JÜLICH
FORSCHUNGSZENTRUM

UNITE

UNiform Integrated Tool Environment
https://apps.fz-juelich.de/unite/

Oct 2013 | Bernd Mohr

Motivation

- Tools are hard to install
  - Almost all depend on MPI library
    - Some even on the compiler
  - Need to be downloaded from a myriad of sites
  - Every tool (seems to) use different configures
  - More and more tool component can be re-used by other tools
  - etc

- Tools are hard to use
  - Of course they are complex powerful programs …
  - But every site installs them differently
    - Makes writing good tool documentation hard
  - etc

UNITE

- **Goal:**
  - Provide portable common access to parallel performance tools
  - Lower bar for inexperienced users and admins
- **Basic ideas:**
  - Standardized and simplified **tool installation**
    - **Meta-Installer** (providing “meta” configure/make/make install)
  - Standardized **tool access**
    - Based on “module” command (www.modules.org)
    - Standardize module names and structure (e.g. help)
    - Standardize tool, compiler, MPI library names
    - Activate by “**module load UNITE**”
  - **But** flexible enough to allow co-existence with site policies

Definitions and Standard Names

- **Package** ::= product, tool, or component which
  - Is available / can be used / can be installed as separate entity
  - Three basic sorts of packages: Tools, Utility, Devel
  - Typically comes in multiple versions
  - Example: vampir, scalasca, marmot, ...
- **Version**
  - <MajorVersion> . <MinorVersion> [.<Plevel>] [(rc|b)<Number>]
  - Example: 2.1b2
- **Specialization** ::= Optional constraints
  - Which limit the applicability of a package and/or version
  - Currently mainly needed on Linux installations
  - Specified as: –<MpiLibrary>–<Compiler>–<Precision>
  - Unnecessary constraints are left out
  - Example: –openmpi–32bit

Installation Space Layout: Module Files

- Install required UNITE components together at **system-specific** installation path **UNITE_ROOT**

```
▶ ${UNITE_ROOT}/
   ▶ modulefiles/              # UNITE module files
      ▶ tools/ | utils/ | devel/
         ▶ <package>/
            ▶ <version>-<specialization>
      ▶ templates/             # "generic" module files
      ▶ scripts/               # basic scripts for templates

   ▶ bin/                      # admin utilities
   ▶ doc/                      # for overall UNITE docu
   ▶ licenses/                 # license files for packages
```

symlinks to templates

Installation Space Layout: Package Files

- Actual package are installed also under ${UNITE_ROOT}/packages
  [Note: if not feasible or to include historic installations, create symbolic-link trees to real installation directories]

  - ${UNITE_ROOT}/
    - packages/
      - *<package>*/
        - *<version>-<spezialization>*/
          - *<package-specific-sublayout>*

Example: "module help scalasca" Output

```
% module help scalasca

-- Module Specific Help for 'scalasca/1.0-mpibull2-intel-64bit‘ --

Scalasca:
Scalable Performance Analysis of Large-Scale Parallel Applications
Version 1.0 (for BullMPI 2, Intel Compiler, 64bit)

Basic usage:
1. Instrument application with skin = "scalasca -instrument"
2. Collect & analyze execution measurement with scan = "scalasca -
   analyze"
3. Examine analysis results with square = "scalasca -examine"

For more information:
- See ${SCALASCA_ROOT}/doc/manuals/quickref.pdf
  or type "scalasca -h"
- http://www.scalasca.org
- mailto:scalasca@fz-juelich.de
```

UNITE Package Meta-Installer

JÜLICH FORSCHUNGSZENTRUM

- **Knows how to configure, build, and install all UNITE tools**
  - Can handle GNU autoconf, cmake, TAU + Scalasca homegrown configure
- **Installs or updates set of tools**
  - According to UNITE installation space layout
  - Automatically creates necessary module files
  - Automatically handles package dependencies
    - Takes installed UNITE + system packages into account
    - Configures maximum tool integration, for example
      - Reusing PAPI, CUBE, OTF, DynInst, PDT components in Paraver, Scalasca, TAU, VampirTrace
      - Marmot / Vampitrace integration
      - Scalasca+VampirTrace integration with TAU

UNITE Package Meta-Installer

- Implemented as
  - **/bin/sh configure script**
    - Auto-detects compiler, MPI, PAPI, CUDA configuration
    - Checks prerequisites (module, QT, Fortran, cmake, wxwidgets)
    - User normally only specifies installation prefix
    - Also generates installed/to-be-installed package list
  - **Makefile** for basic build and install workflow
    - Re-uses make dependency checking + re-makes
  - **/bin/sh scripts** for (flexible) package configuration
    - `<tool>-configure-extra.sh`
      - For configuration of (optional) packages dependencies
      - Outputs corresponding configure options
- Minimal output
  - But keeps extensive log files for debugging

Supported Packages

- **UNITE-1.1**
  - Cube-3.4.3 + 4.1.6
  - Extrae-2.3
  - Lwm2-1.1
  - Marmot-2.4.0
  - Opari2-1.0.7
  - OTF-1.12.3 + OTF2-1.1.1
  - Paraver-4.4.1
  - Pdtoolkit-3.19.1
  - Scalasca-1.4.3 + 2.0b1
  - Score-P-1.1.1
  - TAU-2.22.2
  - UniMCI-1.0.1
  - Vampirtrace-5.14.3
  - Vampir-5.x or 7.x, 8.x
  - VampirServer-1.x or 7.x, 8x
- **Compiler**
  - gnu, ibm, intel, open64, pathscale, pgi, sun
- **MPI libraries**
  - hp, intel, intel2, intelpoe, mpibull2, bullxmpi, mpich, mpich2, mpich3, openmpi, platform, aixpoe, ibmpoe, sgimpt, sun
- **Work in progress**
  - SIONlib and CUDA support, MUST toolset

Example: UNITE configuration (I)

```
% ./configure --prefix=/opt/local/UNITE

Autodetect compilers...
=======================
INFO: Found /usr/bin/gcc
INFO: Found /opt/local/intel/composer_xe_2013.4.183/bin/intel64/icc
INFO: Using intel compiler; select GNU compiler using --compiler=gnu

Autodetect MPI library...
=========================
INFO: Found MPICH2 /home/local122/mpich-2.3.1p1/bin/mpicc

Autodetect PAPI...
==================
INFO: Found version 5 with -L/home/local122/papi-5.0.1/lib64 -lpapi
INFO: Using --with-papi=/home/local122/papi-5.0.1

...
```

Example: UNITE configuration (II)

```
...

Checking optional requirements...
================================
module command: OK
Qt library:     OK
wxWidgets:
  INFO: No suitable wxWidgets package found: need GTK based version
  INFO: Paraver will not be build.
Fortran:        OK

Packages
========
       unite: install package unite-1.1.tar.gz
        lwm2: 1.1-mpich2-papi already installed
        cube: 3.4.3-intel already installed
       cube4: install package cube-4.2.tar.gz
         PDT: 3.19-intel already installed
      opari2: install package opari2-1.1.tar.gz

...
```

Example: UNITE configuration (III)

```
...

         otf: install package OTF-1.12.4salmon.tar.gz
        otf2: install package otf2-1.2.tar.gz
    scalasca: 1.4.3-mpich2-intel-papi already installed
   scalasca2: install package scalasca-2.0.tar.gz
     Score-P: install package scorep-1.2.tar.gz
      marmot: 2.4.0-mpich2-intel already installed
      unimci: 1.0.1-marmot already installed
 vampirtrace: install package VampirTrace-5.14.4.tar.gz
      vampir: install package vampir-8.1.0-x86_64-setup.bin
vampirserver: install package vampirserver-8.1.0-x86_64-setup.bin
   libunwind: 1.0.1-gnu already installed
      extrae: install package extrae-2.3.4.tar.gz
     paraver: use existing module
  wxpropgrid: use existing module
       boost: 1.46.1-gnu already installed
         tau: 2.22.2-mpich2-intel-papi already installed
...
```

Example: UNITE configuration (IV)

```
...

CONFIGURATION COMPLETE
======================
  PREFIX:   /opt/local/UNITE
  PLATFORM: Linux_Cluster
  COMPILER: intel
  MPI:      mpich2
  PAPI:     /home/local122/papi-5.0.1

Next steps
==========
- cd BUILD-Linux_Cluster-mpich2-intel-papi
- make
- Install vampir license as /opt/local/UNITE/licenses/vampir.license
-  Install vampirserver license as
  /opt/local/UNITE/licenses/vampirserver.license
- cd <system-modulefile-directory>;
  ln -s /opt/local/UNITE/modulefiles/UNITE .
```

Example: UNITE build

```
% make
===== UNITE-1.1 =====
Reusing existing UNITE directory /opt/local/UNITE
Done.
===== Cube-4.2-intel =====
Unpack...
Config... (see LOGFILES/cube4/configure.log)
Make... (see LOGFILES/cube4/make.log)
Install... (see LOGFILES/cube4/install.log)
Done.
===== OPARI2-1.1-intel =====
Unpack...
Config... (see LOGFILES/opari2/configure.log)
Make... (see LOGFILES/opari2/make.log)
Install... (see LOGFILES/opari2/install.log)
Done.
===== OTF-1.12.4-intel =====
Unpack...
Config... (see LOGFILES/otf/configure.log)
Make... (see LOGFILES/otf/make.log)
...
```

Issues

- Constantly developing packages
  - API, command / file name changes
    - Not always backward compatible
  - New or dropped dependencies
  - System / context / compiler caused changes
    - e.g. /lib → /lib64
- How to maintain consistent module setup
  - Compiler / MPI / corresponding tools
  - Lmod?
- Frontend/backend systems (e.g. IBM BlueGene, Cray XK/XT/…)