Methods to seamlessly mediate conflicts between the incongruous software stacks present on HPC systems

Working Group
Scalable Tools Workshop 2023
Lead: Jonathon Anderson
http://bit.ly/sw-stw23

Participants

- Jonathon Anderson (Rice)
- Wileam Phan (Rice)
- Dragana Grbic (Rice)
- Sameer Shende (Oregon)
- Jim Kupsch (Wisconsin)
- Josef “Bolo” Burger (Wisconsin)
- Hsuan Heng-Wu (Wisconsin)
- Paul Ferrell (LANL)
- Timothy Goetsch (LANL)
- Franklin Keithley (LANL)
- Ben Woodard (Red Hat)
- Matt Legendre (LLNL)
- Nathan Tallent (PNNL)
- Mahesh Rajan (Trenza)

Spack gripes

- Problems finding external packages and compiler
  - System **`libssl`** vs Spack-built **`libssl`**
  - Doesn't work with oneAPI components
  - Erroneously detects multiple LLVM packages (**`llvm`**, **`llvm-doe`**, **`llvm-amdgpu`**)
- Automatic Cray modules detection
  - Need to manually enter modules in packages.yaml
  - Resulting **`compilers.yaml`** is huge
- Issues with concretizer reuse (v0.20)
  - Doesn't work when binary cache changes
  - Can pick up packages built with the wrong compiler
  - Any small change will cause a re-hash
  - ABI compatibility checkers (e.g. **`libabigail`**) would fix this
- Facility-provided binary cache
  - Users need to follow the exact same prescriptions, otherwise it won't work

Spack resources

- GitHub repo for sharing Spack configs: https://github.com/spack/spack-configs
- E4S tracks `spack.yaml` configs for most DOE facilities: https://dashboard.e4s.io/
- OLCF Spack environments tutorial (outdated): https://docs.olcf.ornl.gov/software/spack_env/index.html
- NERSC Spack configs GitHub mirror repo: https://github.com/NERSC/spack-infrastructure

Embracing containers in HPC? Pros

- Used a lot in the cloud environment
- Portable (build once, run everywhere)
- Success stories at E4S

Embracing containers in HPC? Cons - 1

- Not enough portability across different HPC systems
- Mismatched **`glibc`**/**`libstdc++`** versions
  - e.g. ROCm containers vs system **`glibc`**
  - Proposed solution: **`glibc`** execution introspection at library load time
  - Work in progress: Fedora Silverblue but for containers (currently in internal CI at Red Hat)
- GPU kernel driver mismatch
  - NVIDIA solved this with OCI hooks and driver compatibility table
  - AMD and Intel need to catch up
  - Also happens with other kernel drivers and support libraries (e.g. **`sep5`** for VTune)
  - Need tools to check kernel driver ABI compatibility with userspace
- Semantic versioning lies
  - Vendors tend to change stuff without telling anyone, esp. at testbeds

Embracing containers in HPC? Cons - 2

- Often contains unoptimized builds (e.g. MPI built with no fabrics)
- Matching MPI stacks are non-trivial
  - e.g. ROCm container + Cray MPICH + fabrics
  - MPICH ABI specifications vs OpenMPI implementation
  - See **e4s-cl** for a possible solution
  - Some progress in MPI specs 4.2, but needs more work
  - Generic/"fat-binary" fabric drivers can be useful
  - Can also use compatiblity layers to the host's MPI stack, but there are overhead and security issues
- Things keep evolving – hard to track all of them
- Need tools for debugging processes inside containers

Miscellaneous war stories

- Omni machine at Oregon Frank (all 3 GPU vendors in one machine)
  - Problems with Intel/AMD open source vs NVIDIA proprietary OpenCL stacks
  - Resizable BAR (needed for Intel GPUs) unavailable on servers
  - Eventually gave up

Action items

- `libabigail` is almost ready, just needs more beta testers
- Research topic for (masochistic) PhD student:
  how to match kernel drivers with userspace
- Push AMD and Intel to provide driver compatibility tables
  (hardware support and runtime support)
- Continue efforts in MPI Forum towards a uniform MPI ABI