UNITE Installation Guide

Bernd Mohr
Jülich Supercomputing Centre
May 2010

The goal of **UNITE** (**UN**iform **I**ntegrated **T**ool **E**nvironment) is to provide a robust, portable, and integrated environment for the debugging and performance analysis of parallel MPI, OpenMP, and hybrid MPI/OpenMP programs on high-performance compute clusters. It consists of a set of well-accepted portable, mostly open-source tools. This document describes the installation of the UNITE software and the UNITE debugging and performance tools. For an explanation on how to use the tools in UNITE see the UNITE User Guide.

This first version of the UNITE environment was developed in the EU ITEA2 project "Parallel Programming for Multi-core Architectures" (ParMA) and with support of the "Virtual Institute - High Productivity Supercomputing" (VI-HPS).

Contents

1. Introduction to UNITE

1.1. Why UNITE?

High-performance clusters often provide multiple MPI libraries and compiler suites for parallel programming. This means that parallel programming tools which often depend on a specific MPI library, and sometimes on a specific compiler, need to be installed multiple times, once for each combination of MPI library and compiler which has to be supported. In addition, over time, newer versions of the tools also get released and installed. One way to manage many different versions of software packages, used by many computing centres all over the world, is the "module" software (see: http://www.modules.org). However, each centre provides a different set of tools, has a different policy on how and where to install different software packages, and how to name the different versions.

UNITE tries to improve this situation for debugging and performance tools by

1. specifying exactly how and where to install the different versions of tool software packages (including integrating the tools to the maximum possible degree),
2. defining standard module names for tools and their different versions, and
3. supplying pre-defined module files which provide standardized, well-tested tool configurations,
4. **but** still being flexible enough to be able to co-exist with site-local installations, restrictions, and policies.

1.2. UNITE Terminology

This document uses the following convention:

Words written in `fixed-size font` are used for UNIX/LINUX commands, command options, and filenames. If such a word is enclosed in angle brackets (e.g., `<Item>`), it is a variable which is not supposed to be used verbatim but is to be replaced with items specified by `<Item>`.

UNITE also uses a few terms which are defined here:

Package:
A software product, tool, or component which has its own name and is available / can be used / can be installed as separate entity. UNITE distinguishes between different **types** of packages:
- the actual **tool** packages for debugging, validation, performance profiling and tracing,
- **utility** packages which are typically auxiliary stand-alone components and libraries used with the tools,
- and finally **development** packages which are only necessary for building the tools.

Version:
Basic version naming in the form "`<MajorVersion>.<MinorVersion>`" or "`<MajorVersion>.<MinorVersion>.<PatchLevel>`" optional followed by either "`b<Number>`" in case of a beta version or "`rc<Number>`" for release candidates. Examples: `2.1, 4.1.3, 3.2b2, 4.5rc1`

Specialization:
Optional constraints which limit the applicability of a version of a package. Used on computing systems which provide multiple MPI libraries or compiler suites (e.g., typically on Linux clusters). Takes the form "`-<MPILibrary>-<Compiler>-<Precision>`". The three components are always specified in this order, however unnecessary constraints (not provided by a system) can be left out. Example: `-openmpi-64bit`.

Full-Qualified Version (FQV):
A fully-qualified version is the complete specification of a version of a package installed under specific constraints, i.e., it is the combination of a version followed by a specialization separated by a dash ("-"). Example: `1.4.3-mpich2-intel`.

Platform:
The target computer system on which UNITE is installed. A platform is characterized by its vendor, architecture, and set of compiler suites and MPI libraries available for parallel programming.

1.3. Standard Names

The following table provides the standard names currently in use by UNITE for packages, MPI libraries, and compiler suites. The rule for UNITE standard names is to select the most common name for a package written in one word in all lowercase letters.

Tool Packages	ddt	ddt parallel debugger from Allinea
	hpctoolkit	sampling profiler hpctoolkit from Rice University
	itac	Intel Trace Collector and Analyzer
	marmot	Marmot MPI validation tool from HLRS Stuttgart / TU Dresden
	mpip	mpiP MPI profiler from LLNL
	ompp	ompP OpenMP profiler from NERSC
	scalasca	Scalasca parallel performance toolset from Jülich Supercomputing Centre
	tau	TAU parallel performance toolset from University of Oregon
	totalview	totalview parallel debugger from Totalview Technolgies
	vampir	Vampir event trace visualizer from TU Dresden
	vampirserver	VampirServer parallel event trace visualizer from TU Dresden
	vampirtrace	VampirTrace event trace collector from TU Dresden
Utility Packages	cube	Cube presentation tool from Jülich Supercomputing Centre
	papi	PAPI performance counter access library from University of Tennessee
Development Packages	cmake	cmake cross-platform makefile generator
	otf	OTF open trace format library from TU Dresden
	pdtoolkit	pdtoolkit parser utilty library and tools from University of Oregon
	qt	QT graphics library from Nokia
	vtf3	VTF3 trace format library from TU Dresden
	wx	wxWidgets graphics library
MPI Library	bgl	BlueGene/L MPI library
	bgp	BlueGene/P MPI library
	cray	Cray XT MPI library
	hp	Hewlett-Packard MPI library
	intel	Intel MPI library version 1
	intel2	Intel MPI library version 2 or higher
	lam	LAM MPI library
	mpibull2	Bull MPI library version 2
	mpich	MPICH MPI library version 1
	mpich2	MPICH MPI library version2
	openmpi	OpenMPI MPI library
	aixpoe	IBM POE MPI library for AIX
	ibmpoe	IBM POE MPI library for PPC Linux

	scali	Scali MPI library
	sgimpt	SGI MPT MPI library
	sun	SUN MPI library
Compiler	cce	Cray Compiler Environment
	gnu	GNU compiler collection
	ibm	IBM XL compiler
	intel	Intel compiler
	irix	SGI IRIX compiler
	path	Pathscale compiler
	pgi	PGI compiler
	sun	SunStudio compiler
Precision	32bit	
	64bit	

Table 1: UNITE Standard Names

1.4. UNITE Directory Structure

All files related to UNITE including the installed version of the packages are stored in one common directory specified by the environment variable `${UNITE_ROOT}`. The actual location of this directory can be chosen by the site installing UNITE, e.g., `/usr/local/unite` or `/opt/unite`. However, the directory and file structure inside `${UNITE_ROOT}` is specified by the UNITE standard. It has the following hierarchical structure:

Directory			Content
`${UNITE_ROOT}`	`/doc/`		UNITE documentation
	`/licenses/`		License files for commercial packages
	`/modulefiles`	`/tools/`	Module files for tool packages
		`/utils/`	Module files for utility packages
		`/devel/`	Module files for development packages
		`/templates/`	Module file templates
		`/scripts/`	Auxilary scripts used by module file templates
		`/UNITE/`	UNITE Module file(s)
	`/packages/`		Installation space for all packages

Figure 1: UNITE Directory Structure

The directory `${UNITE_ROOT}/packages/` contains all files related to the installation of a specific version of a package in the directory `<package>/<FQV>`, i.e., each version of a package is installed in a separate directory named after the full-qualified version (`<FQV>`) which is located in a directory named after the package (`<package>`). Example: The version 1.3.1 of the Scalasca toolset configured for SGI's MPT MPI library and Intel compilers will be installed in the directory `${UNITE_ROOT}/packages/scalasca/1.3.1-sgimpt-intel/`.

The same is true for the module files for the package installations. The module file for a specific package version is also named `<package>/<FQV>`, but depending on whether it is a tool, utility, or development package, it is located in the corresponding subdirectory of `${UNITE_ROOT}/modulefiles/`. Example: The corresponding module file of the example Scalasca installation above will be located at `${UNITE_ROOT}/modulefiles/tools/scalasca/1.3.1-sgimpt-intel/`.

2. UNITE Installation

This section describes how to install, update, and maintain an UNITE installation. The installation can be be done in an easy-to-use almost automatic way using the **UNITE Installer** which is strongly recommended. In some rare cases, e.g., if the installation platform is not yet supported by the UNITE Installer, the installation can still be done manually. Platforms currently not yet supported are especially systems which require cross-compilation or use separate frontend node(s) which are different from the actual compute (backend) nodes.

As of version 1.0, the UNITE installer and the associated packages were tested on Linux and AIX cluster systems using various compilers (gnu, ibm, intel, pgi) and MPI libraries (hp, intel2, mpibull2, mpich, mpich2, openmpi, aixpoe, sgimpt).

2.1. UNITE packages

The following table specifies where to download the UNITE software itself and the packages currently supported by UNITE. For packages which can be installed with the help of the UNITE installer, the oldest supported version is also specified.

Package	Download site	Oldest support version
UNITE Installer	http://apps.fz-juelich.de/unite/	—
UNITE Module Files	http://apps.fz-juelich.de/unite/	1.0
OTF	http://www.tu-dresden.de/zih/otf/	1.5.0

pdtoolkit	http://www.cs.uoregon.edu/research/pdt/	3.15
cube	http://www.scalasca.org/	3.3
Scalasca	http://www.scalasca.org/	1.3.0
VampirTrace	http://www.tu-dresden.de/zih/vampirtrace/	5.8
UniMCI	Included in ParMA package	1.0.1
Marmot	Included in ParMA package	2.4.0
Vampir (*)	http://www.vampir.eu/	5.0
VampirServer (*)	http://www.vampir.eu/	1.0

Table 2: UNITE Packages

For convenience, with the help of the EU ITEA2 Project ParMA, an one-in-all download package containing the UNITE module files and installer as well as packages of all open-source tools is provided. Look for the **ParMA Package** at http://apps.fz-juelich.de/unite/.

(*) The commercial tool packages for Vampir and VampirServer are also supported, but these always need to be downloaded separately. You can simply add the corresponding installation packages which you get after purchasing the Vampir software to the `packages` directory of the UNITE installer. In addition, you have to copy your individual license file(s) to the location reported by the UNITE installer configure script. For Linux systems, you can also download a time-limited demo versions of Vampir 5 from the Website www.vampir.eu/download and use this package in the installation. The necessary demo license file will be emailed to you after giving your registration data at the Vampir website. Demo versions for other platforms or for VampirServer are available on request. For this, please contact service@vampir.eu.

2.2. Automatic Installation with the UNITE Installer

2.2.1. Basic Procedure

The automatic installation with the UNITE installer can be done in five easy steps which are summarized below. In most cases, this will be enough to successfully install UNITE. In case of problems, for example due to platforms with an unusual setup, see the detailed installation description below (Section 2.2.2.).

UNITE Installer Quick Installation Guide

Step 1: Check requirements

The following additional software is recommended for a complete install of UNITE. In case of missing items, partial, but functional installations can still be done. The requirements are checked as part of the configuration (Step 3) and, in case of missing prerequisites, the resulting consequences are reported.

- A "module" environment
- QT Version 4.2 or later
- A Fortran compiler and MPI Fortran compilation support
- cmake

Step 2: Download

Download and unpack the ParMA Package in an arbitrarily location for building UNITE.

```
cd <BUILDDIR>
gunzip -c parma-package-1.0.tar.gz | tar xvf -
```

[OPTIONAL] Download the commercial packages for vampir and vampirserver suitable for your platform and place them into the directory `unite-installer-1.0/packages`.

Step 3: Configuration

Configure the package. Use the required option `--prefix=<INSTDIR>` to specify the UNITE installation directory (`${UNITE_ROOT}`, e.g. `/opt/unite`.

```
cd unite-installer-1.0
./configure --prefix=<INSTDIR>
```

This will try to autodetect the compilers and MPI library for building the package, check for an PAPI installation, check the availability of additional software required for building the package, and determine the list of packages to install. Carefully read the configure output. Configure will tell you which optional arguments allow you to guide the configuration process and might be needed in more complex build environments (e.g., when multiple compilers or MPI libraries are detected).

Configure creates a "workspace" for every build configuration named `BUILD-<PLATFORM>-<MPI>-<COMPILER>`

Step 4: Installation

Build and install all packages.

```
cd BUILD-<PLATFORM>-<MPI>-<COMPILER>
```

```
make
```

Step 5: UNITE module installation

Install the UNITE module in a system-wide module files directory.

```
cd <system-modulefile-directory>
ln -s <INSTDIR>/modulefiles/UNITE .
```

2.2.2. Detailed Installation Description

This section contains a more detailed step-for-step description of the installation for cases where the simple installation process discussed in Section 2.2.1. failed.

Step 1: Check requirements

The following additional software is recommended for a complete install of UNITE. In case of missing items, partial, but functional installations can still be done. The requirements are checked as part of the configuration (Step 3) and, in case of missing prerequisites, the resulting consequences are reported.

- **A "module" environment**

  Without a module environment, the UNITE packages will be still built and installed but the unified access to the tools via the UNITE package will not be possible. The necessary software for a module environment and installation instructions can be found at http://www.modules.org.

- **QT Version 4.2 or later**

  Without the QT graphics package, the CUBE graphical result browser (either of the standalone CUBE package or of the Scalasca toolset) will not be built. The CUBE library and the instrumentation and measurement components of Scalasca will still be built and installed. Qt is available at http://qt.nokia.com/.

- **A Fortran compiler and MPI Fortran compilation support**

  In this case, the Scalasca, VampirTrace, and Marmot tools will be build without measurement support for Fortran programs, i.e., only support for C/C++ programs will be available.

- **cmake**

  Without the cmake cross-platform makefile generator, the Marmot and UniMCI packages cannot be configured and built, and will ignored. Cmake can be downloaded from http://www.cmake.org/.

Step 2: Download

Download the desired package(s) from the corresponding website (see Table 2).

For a most complete UNITE installation, download and unpack the ParMA Package (`parma-package-1.0.tar.gz`) in an arbitrarily location for building UNITE.

```
cd <BUILDDIR>
gunzip -c parma-package-1.0.tar.gz | tar xvf -
```

In case only specific tools are to be installed, download and unpack the UNITE installer package (`unite-installer-1.0.tar.gz`) in an arbitrarily location for building UNITE.

```
cd <BUILDDIR>
gunzip -c unite-installer-1.0.tar.gz | tar xvf -
```

Then download (a) the UNITE module package (`unite-1.0.tar.gz`) and (b) the tool packages desired and place both into the directory `unite-installer-1.0/packages`.

In both cases, if desired, download the commercial packages for Vampir and VampirServer suitable for your platform (see also Section 2.1.) and place them into the directory `unite-installer-1.0/packages`. Configure (step 2) will tell you where to install the necessary license files.

Step 3: Configuration

On a platform which provides multiple compiler suites or MPI libraries, make sure the desired compiler and MPI compilation commands (e.g., `mpicc`) for building of specific version of the UNITE packages are in your path before other versions. On a system with a `module` configuration, this is typically accomplished by loading the corresponding modules. Consult your local platform documentation for details.

Then configure the UNITE package. Use the required option `--prefix=<INSTDIR>` to specify the UNITE installation directory (`${UNITE_ROOT}`, e.g. `/opt/unite`.

```
cd unite-installer-1.0
./configure --prefix=<INSTDIR>
```

This will try to autodetect the compilers and MPI library for building the package, check for an PAPI installation, check the availability of additional software required for building the package, and determine the list of packages to install. Carefully read the configure output. If necessary, and only then, configure will tell you which optional arguments allow you to guide the configuration process and might be needed in more complex build environments (e.g., when multiple compilers or MPI libraries are detected):

Compiler
: If the desired compiler suite is not detected by configure, make sure the right compiler invocation commands (e.g. cc) are in your `$(PATH)`. If configure detects more than one suitable compiler, it will print a message how you can use the `--compiler=<COMPILER>` to select the desired compiler suite out of the list of detected ones.

MPI Library
: If the desired MPI library is not detected by configure, make sure the right MPI compiler invocation commands, e.g. mpicc, are in your `$(PATH)`. If configure detects more than one suitable MPI library, it will print a message how you can use the `--mpi=<MPILIB>` option to select the desired MPI library out of the list of detected one.

PAPI
: If configure does not detect your PAPI installation, use the `--with-papi=<PAPIDIR>` option to specify the path where PAPI is installed or make sure the `papi_avail` command is in your `$(PATH)`.

QT
: If configure does not detect your QT installation, use the `--with-qmake=<QMAKE>` option to specify the absolute or relative path to the qmake command or make sure the qmake command is in your `$(PATH)`.

cmake
: If configure does not detect the cmake command on your platform, use the `--with-cmake=<CMAKE>` option to specify the absolute or relative path to the cmake command or make sure the cmake command is in your `$(PATH)`.

The configure command can be repeated as often as needed to setup the desired configuration.

Configure creates a "workspace" for every build configuration (i.e. for a specific compiler and MPI library on a specific platform) named `BUILD-<PLATFORM>-<MPI>-<COMPILER>`. This allows to reuse the build directory for multiple installations.

Step 4: Installation

Build and install all packages inside the workspace created by configure. It contains the necessary configuration setup (`config.cfg`), all logfiles (`LOGFILES/{configure,make,install}/<TOOL>.log`), and will be used to unpack and build the different UNITE tools.

```
cd BUILD-<PLATFORM>-<MPI>-<COMPILER>
make
```

In case of configuration, compiling, or installation errors, please consult the corresponding logfiles for more details. For simple errors, it is typically enough to rerun "`make`" after fixing the source of the error. In more complicated situations, especially if configuration changes are required, it is recommended to remove the workspace directory completely and start again with Step 3: Configuration.

If successful, the workspace directory and all its contents are no longer necessary and can be removed.

Step 5: UNITE module installation

Install the UNITE module in a suitable system-wide module files directory.

```
cd <system-modulefile-directory>
ln -s <INSTDIR>/modulefiles/UNITE .
```

Step 6: Install further versions

If you need the UNITE tools to support multiple compilers or MPI libraries, repeat steps 3) and 4) for every desired compiler/MPI library combination. Otherwise you are done.

2.3. Manual Installation and Setup

If the automatic installation with the UNITE installer fails, e.g., due to a unsupported platform like a system which requires cross-compilation, or, it is desired to make a set of already existing tool installations available under UNITE, UNITE still can be installed manually.

2.3.1. Download and Install UNITE Module Files

The minimal software required for a manual UNITE installation is the UNITE module file package `unite-1.0.tar.gz` (see Table 2).

Next, create the desired UNITE installation directory (`${UNITE_ROOT}`) and unpack the UNITE module file package inside it:

```
mkdir ${UNITE_ROOT}
cd ${UNITE_ROOT}
gunzip -c unite-1.0.tar.tz | tar xf -
chmod -R a+rX .
```

Finally, in the file `modulefiles/UNITE/1.0`, in the 3rd line, change the value of variable `pkg_root` to the UNITE installation directory, i.e., the contents of `${UNITE_ROOT}`.

2.3.2. Make Existing Packages available under UNITE

Existing versions of an already installed package <package> can be easily integrated into UNITE by creating an symbolic link from the corresponding UNITE package directory to the actual installation directory named after the fully-qualified version:

```
cd ${UNITE_ROOT}/packages/<package>/
ln -s <RealInstallationDirectory> <FQV>
```

Next, create the corresponding module file, again by creating an symbolic link, again named after the fully-qualified version, to a module file template provided by UNITE:

```
cd ${UNITE_ROOT}/modulefiles/<PackageType>/<package>/
ln -s  ../../templates/<PackageModuleTemplate> <FQV>
```

The module file templates are named after the standard names of the packages together with some indication for which group of versions they work, e.g., vampirtrace-5.X is for VampirTrace version 5 or later.

Repeat these two steps for every version of each already installed package which should be made available under UNITE.

2.3.3. Install Packages under UNITE

This section provides some basic information on how to install packages supported by UNITE for a UNITE environment. It is recommended to install the packages in the order they are listed in the table below. This allows to configure them so they are integrated to the maximum possible degree. The second column specifies the configuration command to be used and the option to specify the installation directory. In all cases the #_DIR need to be replaced by the name of the UNITE package installation directory name (${UNITE_ROOT}/packages/<package>/<FQV>). The third column describes additional (optional) configure options which allow to integrate the UNITE tools as much as possible. For more detailed installation instructions, please see the installation documentation of each package.

Package	Basic Configuration	Additional Configuration Options
UNITE module files	See Section 2.3.2.	—
cube	`configure --prefix=<CUBE_DIR>`	—
pdtoolkit	`configure -prefix=<PDT_DIR>`	—
OTF	`configure --prefix=<OTF_DIR>`	—

Scalasca	`configure --prefix=<SCA_DIR>`	`--with-papi=<PAPI_DIR>`
		`--with-pdt=<PDT_DIR>`
		`--with-otf=<OTF_DIR>`

Marmot	`cmake`
	`-DCMAKE_INSTALL_PREFIX=<MRMT_DIR>`
	`-DMPI_CC=<PathToMPICC>`
		`-DMARMOT_USE_CUBE=ON`
		`-DCUBE_CONFIG=<CUBE_DIR>/bin/cube-config`

UniMCI	`cmake` `-DCMAKE_INSTALL_PREFIX=<MRMT_DIR>`	`-DMARMOT_HOME=<MRMT_DIR>` `-DUSED_MPI_CHECKER=MARMOT`
Vampirtrace	`configure --prefix=<VT_DIR>`	`--with-papi-lib-dir=<PAPI_DIR>/lib` `--with-unimci-config=<MRMT_DIR>/bin/unimci-config`

Table 3: UNITE Packages Manual Installation Hints

A few, perhaps not immediately obvious remarks:

- In most cases, the packages require to specify the compiler suite (and sometimes the MPI library) to be used for building the package. Consult the package installation instructions for details.
- The configure command of pdtoolkit and cmake (used for Marmot and UniMCI) uses single dashes ("-") for options, all other double dashes ("--").
- UniMCI gets installed into the Marmot installation directory, not in its own separate directory.

After installation of the packages, the corresponding module files have to be created as an symbolic link named after the fully-qualified version to a module file template provided by UNITE:

```
cd ${UNITE_ROOT}/modulefiles/<PackageType>/<package>/
ln -s  ../../templates/<PackageModuleTemplate> <FQV>
```

The module file templates are named after the standard names of the packages together with some indication for which group of versions they work, e.g., vampirtrace-5.X is for VampirTrace version 5 or later.

2.4. Further (Manual) UNITE Installation Management

2.4.1. Set Default Package Version

If more than one version is installed of a package <package>, it is recommended to specify a default version for this package, i.e., the version

which is supposed to be loaded if no version information is specified when loading the package, e.g.,

```
module load <package>
```

The default version is typically the latest stable release of the package configured for the default compiler suite and MPI library of the platform. The default version is specified in a file `.version` located in the module directory of the corresponding package. Create it by make a copy of a template provided by UNITE:

```
cd ${UNITE_ROOT}/modulefiles/<PackageType>/<package>/
cp ../../templates/dot.version .version
```

Then use an editor to change "`X.X`" to the fully-qualified default version "`<FQV>`", for example "`1.3.1-sgimpt-intel`".

2.4.2. Remove a Package from UNITE

To remove (or disable) a specific version `<FQV>` of a package `<package>` from the UNITE environment, simply remove the corresponding module file:

```
cd ${UNITE_ROOT}/modulefiles/<PackageType>/<package>/
rm <FQV>
```

If you also want to remove the actual installation the package, remove the installation directory from the UNITE package directory:

```
cd ${UNITE_ROOT}/packages/<package>/
rm -rf <FQV>
```

2.4.3. Remove UNITE from Platform

To remove (or disable) the complete UNITE environment, remove the UNITE module file from the system-wide module files directory:

```
cd <system-modulefile-directory>
rm -rf UNITE
```

If you want to remove all UNITE related files completely from your system, delete the UNITE installation directory and all its contents.

```
rm -rf ${UNITE_ROOT}
```

2.4.4. Updates of Packages

The UNITE installer can also be used to install (additional) versions of the UNITE tools. If you used the ParMA package, simply replace all packages in the package subdirectory with the desired tool package file(s). Otherwise, download the UNITE installer (`unite-installer-1.0.tar.gz`). Then follow the same installation instructions as for a full install. Only make sure to specify the same (already existing) installation directory (`${UNITE_ROOT}`) for configure (Step 3).

3. Known Problems and Open Issues

Currently, the following problems are known:

- When installing the commercial Vampir or VampirServer packages, the UNITE installer does not check whether these binary packages match the architecture of the platform.