run-time error)
- Don't use sysconf(), use kernelGetPersonality()
- Good porting resources in /usr/local/doc
 - dawn.basics
 - BGP_Application_Development.pdf (use /opt/kde3/bin/kpdf)
 - All the secret sauce for shared libraries, compiler options, etc.
 - Bgp.pdf for power users (kernelGetPersonality() page 8)



Don't use frontend includes and libraries

- Backend headers tweaked for stricter alignment
 - Allows use of double hummer instructions
 - `#include <setjmp.h>` uses this optimization
- Frontend libraries contain unimplemented instructions
 - Often rely on unimplemented system calls
- Compiler wrappers in `/usr/local/bin` warn about
 - `-I/usr/include*`, `-L/usr/lib*`, `-L/lib*`, `-I/usr/local/include*`, `-L/usr/local/lib*`, `-I/usr/X11*`, `-L/usr/X11*`
 - `LLNL_CHECK_COMPILE_LINE` controls response
 - Values: Error, Warning, None



Autosubmit to HTC enables configuration scripts

- Special HTC Partition allows rapid execution of serial jobs
 - ‘submit a.out’ will run serial job on ‘HTC compute nodes’
 - 16 simultaneous 1 node (up to 4 thread) jobs allowed
 - No environment variables passed by default (use `–env` or `–env_all`)
 - ‘submit `–help`’ lists many useful options (e.g. `–raise`)
 - No MPI and, with current defaults, limited error messages
- All backend executables autosubmitted when run on frontend
 - `./a.out` translates to ‘submit `./a.out`’ for you
 - Actually ‘`${BG_PGM_LAUNCHER} ./a.out`’
 - Prepend ‘`noautosubmit`’ to unset `BG_PGM_LAUNCHER`
 - Env var ‘`HTC_SUBMIT_OPTS`’ treated as submit arguments
 - Would making default `–env_all` and `–raise` be useful?
 - Only ‘.’ searched when running (use `~/bin/a.out` if not in ‘.’)



Shared library constraints

- BG/P has the minimum shared library support required to get dlopen() and python working
 - Shared libraries are not shared among processes
 - Entire shared library loaded into memory
 - No demand paging in of shared library
 - Cannot unload shared library to free memory
 - Used by a few apps but not a common path (avoid if can)
- BG/P optimized for static executables
 - Executable shared among MPI tasks
 - Efficient loading/memory use



Threaded codes need mpixl*_r or gnu compilers

- The Gnu compilers are always thread-safe
 - No OpenMP support in default 4.1.2 version
 - Beta gcc 4.3.2 version supports OpenMP (-openmp)
 - /bgsys/toolchains/gomp/gnu-linux/bin/
- The IBM compilers need _r for thread-safety (adds -lpthread)
 - Asking for subtle problems if just link with -lpthread
 - Adds -qthreaded (thread-safe optimizations)
 - Adds -D_REENTRANT -D__VACPP_MULTI__
 - Very small performance impact with _r (default on AIX)
 - -qsmp=omp for OpenMP
- Remember to use 'volatile' for polled shared variables



Signal 7 and bus error reporting controls

- BG/P misaligned data accesses trap to fixup code
 - By default, 1000 free fixups (each taking ~1000 cycles)
 - Then, killed with signal 7 (bus error)
 - Often confusing since worked first 1000 times
 - Other LC platforms will fixup accesses forever by default
 - Found custom malloc that always(!) misaligns data
- Recommend ‘no free fixups’ during testing
 - `mpirun -env BG_MAXALIGNEXP=1` (same as 0 setting)
- Recommend ‘infinite fixups’ during production runs
 - `mpirun -env BG_MAXALIGNEXP=-1`
- `xc varargs` misalignment fixed on Dawndev , Dawn fix 2/25



Minimal MPI buffering can expose deadlocks

- Unexpected MPI messages may block on BG/P
 - Our other platforms will buffer message and continue
 - Has exposed “old” MPI coding bugs in our apps
 - Apps hung only on BG/P until MPI calls made legal
- LC’s Opteron clusters can configured to not buffer messages
 - Set these two environment variables before run
 - export VIADEV_RENDEZVOUS_THRESHOLD=0
 - export VIADEV_SMP_EAGERSIZE=0
 - Recommend adding to Opteron regression tests
- Invalid non-global use of MPI_comm_create has also caused hangs (on BG/L and BG/P only)



Static executables with >67MB of code

- 32bit PowerPC uses 24 bit jump/call instructions
 - This gives ~67MB range for calls and jumps
 - The startup code (crti.o) calls cleanup code (crtn.o)
 - Error: relocation truncated to fit: R_PPC_REL24 against ``.text'`
 - Limits executable to 67MB except gcc > 67MB!
 - Query with `'size a.out'` (`'text'` is limited to 67,108,864)

text	data	bss	dec	hex	filename
2615864	96940	669228	3382032	339b10	a.out
 - Use hidden gcc option meant solely for building gcc
 - `-Wl,--relax`
 - The `-Wl,--relax` option punts on multiple defs
 - May also need `-Wl,--allow-multiple-definition`

How to grab nodes for debugging

- Dawndev (all nodes in pdebug, 16 node granularity)
 - Use 'sinfo' to list available resources
 - nodes in 'Err' or 'drain' state is the 'HTC' partition
 - `salloc -N 16 -t 60 -p pdebug` (exit to release nodes)
 - 30 minute default, 60 minute max (via `-t 60`)
 - `mxterm 16 0 60 -q pdebug` (Never starts if time > 60)
- Dawn (several pools, 128 node granularity)
 - 1 day limit on pdebug, otherwise same commands
 - Jobs with mismatched granularity will never start
 - Special Dawn12 has separate 16 node pdebug
 - 15 minute default, 30 minute limit (via `-t 30`)
 - No moab on dawn12, so no `mxterm/msub` allowed on dawn12



Building serial executables that work on both compute and frontend nodes

- BG/P compute nodes (450d) different than Power5/6
 - Special double hummer instructions (not on power5/6)
 - Limited instruction set (power5/6 instructions will SIGILL)
- By design, standard unmodified libc and libpthread used
 - Fork() works on frontend, returns error on backend
 - C++, fortran, and math libraries use double hummer!
- Gnu C (mpicc) compiled backend code works on frontend
- IBM C (mpixlc) backend code compiled with `-qarch=450` works
- Currently need 'noautosubmit' to run on frontend
- Useful testcase: `/usr/local/tools/autosubmit/nodeinfo.c`



Another reason to avoid static C++ variables

- The order of initialization of static C++ variables is undefined
 - BG/P's order appears different than other platforms
 - Gnu's and IBM's order appear different on BG/P
 - Two large C++ apps segfault before main
 - One works with xlc++ only and one with g++ only
 - May be other issues at work, we are still debugging
- As mentioned last user meeting, avoid static vars if possible
 - Meyers' More Effective C++ Item 26 describes method
 - Clever use of “static in a function” to resolve many issues

