

CUBE 3.3 – Installation Guide

Generic Display for Application Performance Data

February 2010

The Scalasca Development Team
scalasca@fz-juelich.de

Copyright © 1998–2010 Forschungszentrum Jülich GmbH, Germany

Copyright © 2003–2008 University of Tennessee, Knoxville, USA

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the names of Forschungszentrum Jülich GmbH or the University of Tennessee, Knoxville, nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contents

1	CUBE 3.3 Installation Guide	1
1.1	Introduction	1
1.2	Availability & License	1
1.3	Prerequisites	2
1.4	Installation	3
1.5	Configuration	4
1.6	Support	4
	Bibliography	5
2	Appendix	7
2.1	Notes on Compiling and Installing Qt	7
2.2	Compiler- and Platform-specific Notes	8

1 CUBE 3.3 Installation Guide

1.1 Introduction

CUBE (CUBE Uniform Behavioral Encoding) is a presentation component suitable for displaying a wide variety of performance data for parallel programs including MPI [1] and OpenMP [2] applications. CUBE allows interactive exploration of the performance data in a scalable fashion. Scalability is achieved in two ways: hierarchical decomposition of individual dimensions and aggregation across different dimensions. All metrics are uniformly accommodated in the same display and thus provide the ability to easily compare the effects of different kinds of program behavior.

The CUBE package currently consists of five components:

- A simple CUBE writer library written in C.
- The full-featured C++ CUBE library for reading and writing CUBE files.
- A set of command-line tools to explore and modify CUBE files.
- A newly developed graphical user interface based on the Qt application framework.
- An older version of the graphical user interface based on the wxWidgets GUI library (deprecated).¹

Note that there is no need to compile and install both versions of the GUI. Only the Qt-based GUI will be compiled by default as this version is actively developed and provides much higher performance, scalability and a number of unique features. The wxWidgets-based version is only provided in case you have trouble getting the Qt-based version to work and needs to be explicitly enabled.

The remainder of this document describes the generic installation procedure for the different CUBE components (i.e., the C and C++ libraries, the command-line tools, as well as the graphical user interfaces).

1.2 Availability & License

CUBE is available as a source-code distribution for UNIX or UNIX-like platforms. It can be downloaded from <http://www.scalasca.org>. Besides the standalone distribution,

¹The wxWidgets-based GUI is considered deprecated and provided “as is”. It is no longer being developed and will be removed in future versions of the CUBE distribution.

CUBE is also distributed as part of the Scalasca performance analysis toolset available via the same web page.

The CUBE software is free, but by downloading, installing and using it you agree to comply with the license agreement. Please read the file `LICENSE` in the distribution's top-level directory for precise wording.

1.3 Prerequisites

Depending on the components of CUBE that are to be built, various tools and packages are required to be available on the build system. Common requirements—needed to build any of the components—are:

- GNU make
<http://www.gnu.org/software/make/>
- zlib
<http://www.zlib.net/>

Both are typically already installed on most systems. For zlib, you also need the development headers to be installed (might be provided by a separate package, e.g., on Linux systems).

For the graphical user interfaces, additional libraries/frameworks are required:

- Qt Framework (version 4.2 or higher)

<http://qt.nokia.com/>

The Qt framework is only needed to build the Qt-based graphical user interface.

- wxWidgets (wxGTK/v2.6 only)

<http://wxwidgets.org/>

The wxWidgets library is only needed to build the older, wxWidgets-based graphical user interface (deprecated). **Versions other than 2.6 or variants other than wxGTK (e.g., wxX11, wxMac or wxMSN) are known not to work!**

Note that wxGTK itself depends on the GTK+ toolkit which introduces additional dependencies. See the GTK+ homepage for details:

<http://www.gtk.org>

Although wxGTK can be configured to be used with either GTK+ v1.x or GTK+ v2.x, the GTK+ v1.x series has less dependencies and is easier to install.

1.4 Installation

This section describes the general procedure to build and install the CUBE software. Before proceeding with the instructions given below, **please make sure to also read the Qt installation notes in Appendix 2.1 as well as the platform- and compiler-specific notes in Appendix 2.2.**

To configure and install the CUBE package, the following steps are usually required:

1. Unpack the sources

```
gunzip -c cube-3.3.tar.gz | tar xvf -
```

2. Change into the CUBE source code directory

```
cd cube-3.3
```

3. Run the `configure` script

```
./configure
```

The `configure` script tries to determine whether the requirements for building CUBE are fulfilled and sets up the build configuration. Currently, the following options are recognized:

- **-prefix=*dir*** Specifies the installation directory (default: `/opt/cube3`)
- **-with-qmake=*path*** Specifies an alternative name for Qt's `qmake` (e.g., "`qmake-qt4`") or an absolute path to the `qmake` executable in case it is not in your search path (e.g., "`/opt/qt4/bin/qmake`").
- **-disable-gui** Disables building of any graphical user interface, i.e., only the CUBE libraries and tools will be built.
- **-enable-shared** Enables building of a shared CUBE C++ library (supported only for some compilers/platforms)
- **-enable-wx** Enables building of the older, deprecated `wxWidgets`-based GUI.
- **-help** Prints a short usage information.

The `configure` script assumes that the GNU C/C++ compilers should be used to compile CUBE (without checking their availability). If you want to use alternative compilers or compiler optimization options, you can do so by specifying appropriate environment variables before invoking `configure`, e.g.,

```
CC=xlc CFLAGS="-g -O2" ./configure --prefix=/opt/software/cube
```

Recognized variables are `CC` and `CFLAGS` for the C compiler and associated flags, `CXX` and `CXXFLAGS` for the C++ compiler and associated options, as well as `LDFLAGS` for flags passed to the compiler when linking.

4. Edit `Makefile.defs` (optional)

In the case that the `configure` script does not work as expected, or you want to manually customize the configuration, you have to edit the build configuration stored in the file `Makefile.defs` manually. The important variables are:

ECC: C compiler

ECFLAGS: C compiler options

ECXX: C++ compiler. Note that the compiler must be the same as the compiler used for compiling wxWidgets/Qt², unless you build the library only.

ECXXFLAGS: C++ compiler options

LDFLAGS: Linker options

ECXX_AR: Archive tool (e.g., ar or CC)

ECXX_ARFLAGS: Archive tool options

To disable building a particular version of the graphical user interface, comment out all lines with the corresponding “WX_” or “QT_” prefix near the end of the file.

5. Start the build process

```
make
```

6. Install the software

```
make install
```

1.5 Configuration

CUBE provides the option of displaying an online description for entries in the metric tree via a context menu. By default, it will search for the given HTML description file on all the mirror URLs specified in the CUBE file. For the case that no internet connection is available, the Qt-based CUBE GUI can be configured to also search in a list of local directories for documentation files. These additional search paths can be specified via the environment variable CUBE_DOCPATH as a colon-separated list of local directories, e.g.,

```
CUBE_DOCPATH=/opt/software/doc:/usr/local/share/doc
```

Note that this feature is only available in the Qt-based GUI and **not** in the older wxWidgets-based one.

1.6 Support

If you have any question or comments you would like to share with the CUBE developers, please send e-mail to scalasca@fz-juelich.de.

²The GNU and Intel compilers are interoperable to a certain degree. That is, you can use the Intel compilers to compile CUBE even if wxWidgets and/or Qt have been built with the GNU compilers, but typically not vice versa.

Bibliography

- [1] Message Passing Interface Forum: *MPI: A Message Passing Interface Standard — Version 2.2*, September, 2009, <http://www.mpi-forum.org> 1
- [2] OpenMP Architecture Review Board: *OpenMP Fortran Application Program Interface — Version 3.0*, May, 2008, <http://www.openmp.org> 1

2 Appendix

2.1 Notes on Compiling and Installing Qt

This appendix briefly describes how to customize the Qt installation process if it should only be used to build and run CUBE. In this case, some functionality of Qt can be disabled to speed up the build process.

In order to compile and use CUBE, only the Qt libraries are necessary. To disable compilation of other parts, the “`-nomake <part>`” option can be provided to the `configure` script. The following parts can be specified:

tools Although not strictly necessary for building and using CUBE, the “`qtconfig`” tool which provides a convenient way of defining default settings (e.g., window styles, font sizes, etc.) for all Qt-based applications will be built and installed as part of the `tools` part. Recent versions of Qt, however, take their settings from KDE (if available), i.e., this tool might not be needed.

examples This part will build and install a huge number of example programs referenced from Qt’s HTML documentation. These are not required by CUBE.

demos (>= Qt 4.3) Enabling this part will build and install a number of additional demo codes which aren’t required by CUBE as well.

docs (>= Qt 4.4) This part is responsible for creating and installing the HTML reference documentation of Qt. You don’t need to install it unless you plan to develop Qt applications yourself.

translations (>= Qt 4.4) If disabled, English will be the only supported language by various Qt tools.

In addition, Qt 3 backwards-compatibility support is not required by CUBE and could be disabled with the “`-no-qt3support`” switch. Furthermore, the Phonon and WebKit modules provided by Qt >= 4.4 are known to sometimes cause compilation problems. They can be disabled by specifying the “`-no-phonon`” or “`-no-webkit`” options, respectively.

2.2 Compiler- and Platform-specific Notes

This section contains some additional notes with respect to compilers and platforms where we encountered issues during our testing. Please also have a look at the generic platform and compiler notes of the Qt documentation:

<http://doc.qt.nokia.com/supported-platforms.html>

2.2.1 Cray XT

Building Qt and then CUBE with GNU compilers on Cray XT systems should be straightforward, however, it warrants paying attention to the compilers used and library dependencies they introduce. If, for example, the PrgEnv-gnu module is loaded when Qt is built, and PrgEnv-gnu depends on gcc/4.3, then it is likely to be required when CUBE is built. More critically, there is likely to be a dependence on gcc/4.3 when CUBE is executed which is at best inconvenient and likely to interfere with the gcc modules used by various PrgEnv modules. In particular, PrgEnv-pathscale typically expects a different gcc module.

It is therefore recommended to build Qt with the system-default `/usr/bin/g++` (typically v4.1.2) to avoid introducing dependences on gcc modules.

If Qt is already built and installed with dependence on a gcc module, and it is not desired to install a new version, then it is recommended to build CUBE explicitly specifying `/usr/bin/g++` (rather than picking up `g++` from a module on the path). You will need to modify the `ECC` and `ECXX` settings in `Makefile.defs` along with arguments to `qmake`:

```
ECC = /usr/bin/gcc
ECXX = /usr/bin/g++
QT_QMAKE = qmake QMAKE_CXX=$(ECXX) QMAKE_LINK=$(ECXX)
```

2.2.2 Intel Compiler

Various revisions of the Intel compilers are known to have problems compiling Qt. See

<http://doc.qt.nokia.com/compiler-notes.html>

for details. If you are using any of these revisions, we suggest to compile Qt using the GNU compilers instead. CUBE, however, can still be compiled and linked using the Intel compilers.

2.2.3 IBM AIX

Unfortunately, compiling Qt and CUBE on IBM AIX systems is not straightforward and a number of pitfalls need to be avoided. Please follow the instructions given below.

2.2.3.1 Compiling Qt

Officially, Qt only supports IBM's xl compilers up to version 6. Our experience shows that versions 7 to 9 are typically not able to compile Qt, and even if compilation succeeds, serious display issues can show up in Qt-based applications.

Our current recommendation is to use IBM's xl compiler version 10 in conjunction with a slightly modified version of Qt 4.5.1 (patch file provided, see below). If you don't have access to version 10 of the xl compilers, we recommend to use a recent version of the GNU compilers.

1. Unpack Qt sources

You need an up-to-date version of GNU tar; **AIX tar does not work** as it has problems with some very long filenames inside the Qt tar files. Otherwise compiling will not work because some files will be created with wrong (i.e., truncated) names.

Old versions of GNU tar also have a bug affecting unpacking Qt tar archives. Unfortunately, we do not know when this bug got fixed; GNU tar version 1.20 works for us, version 1.13 does not.

```
gtar xzf qt-x11-opensource-src-4.5.1.tar.gz
```

2. Patch Qt sources

To compile Qt with xLC version 10, the Qt source code needs to be patched, as xLC is more pedantic than other compilers. Note that this step is not necessary when using the GNU compilers, although it does not hurt either.

```
cd qt-x11-opensource-src-4.5.1
patch -p 1 < Qt-4.5.1_AIX-xlc10.patch
```

The patch file can be found in the `contrib` directory of the CUBE distribution package. We submitted this patch to Nokia as a bug report; hopefully future versions of Qt will have it integrated already.

3. Configure Qt

The following parameters need to be used in any case for the configure call:

```
./configure -prefix <install_path> -platform <pform>
```

When using the IBM xl compilers, `<pform>` is "aix-xlc" or "aix-xlc-64" to compile the 32 or 64-bit version of Qt, respectively. For the GNU compilers, use either "aix-g++" or "aix-g++-64".

See Appendix 2.1 for additional options that can be specified to save compilation time and disk space in case the Qt installation is only intended to be used for CUBE.

4. Compile and install Qt

Similar to CUBE, compiling and installing Qt requires GNU make.

```
gmake
gmake install
```

2.2.3.2 Compiling CUBE

Once Qt is set up, compiling CUBE is basically straightforward. You just need to make sure to pass the correct options to the configure script.

When using the GNU compilers, configuration should work out of the box, unless Qt has been compiled with a non-default precision. In this case, you need to overwrite the default compiler options:

```
CFLAGS="-O2 -m<prec>" CXXFLAGS="-O2 -m<prec>" ./configure
```

where `<prec>` is either 32 or 64.

For the xl compilers, you also need to provide the compiler name:

```
CC=xlc CFLAGS="-O2 -q<prec>" \
CXX=xlc CXXFLAGS="-O2 -q<prec>" \
./configure
```

where `<prec>` is again either 32 or 64. For the 32-bit version, you should also provide

```
LDFLAGS="-bmaxdata:0x80000000"
```

to increase the amount of memory available to the application.

2.2.4 Mac OS X

When using Mac OS X 10.6 ("Snow Leopard"), you need at least Qt version 4.6. Older versions of Qt are known not to work as expected.