Initial development environment on CTS-2 (TOSS4) and some proposals to improve it
# CTS2 initial environment is focused on Intel compilers and Mvapich2

- Run ‘ml avail’ or ‘module avail’ to list available modules (‘man ml’ for details)
  - LC uses lmod module hierarchies, so you must have a compiler loaded to see MPI modules available
  - CTS2 default similar to CTS1 env but newer versions `intel-classic-tce/2021.6.0 mvapich2-tce/2.3.6`
  - New with CTS2: we now provide both portable and ‘LC magic’ versions of dev environment
- Snapshot of modules available on CTS2 on 9/6/2022 with defaults loaded:

```plaintext
---
/usr/tce/modulefiles/Compiler/intel-classic-tce/2021.6.0
mvapich2-tce/2.3.6 (L)  openmpi-tce/4.1.2
---
/usr/tce/modulefiles/toolchains/Core
gcc-tce/10.3.1          intel-classic-tce/2021.6.0 (L,D)  intel-classic/2021.6.0 (D)  rocm/5.1.1
intel-classic-tce/19.0.4  intel-classic/19.0.4        intel-tce/2022.1.0            rocm/5.2.0
intel-classic-tce/19.1.2  intel-classic/19.1.2        intel/2022.1.0                rocm/5.2.1 (D)
---
/usr/tce/modulefiles/Core
StdEnv  (S, L)        cmake/3.19.2    cmake/3.23.1 (D)  ninja/1.10.2  patchelf/0.13 (D)  subversion/1.14.1
        cloc/1.84       cmake/3.21.1    git/2.29.1 (D)  patchelf/0.10   python/2.7.18
        cmake/3.14.5   cmake/3.22.4    git/2.31.1      patchelf/0.12   python/3.9.12 (D)
---
```

---

*Note: The contents of this document are for informational purposes only and may not be edited or altered in any way.*

---

*Lawrence Livermore National Laboratory  LLNL-PRES-838657*
Summary of differences between ‘LC magic’ (-tce) and portable (without -tce) versions of compilers and MPI

- ‘LC Magic’ versions of compilers/MPI ignore environment and modules minimizes env changes
  - Modules do not modify LD_LIBRARY_PATH, instead compiler wrappers adds extra RPATHs to link line
    - Goal is to get same shared libraries app was built with no matter what modules loaded
  - Modules do not set CC, CXX, F77, FC, MPICH_CC, MPICH_CXX, MPICH_F77, MPICH_FC env variables
    - Setting CXX in default modules tends to break build system subshells, since may redefine what build system specified
  - MPI wrappers (mpicc, mpicxx, mpi90, etc) ignore MPICH_CC, MPICH_CXX, MPICH_F77, MPICH_FC
    - Full path to MPI wrapper determines exactly which compiler is used, good for reproducible builds but non-standard!

- Portable versions of compilers/MPI uses environment and modules may change environment
  - Modules set LD_LIBRARY_PATH and may rely on same modules being loaded at run time as compile time
    - Can make it complicated to get right if running different apps compiled with different modules
  - MPI wrappers (mpicc, mpicxx, mpi90, etc) use MPICH_CC, MPICH_CXX, MPICH_F77, MPICH_FC settings
    - Build system/spack can specify exactly which compiler to use even if not in path. Many build systems expect this!
  - Users may need to manually add additional RPATHs to link line for consistent behavior
    - Many users end up using shortcut of hardcoding compiler and mpi modules into their dot files to workaround this issue
  - Spack builds these portable versions in /usr/tce, so dev env reproducible elsewhere
Mapping compilers/MPI from CTS1 (TOSS3) to CTS2 (TOSS4)

- TOSS3/CTS1’s gcc is 4.9.3 and TOSS4/CTS2 uses gcc/10.3.1 for robust C++17 support
- TOSS3/CTS1’s ‘intel/19.1.2’ module maps to ‘intel-classic-tce/19.1.2’ on TOSS4/CTS2
  - intel-classic is Intel’s name for the older compilers that were the workhorse on TOSS3
  - Avoid intel-classic-tce/19.0.4 since it does not support gcc/10.3.1 base (uses gcc/8.3.1 instead)
- TOSS3/CTS1’s ‘intel/oneapi.2022.1’ module maps to ‘intel-tce/2022.1.0’ on TOSS4/CTS2
  - This is Intel’s new clang-base ‘intel’ compiler, but clang-derived fortran compiler not yet stable
  - We recommend using very stable intel-classic instead, especially if you plan to compile any fortran code
- TOSS3/CTS1’s ‘mvapich2/2.3.6’ module mostly maps to ‘mvapich2-tce/2.3.6’ on TOSS4/CTS2
  - Same mvapich2 but the -tce version currently ignores MPICH_CC, MPICH_CXX, etc. so not perfect mapping
  - ‘LC Magic’ follows rules used on CORAL1 but not TOSS3 due to use on CORAL2, see proposals to change
- CTS2 Systems currently defaults to intel-classic-tce/2021.6.0 mvapich2-tce/2.3.6
Flux is replacing Slurm as resource scheduler and job launcher soon

- A subset of CTS2 will start with Flux and not Slurm. Eventually all clusters will run Flux.
  - RZWHIPPET (CTS2), TIOGA (EAS3/CORAL2), CORONA run flux natively today
  - Goal for ‘friendly users’ to help us find and fix limitations with flux relative to Slurm

- My ‘mnemonic’ for mapping to flux: ‘flux mini ‘ replaces the ‘s’ in three key slurm commands
  - `salloc` -> `flux mini alloc` // `flux mini alloc -N 2 -t 60m` (currently default time units is seconds)
  - `srun` -> `flux mini run` // `flux mini run -N 2 -n 4 ./a.out`
  - `sbatch` -> `flux mini batch`

- Other slurm commands that don’t follow ‘flux mini’ rule however
  - `sinfo` -> `flux resource list`
  - `squeue` -> `flux jobs -A`
  - See ‘man flux mini’ and ‘man flux jobs’ for more details

- Slurm emulation wrappers available via ‘ml flux_wrappers’
  - `salloc`, `srun`, `sbatch`, `sxterm`, `squeue`, and `showq` (I have requested a `sinfo` wrapper)
Recent feedback has LC exploring different configurations of development environment. Some current proposals...

- Instead of -tce on compiler package name, place -llnl at end of compiler version instead:
  - LC magic designator `intel-classic-tce/2021.6.0` becomes `intel-classic/2021.6.0-llnl`
  - Advantage: Change has ‘ml avail’ group portable and magic versions under same package name ‘intel-classic’

- Instead of -tce on MPI package name, use compiler ‘magic’ mode to trigger MPI magic
  - Advantage: Change allows same MPI package name to work in both modes without reloading MPI
  - New ‘magic’ version default could become: `intel-classic/2021.6.0-llnl mvapich2/2.3.6`

- Stop setting CC, CXX, FC, F77 (and don’t set MPICH_CC, etc.) in portable version modules
  - Setting CC, CXX, FC, F77 breaks complex builds systems for huge codes due to subshell builds
  - Goal of reducing use of LC Magic to only those that need magic by making portable version default instead
  - If CC, etc. removed, new default could become: `intel-classic/2021.6.0 mvapich2/2.3.6`

- Suggestions and feedback welcome! We are exploring many proposals now.
Burning Questions?