Sierra EA software status and plans

B453 R1001

Presented by John Gyllenhaal

February 8, 2018

Lawrence Livermore National Laboratory

EA Compiler/MPI scheme works the same on Sierra

- Compilers auto-select GPU target based on compile host
  - Can use LLNL-specific –qv100 option to cross compile for Sierra on EA systems
    - Many executables built on RAY/RZMANTA used during December Sierra acceptance
    - Cross-compiling works only for specific XL, clangs, and MPIs right now
    - Currently –qv100/-qp100 option doesn't specify cpu target optimizations
  - Must pick GPU (v100/p100) at compile time due to GPU runtime inlining
    - Inlining key to getting high GPU performance with OpenMP 4.5
  - More cross-compiling details when first early users get on Sierra in March
- Monthly beta compiler drops will continue
  - Clang EA updates held up on CUDA 9.1 kernel update for CORAL EA
  - CUDA 9.1 kernel update held up on Spectre/Meltdown Power kernel updates
  - Hope to schedule CUDA 9.1 kernel update later this month
- Most of April 19, 2017 Compiler/MPI tutorial still applies today
  - https://computing.llnl.gov/tutorials/CORAL-EA/

Current compiler guidance

- Recommend use XL FORTRAN for all FORTRAN
  - Much higher performance than gfortran
  - Supports OpenMP 4.5 GPU offloading and CUDAFORTRAN (xlcuf)
  - Discontinued xlflang efforts (may be used for llvm tool development testing)
- Recommend use XL or Clang for C/C++
  - Both now have similar GPU OpenMP 4.5 runtime inlining and debug info
  - XL may be currently much faster for long double math (investigating app issue)
  - Use whichever works better for your app
- Non-export controlled reproducers required for getting compiler fixes
  - Simple code examples based on issue descriptions rarely reproduce issue now
  - We are doing complicated things no one thought about in the OpenMP Spec
    - OpenMP/CUDA in multiple files often required to trigger issues
  - I can help you generate these reproducer or train you how to generate them
    - User-generated reproducers are reported and solved much faster

Shift from mpirun to jsrun coming soon

- jsrun designed from ground up to support regression testing and UQ
  - Has a very different design focus and thus interface than srun and mpirun
    - Can specify number of GPUs, CPUs, and memory needed per MPI task in a run
    - Can specify complex sharing of GPUs and CPUs among multiple tasks in a run
    - Designed to pack multiple simultaneous runs onto node allocation per user specification
    - IBM plans develop jsrun wrappers later to emulate an interface closer to mpirun/srun
  - ORNL has good documentation on the new bsub 'easy mode' and jsrun:
    - https://beta.olcf.ornl.gov/for-users/system-user-guides/summit/running-jobs/
  - New salloc-like recipe: bsub -nnodes 3 -W 60 -Is -XF -G guests /usr/bin/tcsh

- jsrun (beta Feb 5, 2018 release) just became usable on Sierra test system
  - Appears functional enough for use on EA systems after we kick the tires more
  - jsrun is beta software and IBM is working on some known issues
    - Currently slow to launch larger jobs
    - Does not provide enough hooks to enable mpibind functionality

- Sierra uses a few shared launch nodes for running bsub scripts/jsruns
  - Like BG/Q, need to launch commands with jsrun to actually run on compute nodes
  - Will like convert CORAL EA systems to same model with jsrun rollout

- Plan to make CORAL EA systems more like Sierra in perhaps March 2018
  - Everything will need to be rebuilt with new MPI for jsrun rollout

mpibind package's 'lrun' fills beta jsrun gaps

- lrun written by Edgar Leon to enable mpibind under beta jsrun
  - Binding key for many apps to get good and predictable performance

- lrun wraps jsrun and provides 'good' common case interface and binding
  - Works around missing binding hooks expected in future (perhaps April) jsrun drop
    - lrun automatically invokes mpibind (do not call mpibind explicitly with lrun)
  - Generates 'normal' jsrun options when possible (i.e., perfectly uniform distributions)
    - Generates a desired 'spread out' mpi task map automatically otherwise
  - Should not be used with regression tests or UQ harnesses (that is jsrun's strength)
    - Also will get poor binding and mappings since lrun assumes one jsrun per allocation

- Usage: specify -N<nodes> and either -p<tasks> or -T<tasks_per_node>
  - No spaces between lsrun or jsrun arguments and values (i.e., -N3 not -N 3)
  - Running app mpitest on 3 nodes of allocation and 10 MPI tasks would be
    - lrun -N3 -p10 ./mpitest
  - Environment variable MPIBIND enables verbose output
  - Evolving, full documentation coming soon

Lawrence Livermore National Laboratory