Maintaining Large Software Stacks in a Cray Ecosystem with Gentoo Portage

Colin MacLean
Large Software Stacks

• Most modules are, at most, a handful of packages
• Currently available tools ill-suited for large comprehensive sets of packages on Cray
  – Most notably Python
• Need tools to build software distributions
  – Resolve complicated dependency trees
  – Periodic releases of collections of packages, not many individual modules
    • Avoid complex runtime dependencies
    • Avoid environment bloat
Use of Gentoo Prefix on Blue Waters

- Gentoo Prefix used to manage the Python stack on Blue Waters
  - Over 400 packages installed
    - Includes non-Python dependencies. Some system libraries too old.
    - Built against the optimized cray-mpich and libsci libraries
    - Both MPI and non-MPI versions with minimal redundancy
    - Most ebuilds work without modification
      - 18 modified (mostly minor)
      - 6 new
Potential Package Managers

- **Pip**
  - Original scope was limited to a Python stack
  - Can’t build non-Python dependencies
  - Can’t tweak build environment on a per-package basis

- **SWTools**
  - HPC software build and install tool
  - Creating build script too complex for large dependency trees
Potential Package Managers

• EasyBuild
  - HPC software management tool
  - Basic level of dependency calculation
    • Acceptable for a handful of packages
    • Inadequate for hundreds of packages
    • Hard coded versions and configuration options in build files
  - Has package repository

• Anaconda
  - Python distribution
  - Primarily binary based, but can build from source
    • Hard to control build environment
  - Anaconda binaries using MPI need rebuild to use cray-mpich
Prefixed Gentoo Portage

- Gentoo Portage Alt project allows Gentoo GNU userland install on top of Linux/Unix hosts
- Dependency calculations based on range of compatible versions. Suitable for rolling periodic distributions.
- Portage USE flags
  - Allow for fine-grained control of configuration options
  - Used in dependency calculations
- SLOTs
  - Feature allows for simultaneous install of multiple versions of some packages

- SUBSLOTs
  - Feature used to trigger required rebuilds on package updates
- Ebuild files
  - Bash scripts with access to powerful eclass APIs
  - Eclasses can provide default build procedures that may also be heavily modified if necessary
- Repositories with tens of thousands of ebuilds available
- Overlay repositories allow for customized ebuilds
- Consistent default build environment. Per-package environment customization.
Package Manager Selection

• Gentoo Prefix chosen
  - Make Environment Modules aware
  - Easier than significantly improving package management features
  - Need to install more than just Python packages
Initial state of Gentoo Prefix

- Bootstrapping script self-sufficient
  - Bootstrapped its own gcc/binutils
  - Host programs not available in prefix environment
  - Host MPI only major library visible to prefix (through sys-cluster/native-mpi package)
- After bootstrap, environment still constrained to prefix
- Unaware of Environment Modules framework
Compiler Environment for Gentoo Prefix

- Unpatch Gentoo Prefix uses GCC compiled with --sysroot
  - Need to use host compilers
  - --sysroot won’t work
- Add prefix directories via -I and -L options to compiler flags
  - Breaks some packages
- Set CPATH and LIBRARY_PATH
  - Searched after -I and -L like system paths
- Provide compiler a modified “specs” file
Step 1: Modify Bootstrapping Script

- Modify environment to allow Prefix to see host compilers and libraries
- Change header and library search method
  - Generate CPATH/LIBRARY_PATH for host compiler.
  - Prefix had expected gcc built with --sysroot option.
- Generate rpaths for LDFLAGS
- Remove checks for host paths in environment
- Set up basic default Environment Modules
Step 2: Modify Prefix Environment

- Gentoo Prefix loads its environment from $EPREFIX/etc/profile
  - Fix paths to make host system visible to Prefix environment
  - Load default set of environment modules if Eselect Environment Modules not available
- Gentoo Portage executes in non-interactive non-login Bash shell
  - Source $EPREFIX/etc/profile
- Add Prefix paths to CPATH and LIBRARY_PATH in $EPREFIX/etc/env.d
- Create Portage profile for Blue Waters
  - Default compiler flags, USE flags, forced “cray” USE flag, package.provided of host provided dependencies
Enhancement: Environment Modules Awareness

- Environment Modules module for Eselect
  - Sets default set of Environment Modules for the Prefix
  - Generates environment file
    $EPREFIX/etc/env.d/01modules
  - Stores settings in $EPREFIX/etc/env-mod.conf
  - Regenerate environment with `eselect envmod update`
  - Ensures reproducible build environment
Enhancement: Environment Modules Awareness

- **Environment Modules in Portage**
  - Sometimes, an odd package may require a different build environment
  - Patch Portage to load, unload, and swap modules based on environment variable
    - Use package.env
  - Prevent build of packages known to break with certain module configurations
  - Add module requirements for packages

- **ENVMOD**
  - Load: ENVMOD="module"
  - Unload: ENVMOD="-module"
  - Swap: ENVMOD="%module1:module2"
  - Space separated list

- **ENVMOD_RESTRICT (ebuild)**
  - Space separated list of forbidden modules

- **ENVMOD_REQUIRE (ebuild)**
  - Space separated list of required modules
Enhancement: Prefix Chaining

• Based on old patch to Portage
• Creates a child prefix
• Minimal setup
  – $EPREFIX/etc/profile
  – $EPREFIX/etc/portage/*
• Chain setup script
  – Generates etc/profile and etc/portage/* based off parent prefix
  – Optionally generates and installs environment module into parent prefix

• Portage uses parent prefixes for dependency resolution
  – Specify DEPEND, RDEPEND, HDEPEND, PDEPEND
  – Allows for a chained prefix with independent runtime dependencies
• Interaction with parent prefix entirely read-only
Enhancement: Prefix Chaining: Different Configurations

- Chained Prefixes can change any Portage build setting
  - USE flags
  - Compiler
  - Even CHOST, if desired
- Can have different default Environment Modules
- Can be used to support multiple BLAS choices

On Blue Waters
- BWPY: Built for login nodes. No MPI.
- BWPY-MPI: h5py rebuilt with USE="mpi". Mpi4py added.
- BWPY-Tensorflow: Required beta Google-Protobuf. Installed in chained prefix to keep stable BWPY.
Multiuser: Create chained prefixes in home directories

- Users can build specific versions and configurations of software using ebuilds (ex: PETSc)
• Add “cray” to ebuild’s IUSE variable
• Put Cray platform-specific modifications in “if use cray” block
• Don’t break ebuild for non-Cray platforms