jCubeR 4.8 | Library Usage

Intoduction in Cube tool development guide

November 2022
The Scalasca Development Team
scalasca@fz-juelich.de

Attention

The Cube Tool Developer Guide is currently being rewritten and still incomplete. However, it should already contain enough information to get you started and avoid the most common pitfalls.

Contents

1 Makefile for provided examples

1.1 Quick info about makefile.

Here we provide a small example of a makefile, which is used to compile and build examples delivered with CUBE. Similar makefiles can be used by developers to compile and build own jCubeR tools.

1.2 Commented source

First we specify the installation path of CUBE and its "jcuber-config" script. This script delivers correct flags for compiling and linking, paths to the CUBE tools and GUI. (besides of another useful technical information)

```
CUBE_DIR = /path/CubeInstall
CUBE_CONFIG = $(CUBE_DIR)/bin/jcuber-config
```

Additionally we specify CLASSPATH and SYSTEM_CLASSPATH to compile and link examples.

```
CLASSPATH =  $(shell $(CUBE_CONFIG) --classpath)
SYSTEM_CLASSPATH = $(shell echo $$CLASSPATH)
```

Here a compiler is selected to compile and build the example.

```
JAVAC = javac
JAVA= java
```

We define explicit suffixes for an executable file, created from C source, from c++ source and an MPI executable. If one develops a tool, which is using MPI, it is useful (sometimes) to define a special suffix for automatic compilation.

```
.SUFFIXES: .java .java.class
.PHONY: all  clean
```

Object files of examples and their targets

```
# Object files
OBJS =

TARGET =jcuber_example.java.class
```

Automatic rule for the compilation of every single Java source into .o file and for building targets.

```
%.java.class : %.java
         $(JAVAC) -d . -cp "$(SYSTEM_CLASSPATH):.:$(CLASSPATH)" $<
```

Automatic rule for the compilation of every single java source into .class file and for building targets.

```
#-------------------------------------------------------------------------------
# Rules
#-------------------------------------------------------------------------------

all: $(TARGET)
    $(JAVA) -classpath "$(SYSTEM_CLASSPATH):.:$(CLASSPATH)" $$japp example.cube; \
        $(JAVA) -classpath "$(SYSTEM_CLASSPATH):.:$(CLASSPATH)" $$japp example.cube dump; \
```

2 Examples of using Cube Reader Java library

Present example shows in several short steps the main idea of using the jCubeR library and obtaining different values from this cube file.

2.1 Commented source

Import necessary modules

```
....
import scalasca.cubex.cube.*;
import scalasca.cubex.cube.errors.*;
import java.util.*;
import java.lang.*;
import java.lang.String;
import java.io.*;
```

Import own jCubeR modules.

```
import scalasca.cubex.cube.services.transformation.*;
import scalasca.cubex.cube.datalayout.data.value.*;
```

Start as usual with a public static main call

```
public class jcuber_example
{
    public static void main(String[] args)
    {
```

Create an instance of Cube object,.

```
Cube cube = new Cube();
try
{
```

Open an existing .cubex file. If file is not found an exception is thrown. Hence try-catch.

```
cube.openCubeReport(args[0]);
```

With various *get_* calls obtain information about structure of the .cubex file.

```
    ArrayList<Metric> metrics = cube.get_metv();
    ArrayList<Metric> root_metrics = cube.get_root_metv();
    ArrayList<Region> regions = cube.get_regionv();
    ArrayList<Cnode> cnodes = cube.get_cnodev();
    ArrayList<Cnode> root_cnodes = cube.get_root_cnodev();
    ArrayList<SystemTreeNode> machines = cube.get_root_stnv();
ArrayList<SystemTreeNode> stns = cube.get_stnv();
    ArrayList<Node> nodes = cube.get_nodev();
    ArrayList<scalasca.cubex.cube.LocationGroup> lgs = cube.get_location_groupv();
ArrayList<scalasca.cubex.cube.Location> locs = cube.get_locationv();
    ArrayList<scalasca.cubex.cube.Cartesian> topologies = cube.get_cartv();
```

For example you can print out data for every element.

```
System.out.println(
        "Version:" + cube.get_version() + "\n" +
                "Metrics:" + metrics.size() + "\n" +
                "Root Metrics:" + root_metrics.size() + "\n" +
                "Regions:" + regions.size() + "\n" +
                "Cnodes:" + cnodes.size() + "\n" +
                "SystemTreeNodes:" + stns.size() + "\n" +
                "Machines:" + machines.size() + "\n" +
                "LocationGroups:" + lgs.size() + "\n" +
                "Locationa:" + locs.size() + "\n" +
                "Topologies:" + topologies.size()
                  );
System.out.println("------- Metrics  in " + args[0] + " ------------------");
for (Metric met : root_metrics)
{
    printMetrics(met, 0);
}
```

To obtain the data for a specific metric, call path and location use the call `get_sev_adv`

```
for (scalasca.cubex.cube.Metric met: metrics)
{
    System.out.println(" ================================ Metric "+met.getDisplayName() + "
       ============================");
    for (scalasca.cubex.cube.Cnode cnode: cnodes)
    {
    System.out.println(" -------------------------------- Cnode "+cnode.getName() + "
       ----------------------------");
    for (scalasca.cubex.cube.Location location: locs)
    {
        System.out.print( cube.get_sev_adv(met, cnode, location).toString() + " ");
    }System.out.println();
    }
}
```

Various exceptions can be thrown and should be cought and processed properly

```
        }catch (BadSyntaxException e)
        {
        ...
                System.out.println("General error:" + e.getMessage());
        }
}
```

Example of usage of the jCubeR library one can find in `[prefix]/share/doc/jcuber/examples`

cube

scalasca

JÜLICH
FORSCHUNGSZENTRUM

www.scalasca.org