Build Systems Wrap Up

Todd Gamblin
Martin Schulz
Drew Bernat
Bronis de Supinski
Mike Fagan
Jim Galarowicz
Jeff Hollingsworth
Madhavi Krishnan
Al Malony
Bernd Mohr
Dave Montoya
Dan Quinlan
Bill Williams
Felix Wolf

CScADS Workshop on Tools 2011

We identified three major issues with build systems and software packaging

- Build Environment Issues
  - **Affects:** developers
  - Ease of use
    - Learning curve, configurability
  - Support for HPC platforms, as well as desktop environments
  - Integration of component builds with others
    - How do builds learn about dependencies?
- Packaging issues
  - **Affects:** users, system maintainers
  - How to bundle software for package management systems
  - Easy installation on production machines
- Versioning
  - Need for build-time and runtime version querying

Topics Discussed

- Build systems currently in use
  - **Custom shell scripts and Makefiles:**
    Scalasca, TAU, Dyninst
  - **Autotools:**
    ROSE, HPCToolkit, Open|SpeedShop, Dyninst (autoconf-ish), Score-P
  - **CMake:**
    LLNL Performance Tools, ROSE, CBTF, Marmot

- **Many projects are trying out CMake**
  - LLNL Tools, ROSE, CBTF, Marmot
  - some traction within Dyninst group

Issues with existing build systems

- Many custom build systems evolved because they predate Cmake/ Autotools
  - Need custom support for HPC hardware
  - Libtool broken
  - Autoconf doesn't handle many hpc machines
  - Old versions of Cmake not robust enough

- Reasons for not revamping build systems
  - Inertia, no time to rewrite code
  - Need to support lowest common denominator
  - Yet another dependence (i.e. Building Cmake)
  - Autotools too complex
    - Writing m4 macros and generating configure is time consuming and error-prone

Build environment issues

- Need rapid integration of component modules
  - Set of build-time queries one has to do is growing
    - Many variables involved in using a library
    - Need libraries to expose their own information, dependencies
- Want to reduce this to one simple variable
  - Where does this thing live?
  - Deduce the rest automatically according to some standard
- Proposal: use a standard set of useful information about software to include in distributions
  - Basic build info (libraries, includes, flags, etc.)
  - Dependence information
    - What dependencies and what versions of them do I need?
    - Conflicts with other libraries
    - What compilers did I use? (C++, MPI, thing without ABIs)
  - **Action Item:** Todd will make an initial "spec", and a test validator for it
    - Group will hammer out the details

Packaging Issues

- Affects users of the software
  - Production environments need to bundle and keep software up to date on many machines
  - Can make life easier for developers
  - Don't have to build tools they don't intend to change

- Proposal: HPC package repository for tools
  - Package server with packages for RPM, Deb, and MacPorts
  - Action Item: Find a hosting location for this
    - Jeff volunteers UMD if noplace else works.
  - Action Item: Developers should post detailed instructions on how to make different types of packages to a wiki

Versioning Issues

- Libraries need way to check version at both build and runtime
  - Allows workarounds for version –specific issues that make it into production
  - Allows graceful fail at runtime if incompatible dependent libraries are detected
- Proposals:
  - **Action Item:** Todd will make a list of recommendations for version information to be included with installed config headers
  - **Action Item:** Todd and Drew come up with naming convention/ recommendations for runtime version query routines
    - These are extensions of existing build recommendations
  - Builds should allow forcing things if these constraints are too strict
    - --force, --dammit, etc.

Conclusions

- **Build discussion was very useful**
  - Allowed groups to become familiar with others' systems and their issues
  - Came up with proposals for standard ways to enable components between build systems
    - Exporting library information
      - Enable finding components more easily
    - Runtime, build time conventions
  - Got familiar with different types of packaging
    - Usefulness of packaging for users, sysadmins, and developers
    - Started discussion on having standard repositories

- **Build system discussion is useful**
  - Details usually swept under the rug in research environments
  - Ironing out the details revealed more commonality than differences
    - Same set of issues

- **Recommendation: Common tools wiki for further ironing out these details**