JÜLICH
FORSCHUNGSZENTRUM

CUBE 4.2.2 – Installation Guide

Generic Display for Application Performance Data

February 25, 2014

The Scalasca Development Team
scalasca@fz-juelich.de

Copyright

Contents

1 CUBE 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 and OpenMP 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.
- A simple CUBE reader library written in Java.
- 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 graphical user interface based on the Qt application framework.

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, 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 (CUBE can be build without zlib support if zlib is not available)

  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 interface an additional library/framework is required:

- Qt Framework (version 4.3 or higher)

  http://qt.nokia.com/

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'Notes on Compiling and Installing Qt'as well as the platform- and compiler-specific notes in Appendix'Compiler- and Platform-specific Notes'.**

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

1. Unpack the sources

   ```
   gunzip -c cube-4.x.x.tar.gz | tar xvf -
   ```

2. Change into the CUBE source code directory

   ```
   cd cube-4.x.x
   ```

3. Create a `VPATH` directory, where do you build the CUBE, and change into it

   ```
   mkdir -p vpath; cd vpath
   ```

4. 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. The configure script provides a set of various options. The full list is displayed when invoking:

```
../configure --help=recursive
```

A list of main options:

--prefix=*dir* Specifies the installation directory (default: `/opt/cube`)

--with-nocross-compiler-suite=*(gcc|ibm|intel|pathscale|pgi|studio|open64|clang)* The compiler suite to build this package in non cross-compiling environments with. Needs to be in `$PATH` [gcc].

--with-frontend-compiler-suite=*(gcc|ibm|intel|pathscale|pgi|studio|open64|clang)* The compiler suite to build the frontend parts of this package in cross-compiling environments with. Needs to be in $PATH [gcc].

--with-cubelib | **--without-cubelib** Enables (default) or disables building and installation of the cube c++ library.

--with-tools | **--without-tools** Enables (default) or disables building and installation of cube tools.

--with-gui | **--without-gui** Enables (default) or disables building and installation of the Cube GUI.

--with-cwriter | **--without-cwriter** Enables (default) or disables building and installation of the C cube writer.

--with-java-reader | **--without-java-reader** Enables (default) or disables the compilation, build and installation of the cube java reader

--with-xerces-path=*path* Specifies the path to the xerces.jar (default: `/usr/share/java`)

--with-xerces-name= *Name.jar* Specifies the name of the jar file with the xerces xml parser (default: `xerces.jar`)

--with-frontend-zlib=*"path to frontend zlib"* Defines the zlib library, used by cube on frontend

--with-backend-zlib=*"path to backend zlib"* Defines the zlib library, used by cube on backend

--with-compression=full|ro|none Enables or disables zlib compression support in cube:

- **full** All parts of the framework create compressed cubes and read compressed and uncompressed cube files.
- **ro** Read only configuration. Created cube files are uncompressed, but tools and GUI can open and operate with compressed cube files. This is default value.
- **none** Zlib library is not used in this case and compressed cube files are not supported.

--with-qt=*path* Forces to look for Qt in the `path` in first place. If this option is omitted the configure script searches for a Qt installation using the following sequence:

a) `$QT_DIR`

b) `$PATH`

c) `/usr/local/Trolltech/`

d) `/opt/local/libexec/qt4-mac/` (to support Mac)

e) `/opt/lib/qt4`

f) `/usr/lib/qt/bin`

g) `/usr/lib32/qt4`

h) `/usr/lib32/qt`

i) `/usr/lib64/qt4`

j) `/usr/lib64/qt`

k) `/opt/qt`

l) `/opt/qt4`

m) `/usr/local/qt4`

n) `/usr/local/qt`

--with-qt-specs=*spec* Defines a specs for Qt (Default is "default")

--with-vampir | **--without-vampir** Enables (default) or disables support of TraceBrowser connection with Vampir.

The support of TraceBrowser is only possible if DBUS headers and DBUS library are available. The configure script tries to find them automatically. If it fails, you have to specify the exact place of two headers: `dbus/dbus.h` and `dbus/dbus-arch-deps.h`. You can do it like

```
../configure --prefix=/opt/software/cube \
    CPPFLAGS='-I/usr/include/dbus-1.0  \
      -I/usr/lib64/dbus-1.0/include'"
```

--with-paraver | **--without-paraver** Enables support of TraceBrowser connection with Paraver.

See the note above.

--with-paraver-cfg=FILE Uses FILE as a configuration file for Paraver.

--disable-shared Disables building of a shared CUBE C++ library. (Build with shater libraries is supported only for some compilers/platforms)

The `configure` script assumes that the GNU C/C++ compilers should be used to compile CUBE. You can select a different compiler using one of the options above e.g.

```
../configure - -with-nocross-compiler-suite=intel
```

If you want to use alternative compilers or compiler optimization options, which are not listed in the description of the options, you can do so by specifying appropriate variables in the command line when invoking `configure`, e.g.,

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

For a list of the recognized variables, please inspect the output of the command `./configure - -help=recursive`

5. Start the build process

   ```
   make
   ```

   or

   ```
   make -j N
   ```

6. You can invoke a test suite of the CUBE framework

   ```
   make check
   ```

   If all tests pass, everything looks good. If some tests do not pass, please report it to scalasca@fz-juelich.de and attach the file `configure.log`

7. 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. In case there is no Internet connection, 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.

To prevent CUBE from trying to load the HTML documentation via HTTP or HTTPS mirror URLs (e.g., in restricted environments were outbound connections are blocked by a firewall and the timeout is taking very long), the environment variable `CUBE_DISABLE_HTTP_DOCS` can be set to either `1`, `yes` or `true`.

1.6 Support

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

2 Appendix

2.1 Notes on Compiling and Installing Qt

This appendix briefly describes how to customize the Qt installation process if it is only 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 can 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 dependency 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 dependencies on gcc modules.

If Qt is already built and installed with a dependency 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 was fixed; GNU tar version 1.20 works for us, version 1.13 does not.

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

2. **Patch Qt sources**

   To compile Qt with XL C version 10, the Qt source code needs to be patched, as XL C 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 already integrated.

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'Notes on Compiling and Installing Qt'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.

The configure script performs a set of checks trying to find out the parameters, how to compile the CUBE GUI. If it fails, you 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:

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

where <prec> is either 32 or 64.

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

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

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.

Some parts of CUBE code use operator dynamic_cast<...>.

According to http://www-01.ibm.com/support/docview.wss?uid=swg21007500 , the dynamic_cast operator will always return NULL if the compiler isn't instructed to generate the necessary dynamic type information.

Till this issue is not resolved by CUBE build system automatically, please configure CUBE on POWER/AIX with

```
CXXFLAGS="-qrtti=dynamiccast"
```

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.

3 Bibliography

Message Passing Interface Forum: *MPI: A Message Passing Interface Standard — Version 2.2*, September, 2009, http://www.mpi-forum.org

OpenMP Architecture Review Board: *OpenMP Fortran Application Program Interface — Version 3.0*, May, 2008, http://www.openmp.org