Coral Early Access (EA) Systems Compilers and MPI Overview

B453 R1001

Presented by John Gyllenhaal

April 19, 2017
Compiler goals for CORAL EA systems

- Enable testing and using OpenMP 4.5 with GPU offloading
  - IBM’s XL C/C++/Fortran and CORAL Clang C/C++
    - Using g++ 4.9.3 headers for reasonable C++11 support
  - Hardening of IBM’s new xlflang FORTRAN frontend for CORAL clang

- Testing and use of CUDA and IBM’s CUDA FORTRAN
  - nvcc with xlc++ and clang++ (really 4.0+ but nvcc requires 3.8.0 so it lies)
    - /usr/local/cuda-8.0/bin/nvcc right now (/usr/tce links in future)
  - IBM’s xlcuf provides CUDA FORTRAN support

- Mixing OpenMP and CUDA in bigger applications
  - Unified virtual memory popular but portability with OpenMP unclear

- Comparisons to PGI’s OpenACC and CUDA FORTRAN
  - Useful for comparing IBM products to but NOT for production use
  - OpenMP and OpenACC unlikely to be able to mix in same application
Recommend staying on most worn path

- **Recommend use XL FORTRAN for all FORTRAN**
  - Much higher performance than gfortran and supports GPU
  - Xlflang testing welcome to discover bugs but relatively new

- **Recommend use XL or Clang for C/C++**
  - XL on Power8LE now uses clang parser for better C++11 support
  - XL can generate debug info in GPU code, but cannot inline GPU runtime yet
    - This impacts GPU performance in some cases but GPU debuggers work
  - Clang can inline GPU runtime but cannot generate debug info for GPU yet
    - Often faster GPU code but debuggers cannot look at GPU code
  - Both are complex issues that may not be resolved until end of 2017

- **Don’t use –O4, -O5, IPA, correctness checkers, or rare options**
  - Some XL and clang options will likely result in Internal Compiler errors
    - Clang’s -fsanitize=address causes problems recently with GPU options enabled

- **Use IBM’s Spectrum-MPI**
  - Pure OpenMPI issues have been reported and are being investigated
Known gotchas building applications

- See link for more details:
  - [https://lc.llnl.gov/confluence/display/CORALEA/CORAL+EA+Systems](https://lc.llnl.gov/confluence/display/CORALEA/CORAL+EA+Systems)

- Clang doesn’t support GPU code in static libraries yet
  - Only ONE shared library can contain GPU kernels
  - XL compilers support GPU code static libraries, same shared limitation

- GPU code in multiple shared objects problematic for everyone
  - No dynamic linker for GPU code yet (on any platform, including with straight CUDA)
  - Don’t put GPU code in “main” if GPU code is in shared libraries
  - Still having problems in complex apps that we are trying to understand

- Compiler more fragile and much slower with GPU offload support
  - If run into problems, report problems and then only use GPU options where needed
  - Problems often in third party libraries where odd options used

- Complex build systems can get different compilers by mistake
  - Dangers of module system getting reset in sub-make command
  - Solution is to hard code full paths to compilers
Three methods to access compilers and MPI

- `/usr/tcetmp/bin` – Primary method until update week of 4/10/2017
  - Unqualified compiler names are ~latest versions of MPI, XL, clang, gcc, PGI
    - `xlC`, `mpixlC`, `mpixlC-gpu`, `clang++`, `mpiclang++`, `mpiclang++-gpu`, `g++`, `mpig++`, etc.
  - Default compiler versions expected to change monthly for XL and Clang
  - `gcc 4.9.3` headers and libraries used by default (LLNL-specific change)

- Modules – TOSS3-like and good for seeing options and full paths
  - Defaults to ‘xl’ and ‘spectrum-mpi’ modules as specified in ‘StdEnv’ module
  - ‘ml clang’ or ‘module load clang’ will switch to latest clang compiler
  - Portable MPI wrappers intentionally now only defined in modules (as of 4/10)
    - `mpicc` `mpiCC` `mpic++` `mpicxx` `mpif77` `mpif90` `mpifort`
    - Helps detect makefile issues with modules (e.g., ‘mpicc not found’ flags makefile issue)
  - Modules define compiler-specific wrappers as appropriate: `mpixlC`, `mpixlf90`
  - Allows mixing `mpiclang++` and `mpixlf90` (from `/usr/tcetmp/bin`) if needed

- Full path – Ignores modules loaded for consistent builds
  - After loading modules, run ‘which mpixlC’ to find path
    - `/usr/tcetmp/packages/spectrum-mpi/spectrum-mpi-2017.04.03-xl-beta-2017.03.28/bin/mpixlC`
    - Will not see our compiler/MPI version updates until you change path
Brief Lmod ‘ml’ command overview

- Taken from this TOSS3 presentation which has longer module overview: https://lc.llnl.gov/computing/meetings/2016Sept_Gyllenhaal.pdf

- ml: A convenient tool (included with lmod)
  - ml means module list (what’s loaded)
  - ml avail means module avail (what can be loaded)
  - ml foo means module load foo
  - ml -bar means module unload bar
  - ml foo -bar means module unload bar; module load foo
  - ml can be used everywhere module can be
    - Considered a bug if you ever have to type ‘module’ again
    - ml show foo (show what module foo will do)
    - ml whatis foo (show description of module foo)
    - ml keyword xl (show modules with xl in the description text)

- See http://lmod.readthedocs.io/en/latest/010_user.html
Brief CORAL EA module overview

- Lmod filters modules based on compiler and MPI loaded
  - ‘ml avail’ display on 4/11/17 (L means loaded, D means default):

  --- /usr/tcetmp/modulefiles/Compiler/xl/beta-2017.03.28  
  spectrum-mpi/2017.04.03 (L)

  --- /usr/tcetmp/modulefiles/Core

  StdEnv  (L)  git/2.9.3  pgi/17.3  (D)
  clang/coral-2017.03.15  (D)  gmake/4.2.1  totalview/2016.07.22
  clang/3.9.1  gsl/2.3  totalview/2017.0.12  (D)
  cmake/3.7.2  makedepend/1.0.5  xl/beta-2017.03.28  (L,D)
  gcc/4.8-redhat  pgi/16.10  xl/2016.12.02
  gcc/4.9.3  (D)  pgi/17.1

- ‘ml xl’ will load xl/beta-2017.03.28 (on 4/11) (‘ml xl’ done by default)
  - ‘ml xl/2016.12.02’ would switch to older GA version (not recommended)
  - ‘ml clang’ would switch to clang/coral-2017.03.15
  - Would switch to appropriate copy of spectrum-mpi/2017.04.03 for you
You can get at original IBM bits but you don’t get them by default
— Multiple compiler and MPI versions not seamless supported by IBM
— Our CORAL EA environment different from IBM’s (ORNL not IBM’s env either)

Made clang’s --gcc-toolchain option supported by XL and MPI
— We also set default to gcc 4.9.3 instead of RHEL system default of 4.8.5
— ‘--gcc-toolchain=/usr’ option sets it back to RHEL system default, if needed

Added --gpu extension for OpenMP 4.5 + GPU support (i.e., clang++-gpu)
— XL: Adds ‘-qsmp=omp –qoffload’
— Clang: Adds ‘-fopenmp -fopenmp-targets=nvptx64-nvidia-cuda’
  • Also made ‘-qoffload’ map to ‘-fopenmp-targets=nvptx64-nvidia-cuda’ for clang
— Using full ‘real’ options in makefiles recommended for portability

Enable rpaths (needed for HPC) by removing --enable-new-dtags options
— OpenMPI and XL added by default in non-overridable way (changes requested)

Our wrappers add appropriate compiler and MPI rpaths for you
— We also pick the appropriate CUDA libraries for that compiler version
— -vvv shows MPI wrapper output and -vvvv shows compiler wrapper output
Example of compiling and running MPI OpenMP 4.5 GPU code on CORAL EA

- Example: Allocate interactive backend node for GPU access and switch to clang
  - No GPUs on login nodes!

rzmanta23{gyllen}9: bsub -x -n 20 -Is -XF -W 60 -G guests /usr/bin/tcsh
Job <5592> is submitted to default queue <pbatch>.
<<ssh X11 forwarding job>>
<<Waiting for dispatch ...>>
<<Starting on rzmanta39>>

rzmanta39{gyllen}2: ml clang
Lmod is automatically replacing "xl/beta-2017.03.28" with "clang/coral-2017.03.15"

Due to MODULEPATH changes the following have been reloaded:
  1) spectrum-mpi/2017.04.03

rzmanta39{gyllen}3: mpiclang-gpu mpihasgpu.c

rzmanta39{gyllen}4: mpirun -n 4 mpibind ./a.out
Rank  3 Host rzmanta39  Able to use GPU 3  CPUs 120
Rank  0 Host rzmanta39  Able to use GPU 0  CPUs 0
Rank  2 Host rzmanta39  Able to use GPU 2  CPUs 80
Rank  1 Host rzmanta39  Able to use GPU 1  CPUs 40
Migration from /usr/tcetmp to /usr/tce planned

- When done (in a few months), we will switch to using /usr/tce
  - Automated RPM buildfarm replication of /usr/tcetmp environment
  - Facilitates syncing of RZMANTA, RAY, and SHARK environment
  - All the install rules in RPM scripts, so install procedure codified
    - RPM build scripts non-trivial, this is why it may take months to get right

- Please don’t manually switch to /usr/tce yourself
  - We will be iterating in /usr/tce to get it the same as /usr/tcetmp
    - So may break randomly if you use /usr/tce directly
  - Plan is for /usr/tcetmp to slowly convert to symlinks to /usr/tce
    - So existing /usr/tcetmp paths expected to work after migration
    - Will only do after we think that specific package on /usr/tce is correct
      - gcc-4.9.3 already lives in /usr/tce and will see in ldd output now

- Plans may change as we get knee deep into the /usr/tce port
How to determine what the MPI wrappers added?

- Use `-vvv` to show how LLNL’s wrapper called system MPI wrapper
  
  \[\text{rzmanta23\{gyllen\}212: mpiclang++-gpu mpihasgpu.cc -vvv}\]

  spectrum-mpi-wrapper executing after fixup:
  + export OMPICXX=/usr/tcetmp/packages/clang/clang-coral-2017.03.29/bin/clang++-gpu
  + exec /usr/tcetmp/packages/spectrum-mpi/spectrum-mpi-2017.04.03/bin/mpicxx
    -Wl,-rpath, /usr/tcetmp/packages/spectrum-mpi/spectrum-mpi-2017.04.03/lib
    mpihasgpu.cc

- Use `–show <source_file>’ to show final MPI wrapper output
  — Spectrum-MPI and OpenMPI require source file for accurate –show output

  \[\text{rzmanta23\{gyllen\}213: mpiclang++-gpu mpihasgpu.cc –show}\]

  /usr/tcetmp/packages/clang/clang-coral-2017.03.29/bin/clang++-gpu
  -Wl,-rpath, /usr/tcetmp/packages/spectrum-mpi/spectrum-mpi-2017.04.03/lib
  mpihasgpu.cc
  -I/usr/tcetmp/packages/spectrum-mpi/spectrum-mpi-2017.04.03/include
  -pthread
  -L/usr/tcetmp/packages/spectrum-mpi/spectrum-mpi-2017.04.03/lib
  -lmpiprofilesupport
  -lmpi_ibm
How to determine what the LLNL-specific compiler wrappers did?

- LLNL’s –vvvv option shows what wrapper did to compile line
  - Also triggered with traditional –v option

rzmanza23{gylten}211: clang++-gpu hasgpu.cc -vvvv

clang-wrapper executing after fixup:
+ exec /usr/tcetmp/packages/clang/clang-coral-2017.03.29/ibm/bin/clang++
  --gcc-toolchain=/usr/tce/packages/gcc/gcc-4.9.3
  -I/usr/tcetmp/packages/clang/clang-coral-2017.03.29/ibm/omprtl
  -L/usr/local/cuda-8.0/lib64
  -L/usr/local/cuda-8.0/nvvm/libdevice
  -L/usr/tcetmp/packages/clang/clang-coral-2017.03.29/ibm/omprtl/lib
  -rpath /usr/tce/packages/gcc/gcc-4.9.3/lib64:\
    /usr/tcetmp/packages/clang/clang-coral-2017.03.29/ibm/lib:\
    /usr/tcetmp/packages/clang/clang-coral-2017.03.29/ibm/omprtl/lib:\
    /usr/local/cuda-8.0/lib64
  -fopenmp
  -fopenmp-targets=nvptx64-nvidia-cuda
  hasgpu.cc

- Pass –vvvv to MPI wrapper to see final full compile line with MPI