The Harness Workbench: Unified and Adaptive Access to Diverse High-Performance Computing Platforms

Al Geist, Christian Engelmann (Oak Ridge National Laboratory), Jack Dongarra, George Bosilca (University of Tennessee, Knoxville)
Vaidy Sunderam, Magdalena Sławińska, Jarosław Sławiński (Emory University)

Harness workbench core components
- Next-generation runtime environment
  - Flexible, adaptive, lightweight framework
  - Management of runtime tasks
  - Support for diverse HPC platforms

Research and development goals
- Increasing the overall productivity of developing and executing computational codes
- Optimizing the development and deployment processes of scientific applications
- Simplifying the activities of application scientists, using uniform and adaptive solutions
- “Automatically” supporting the diversity of existing and emerging HPC architectures

Harness workbench core technologies
- Automatic adaptation using pluggable modules
  - Harness Workbench Toolkit plug-ins
  - Runtime environment plug-ins
- Development environment and toolkit interfaces
  - Easy-to-use interfaces for scientific application development, deployment, and execution

Harness workbench toolkit (HWT)
- Unifying abstraction over diverse HPC resources
- Command-line and GUI tools
- Translation into fine-grained mappings of native toolchains
- Behavior considered in plug-ins
- Configurations through profiles
- T镜子 for end users

HWT porting assistant
- Facilitates source code adaptation through specialized plug-ins
- Suggests safe changes
- Highlights manual code replacement areas
- Guided by situation-specific profiles
- Prototype
  - Plug-ins as Python scripts
  - Ported CPMD across Jaguar (Cray XT), Phoenix (Cray X1), Cheyenne (IBM p650)
  - Example conversions: detection, function mappings, data type size changes

HWT virtualized command toolkit (VCT)
- Abstraction layer between users and platform-specific compilers, linkers, libraries, testing and debugging software, launching systems, etc.
- Enables users to issue generic build commands that are processed to produce a target-specific set of commands
- Pluggable modules to deliver back-end functionality

VCT prototype approach
- Target-specific knowledge encapsulated in profiles
- Generic build-related files
- Late binding – connecting generic build files at runtime

VCT prototype architecture
- Scenario
  1. End-user stage-in application source code with application’s generic build system on the front-end node to ‘wdir’
  2. HWT starts VCT server that mirrors ‘wdir’
  3. The build system is launched on the front-end node in ‘mirror’
  4. OS redirects filesystem operations to VCT server (e.g., reading Makefile)
  5. To concrete generic files VCT server interacts with HWT and, based on profiles, resolves vct references at runtime through size binding
  - Implementation details
    - VCT server exploits FreeBSD (Filesystem in User-space)
    - Python scripts
    - Tested on CPMD

HWT profiles
- Declarative descriptions created by vendors, administrators, developers, and adjusted by users
- Encapsulate target-specific knowledge at the system and application levels
- Support mechanisms
  - Interception
  - Dynamic recursive resolution to allow switching among predefined settings

HWT virtualized environments
- Problem
  - Application dependencies may cause conflicts with system-wide installed libraries.
- Solution
  - Use co-existing, alternative user-space installations.
- Approach
  - Provide isolated installation environments (“sandboxes”). These can inherit from one another to build nested hierarchies.

Configurable “sandboxes” for scientific applications
- Virtualized adaptation of system properties to actual application needs
- System and runtime environment virtualization
  - Install environment
  - Net env start-conf application
  - Tunable scientific application portability for day-one operation
  - HWT for adaptation of application to HPC system properties
  - Virtualized environment for adaptation of HPC system to application needs
  - Tunable solution, where applications can move from one HPC system to another with initial minimal changes enabling day-one operation
  - Continued adaptation to actual target platform properties assumes eventual performance gains by gradually taking advantage of new system features

VCT prototype use-case

HWT profiles

Cray XT architecture

Hardware profile

Jaguar

Red Storm

System profiles

User profile

HWT profiles

VCT prototype approach

VCT prototype architecture

VCT prototype use-case

HWT virtualized environments

Configurable “sandboxes” for scientific applications

Tunable scientific application portability for day-one operation

Point of Contact: Al Geist (geist@ornl.gov)
www.csm.ornl.gov/harness | id.cs.utk.edu/harness | dd.mathics.emory.edu

Office of Science
U.S. Department of Energy

Oak Ridge National Laboratory

EMORY UNIVERSITY