version 1.1

a debugger for task-parallel programming

Steffen Brinkmann et al.
HLRS, Universität Stuttgart

Disclaimer: The information contained in this manual is not guaranteed to be complete at this stage. It is subject to changes without further notice. Please send comments, corrections and additional text to temanejo@hlrs.de.

Steffen Brinkmann et al.: TEMANEJO – a debugger for task-parallel programming.

Contents

1 Introduction

TEMANEJO [1] started as a graphical debugger for the task-parallel, data-dependency-driven programming model SMPSs. The foremost goal was to display the task-dependency graph of SMPSs applications, and to allow simple interaction with the SMPSs runtime system in order to control some aspects of the parallel execution of an application.

Since then, we aim at supporting *any* programming model that can meaningfully define the concepts of task and dependencies between them. Notably, this includes OpenMP and MPI, i.e. the working horses in high-performance computing.

Today, TEMANEJO is able to assist debugging (to varying extent) for the programming models SMPSs [2], OmpSs [3], StarPU [4, 5], OpenMP [6], CppSs [7] and ParRSEC[8].

TEMANEJO actually is only the graphical frontend. Most of the real work is done by a library called AYUDAME which is used to receive information, so called events, from supporting runtime systems, and to excert control over a runtime system by issuing requests to it.

Directory layout of the distribution tarball

TEMANEJO is distributed as a tarball `temanejo-1.1.tar.gz`. Unpacking that will result in the following directory structure:

```
temanejo-x.y/ ----|
                  |-- doc/
                  |-- Temanejo/
                  |-- Ayudame/
                  |-- modulefile/
```

Usually, you need only descend into the directory `doc/`, for instance to read this manual. The directories `Temanejo/` and `Ayudame/` contain the sources for TEMANEJO, the graphical debugger, and AYUDAME,the library talking to the runtime systems, respectively.

How to read this manual

TEMANEJO is designed to be widely self-explanatory. Therefore you might want to read the *Quick start guide* (section 2) and start debugging your application right away.

If you have trouble installing packages on which TEMANEJO depends on, section 3 about *Installing* TEMANEJO*'s requirements* is the starting point for you.

For more in depth information of how to get most out of TEMANEJO, read *Hands on* (section 4).

2 Quick start guides

2.1 General requirements

TEMANEJO is a Python application. You will need the following modules

- ▷ Python v2.7 or higher, not tested with Python v3
- ▷ PyGTK (http://www.pygtk.org)
- ▷ NetworkX (http://networkx.lanl.gov/)

2.2 Installing Temanejo and Ayudame

1. Untar the TEMANEJO distribution tarball and change into the directory created by it. This directory will be referred to as TEMANEJO source directory, `$TEMANEJO_SRC`.

   shell:
   ```
   # tar xvzf temanejo-1.1.tar.gz
   # cd temanejo-1.1; export TEMANEJO_SRC=$(pwd)
   ```

2. Choose your installation location, e.g. `/opt/temanejo/` and save this directory in the environment variable `$TEMANEJO_HOME`.

   shell:
   ```
   # export TEMANEJO_HOME=/opt/temanejo/or/other/path
   ```

3. Run the script `./configure`, build the source and install as

   shell: $TEMANEJO_SRC directory
   ```
   # ./configure --prefix=$TEMANEJO_HOME
   # make && make install
   ```

   In order to install without debug symbols, type

   shell: $TEMANEJO_SRC directory
   ```
   # make install-strip
   ```

   - ▷ In case the `./configure` complains about a missing Python installation, make sure it is available in the search path. On many HPC systems you might need to load a Python module

     shell: $TEMANEJO_SRC directory
     ```
     #   module load python
     ```

   - ▷ In case the `./configure` scripts returns other errors, try to regenerate it with

shell: $TEMANEJO_SRC directory

```
# autoreconf -fiv
```

4. After installation, you might want to add `$TEMANEJO_HOME/bin` to the search path `$PATH`, and `$TEMANEJO_HOME/lib` to the library search path `$LD_LIBRARY_PATH`. On bash, e.g.:

shell

```
# export PATH=$TEMANEJO_HOME/bin:$PATH
# export LD_LIBRARY_PATH=$TEMANEJO_HOME/lib:$LD_LIBRARY_PATH
```

▷ If necessary, you can disable the installation of the Temanejo GUI or the Ayudame library, respectivey, with the configure options:

shell: $TEMANEJO_SRC directory

```
--disable-temanejo
--disable-ayudame
```

In particular, building the Ayudame library on some older Mac OS versions is not supported and you might have to disable it.

▷ You can also ask the build system to install a modulefile from a user-provided template with

shell: $TEMANEJO_SRC directory

```
--enable-modulefile[=template_file]
```

If no template file is provided, it will use the one in `$TEMANEJO_SRC/modulefile/example_modulefile.in`.

▷ See the output of `./configure --help` for further ways to control the building and installation process.

2.3 SMPSs

Temanejo is supported by SMPSs v2.6 and above. However, at installation time, SMPSs needs to be configured to activate Temanejo support. To get started as quickly as possible follow these few steps:

1. Make sure your SMPSs installation is instrumented with calls to Ayudame. Simply check for occurance of the Ayudame event-handler `AYU_event` by typing

shell

```
# strings $SMPSS_HOME/lib/libSMPSS.so | grep AYU
```

where `$SMPSS_HOME` is the installation directory of SMPSs. The output should show a couple occurrences of the string `AYU` including `AYU_event`.

If no occurence of the string `AYU` is found, you will have to rebuild SMPSs with support for Ayudame. Go to the SMPSs source directory, referred to as `$SMPSS_SRC`, and type (if `configure` is already present in the directory you can skip the first commad):

shell: $SMPSS_SRC directory

```
# autoreconf -fiv
# ./configure ... --with-ayudame-include=$TEMAMEJO_HOME/include ...
# make && make install
```

`$TEMANEJO_HOME` is the absolute path to the Temanejo installation directory. Make sure to add other configure options as appropiate. Set the environment variables `$PATH` and `$LD_LIBRARY_PATH` according to the SMPSs documentation.

2. Now you are ready to compile your SMPSs program as usual

shell: application directory

```
# smpss-cc smpss_app.c -o smpss_app
```

3. Start Temanejo and pass your application

shell: application directory

```
# Temanejo ./smpss_app
```

and then press the "connect"-button. You might have to add the path to the Ayudame library, i.e. `$TEMANEJO_HOME/lib/libayudame.so`. Press "connect" at the bottom. At this stage the Temanejo window on the left will show a list of potential tasks in your application. In order to start stepping through your application you will want to press "Run" (the triangular shaped "Play"-button in the toolbar).

2.4 OmpSs

Temanejo is supported by the OmpSs runtime Nanos v0.7 and above. Partial support is available in snapshots of the v0.7a series (in particular this version of Temanejo is known to work with Nanos snapshots from April 2014). Temanejo and OmpSs/Nanos talk to each other through an Ayudame plugin for OmpSs/Nanos. Both, Temanejo and OmpSs/Nanos are shipped with a copy of this plugin and there are thus two different ways to enable OmpSs support in Temanejo. Currently, we recommend to use only one of the two plugins at a time. Specifically, if OmpSs/Nanos has already been installed with Ayudame support, use it and skip ahead to compiling OmpSs applications at list item 3 below.

If you have not yet installed OmpSs, we recommend to ignore the plugin shipped with OmpSs/Nanos and instead use the plugin shipped with Temanejo.

To get started as quickly as possible follow these few steps:

1. Test whether your OmpSs/Nanos installation is instrumented with calls to Ayudame. Simply check by typing

```shell
# nanox --version | tr '[:blank:] ' '\n' | grep "with-ayudame"
```

   Look for a message of the form

```shell
  --with-ayudame=/some/directory/here
```

   - ▷ If this is present, OmpSs/Nanos has been build with Ayudame support. You can skip ahead to compiling OmpSs applications, see list item 3. (If you experience problems when debugging OmpSs applications, it might be worth to replace the instrumentation as explained in the following.)
   - ▷ If the string `with-ayudame` does not appear, then your Nanos installation is not build with support for Ayudame and you should use the plugin shipped with Temanejo.

2. If OmpSs/Nanos has been installed without Ayudame support, install Temanejo's own OmpSs plugin with

```shell: $TEMANEJO_SRC directory
# autoreconf -fiv
# ./configure ... --with-ompss=$OMPSS_HOME \
# make && make install
```

   `$OMPSS_HOME` is the absolute path to the OmpSs/Nanos installation directory. Make sure to add other configure options as appropiate.

3. Now you are ready to compile your OmpSs program requesting instrumentation

```shell: application directory
# mcc --ompss --instrumentation -o ompss_app ompss_app.c
```

4. Before starting Temanejo with your OmpSs application you have to set

```shell:
# export LD_LIBRARY_PATH=$TEMANEJO_HOME/lib:$LD_LIBRARY_PATH
```

5. Start Temanejo and pass your application

```shell: application directory
# Temanejo ./ompss_app
```

   and then press the "connect"-button. You might have to add the path to the Ayudame library, i.e. `$TEMANEJO_HOME/lib/libayudame.so`. Press "connect" at the bottom. Unlike with SMPSs, Temanejo's left window will not show the list of

your application's tasks initially, as these are created at runtime, only. Nonetheless, in order to start stepping through your application you will want to press "Run" (the triangular shaped "Play"-button in the toolbar).

2.5 StarPU

TEMANEJO is supported by StarPU version v1.1.0 and above. Partial support is available in the late v1.0 series. However, at installation time, StarPU needs to be configured to activate TEMANEJO support. Do the following steps:

1. Make sure your StarPU installation is instrumented with calls to Ayudame. Simply check by typing

   shell:
   ```
   # strings $STARPU_HOME/lib/libstarpu*.so | grep AYU
   ```

   where `$STARPU_HOME` is the installation directory of StarPU. The output should show a couple occurrences of the string `AYU_event`.

   If no occurence of the string `AYU` is found, you will have to rebuild StarPU with support for AYUDAME. Go to the StarPU source directory, referred to as `$STARPU_SRC`, and type (if `configure` is already present in the directory you can skip the first command):

   shell: $STARPU_SRC directory
   ```
   # autoreconf -fiv
   # ./configure CPPFLAGS=-I$TEMAMEJO_HOME/include ...
   # make && make install
   ```

   `$TEMANEJO_HOME` is the absolute path to the TEMANEJO installation directory. Make sure to add other configure options as appropiate. Set the environment variables `$PATH` and `$LD_LIBRARY_PATH` according to the StarPU documentation.

2. Now you are ready to compile your StarPU program as usual.

   To debug a StarPU application with Temanejo just pass its name to it

   shell: application directory
   ```
   # Temanejo ./starpu_app
   ```

   Then press the "connect"-button. You might have to add the path to the AYUDAME library, i.e. `$TEMANEJO_HOME/lib/libayudame.so`. Press "connect" at the bottom. Unlike with SMPSs, TEMANEJO's left window will not show the list of your application's tasks initially, as these are created at runtime, only. Nonetheless, in order to start stepping through your application you will want to press "Run" (the triangular shaped "Play"-button in the toolbar).

3 Installing Temanejo's requirements

Temanejo requires Python version >= 2.7 and some additional packages which are freely downloadable for most common systems. In case your system is not mentioned below or you want to get newer versions than the ones that come with your system, please visit the respective websites:

Python http://www.python.org/ - recommended version 2.7.5

PyGTK http://www.pygtk.org/ - recommended version 2.24.0

PyGObject http://live.gnome.org/PyGObject - recommended version 3.0.0

PyCairo http://cairographics.org/pycairo/ - recommended version 1.8.8

NetworkX http://networkx.lanl.gov/ - recommended version 1.5

Please note that these software packages may depend on other software such as C libraries etc. In the following you find the package names for different Linux[1] distributions and a short "Howto" for Mac OS[2].

3.1 Linux

Temanejo is easy to install on Linux systems. Usually some extra packages need to be installed. You find the names for these packages for some distributions below.

3.1.1 Debian/Ubuntu

Install the following packages with

shell: as superuser

```
# apt-get install <package>
```

- **python**
- **python-gtk2**
- **python-gobject**
- **python-cairo**
- **python-networkx**

[1] Linux is a trademark owned by Linus Torvalds
[2] Mac OS is a trademark of Apple Inc.

3.1.2 Gentoo

Install the following packages with

shell: as superuser

```
# emerge <package>
```

- **dev-lang/python**
- **dev-python/pygtk**
- **dev-python/pygobject**
- **dev-python/pycairo**
- **dev-python/networkx**

3.1.3 arch

Install the following packages with

shell: as superuser

```
# pacman -S <package>
```

- **python2**
- **pygtk**
- **python2-gobject2**
- **python2-cairo**
- **python2-networkx**

3.1.4 openSUSE (untested)

Install the following packages with

shell: as superuser

```
# zypper install <package>
```

- **python**
- **python-gtk**
- **python2-gobject2**
- **python2-cairo**
- **python2-networkx**

3.1.5 Fedora (untested)

Install the following packages with

shell: as superuser

```
# yum install <package>
```

- ▷ **python**
- ▷ **pygtk2**
- ▷ **pygobject2**
- ▷ **pycairo**
- ▷ **python-networkx**

3.2 Mac OS

Building the AYUDAME library on Mac OS requires a fairly recent C++ compiler as for instance gcc v4.8 available through MacPorts. You will also need python v2.7.5 and some python modules which may not be part of standard Mac OS.

Most package managers, as MacPorts and Fink, will provide packages for these requirements. The rest of this guide assumes you are using MacPorts (http://www.macports.org). The same basic procedure applies for other package managers or source installation.

To install all required packages safely, you will make sure your MacPorts is up-to-date, install Python v2.7.5 (this will take a while) and install the requirements for TEMANEJO:

shell: as superuser

```
# port selfupdate
# port install python27
# port install py27-pygtk py27-networkx
```

4 Hands on – the users guide

4.1 Starting Temanejo

There are two ways to start your Temanejo session. The difference lies in how you connect Temanejo to the application you want to debug. The first way is to let Temanejo start your application on the same machine that Temanejo is running on. Typically you will use this way for running the application and debugging it on your laptop or desktop machine. We will refer to this debugging mode as *"local"*.

The other way is to start your application and Temanejo independently and specify the hostname and port via which Temanejo will connect to the application. In this case, the application can run on a remote computer (or computing node) as long as it is accessible from your local system via the network. This type of connection will be referred to as *"remote"*. Please note that the remote machine may as well be *"localhost"*.

4.1.1 Local session

For a local debugging session simply invoke Temanejo from the command line:

shell

```
# Temanejo
```

Please note that Temanejo is a Python v2 application. That means that on systems with Python v3 as the default Python installation you might need to call the correct Python version, e.g.

shell

```
# python2 $(which Temanejo)
```

On many HPC systems you might need to load a Python module

shell: $TEMANEJO_SRC directory

```
#   module load python
```

You may as well place a link to Temanejo in your launch bar or on your desktop and start it with a simple click.

Temanejo will open the connect dialogue automatically. In this dialogue you must specify the application you want to run. Temanejo also needs to know the location of the Ayudame library. When the installation process didn't fail, this library is found at the default place and you do not have to change this parameter. Default places where Temanejo looks for Ayudame are:

Figure 1: Temanejo icon

1. the environment variable `AYUDAME_LIB`. It should hold
   the path including the filename, which usually is `libayudame.so`
2. /path/to/Temanejo/executable/../lib/libayudame.so
3. /path/to/Temanejo/installation/lib/libayudame.so

All other settings are optional. The optional settings are: the command line options for your application, the configuration file for the SMPSs runtime environment, the port and the number of threads you want to run your application on when using SMPSs.

For convenience, you can specify the application which you want to debug in the command line, appending it to the call to Temanejo:

shell
```
# Temanejo my_app
```

and/or give the host and port to which to connect:

shell
```
# Temanejo -c the_remote_computer:5977
```

There are many command line options to Temanejo which will find in the help string:

shell
```
# Temanejo --help
Usage: Temanejo [options] [app [app_args]]
Temanejo connects to a task-parallel application which has been instrumented
with calls to the Ayudame library. Supported runtimes include SMPSs, OmpSs,
StarPU, and CppSs. In the simplest case, Temanejo will start the parallel
application "app" and pass it its arguments "app_args". Temanejo will
interpret the first argument it does not support as the application name. It
is also possible to set up remote connections. Please refer to the README file
and the documentation in the doc/ directory.
Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -v, --verbose         Switch on verbose output, which prints warnings. Use
                        -vv (info) and -vvv (debug) for increasingly epic
                        information.
  -q, --quiet           Switch off any output except critical error messages.
                        This overwrites -v, -vv and -vvv.
  -c HOST:PORT, --connect=HOST:PORT
                        Specify parameters for remote connection of form
                        "host:port". Using this option will cause Temanejo to
                        omit the initial connect dialogue. The option argument
                        is required. One of the arguments (host or port) can
                        be omitted. So if you want to connect to localhost at
                        port 111, type "-c :111". If you want to connect to
                        localhost at port $AYU_PORT type "-c :" or use the
                        "-C" option.
```

```
  -C, --auto-connect    This option will cause Temanejo to omit the initial
                        connect dialogue and connect directly to localhost at
                        port $AYU_PORT.
  -l LIB_FILE, --library=LIB_FILE
                        Specify library file to communicate with the
                        application. By default, libayudame.so is expected in
                        temanejo_dir/lib/.
  -t TITLE, --title=TITLE
                        Window title (appended to "Temanejo:").
  -d LAYOUTDIR, --layout-direction=LAYOUTDIR
                        How to draw the graph: top to bottom ('t2b'), left to
                        right ('l2r'), Bottom to top ('b2t') or right to left
                        ('r2l') [default: t2b].
  -Z, --autozoom        Switch on autozooming of the graph.
  -z, --no-autozoom     Switch off autozooming of the graph [default].
  -n NODE_COL, --node-colour=NODE_COL
                        Choose what the node colour means. One of "nothing"
                        (no colours drawn), "status" (colour indicates task
                        status), "thread" (thread Id), "function" [default]
                        (function Id), "task_dur" (task duration).
  -m MARGIN_COL, --margin-colour=MARGIN_COL
                        Choose what the margin colour means. One of "nothing"
                        (no margins drawn), "status" [default] (margin
                        indicates task status), "thread" (thread Id),
                        "function" (function Id), "task_dur" (task duration).
  -e EDGE_COL, --edge-colour=EDGE_COL
                        Choose what the edge colour means. One of "nothing"
                        (black edges), "orig_addr" [default] (original address
                        of dependency), "addr" (address of dependency after
                        renaming).
  -s SHAPES, --shapes=SHAPES
                        Choose what the node shape means. One of "nothing" (no
                        margins drawn), "thread" [default] (thread Id),
                        "function" (function Id).
  --pygraphviz          Specify whether to use pygraphviz for the graph
                        layout. Pygraphviz is part of the networkx package,
                        but may cause problems on some systems. Barriers are
                        not going to be displayed.
  --global-tasks-in-first-row
                        Specify whether global tasks should be drawn in first
                        line.

Temanejo is published under the terms of the BSD license. (C) HLRS, University
of Stuttgart. For further information please refer to the LICENSE file.
```

4.1.2 Remote session

In order to start a remote session, you have to start your application with some environment variables set. Choose an unused port and set `AYU_PORT`:

shell
```
# export AYU_PORT=5977
```

Usually port numbers higher than 1024 are safe. To be sure type:

shell
```
# netstat -an | grep -i listen
```

and use a port which is not in the list. If you do not specify `AYU_PORT`, the AYUDAME library will choose one and report it on the terminal as soon as your application is started. Next, start your application preloading the AYUDAME library:

shell
```
# LD_PRELOAD=/path/to/libayudame.so ./my_app
```

The application will run until it reaches the AYUDAME's initialisation routine at which point it will pause and wait for command from TEMANEJO. This means that for some of the supported programming models, code user code may have been already been executed before control is transferred to TEMANEJO, for others not. As soon as the AYUDAME initialisation routine is reached, the port number (i.e. `AYU_PORT`) for the connection with TEMANEJO is displayed in the console. Now, launch TEMANEJO.

shell
```
# Temanejo
```

In the connection dialogue, choose the "remote" tab and set the remote host and the AYUDAME port to the corresponding values. If you exported `AYU_PORT` before launching TEMANEJO, the specified port number will automatically be set.

Now press "Connect" and debug your program.

The connection dialog can be bypassed (partly or fully) by specifying the remote host and the AYUDAME port on the command line as

shell
```
# Temanejo -c [remote_host]:[ayu_port]
```

If omitted, remote host and AYUDAME port default to `localhost` and `$AYU_PORT`, respectively.

4.2 The graphical user interface

The graphical user interface (GUI) is divided into two main sections: the graph display to the right and the information display to the left. Additionally there is a toolbar (by default horizontally at the top, or vertically between the panels if set in the preferences menu) and a status bar at the bottom of the window (see figure 2).

In the following we will describe the full functionality the program in detail.

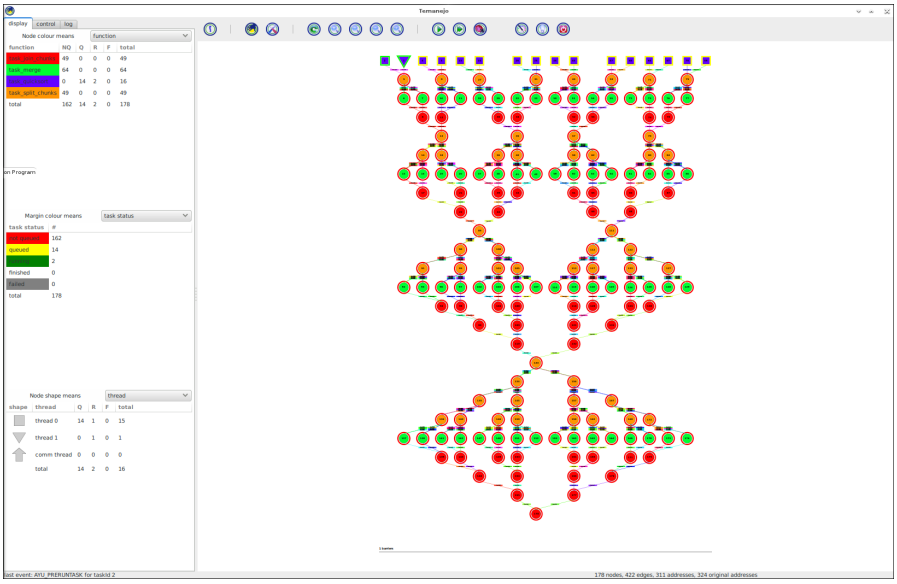

Figure 2: TEMANEJO screenshot

4.2.1 The graph display

The graph display, or *graph widget*, makes up most of the program surface. When starting TEMANEJO, it will consist of a white area, because TEMANEJO will pause execution of the application right after the initialisation of the runtime. As soon as you advance the application execution by at least one step (see section 4.2.3), the runtime will start adding tasks to the graph. Consecutively the display will be filled with the dependency graph as it is constructed by the runtime.

Looking at the whole graph: To change the view on the dependency graph you can use the following commands:

Action	Keyboard	Mouse
Zoom In	PageUp *or* +	Scroll wheel up
Zoom Out	PageDown *or* -	Scroll wheel down
Standard Zoom Level	z	
Zoom to fit the whole graph	Esc	
Magnify area		Hold middle button and mark rectangle
Move graph	*arrow buttons*	Hold left button and mark rectangle

Looking at a single task: With a right-click on a node (task), a context menu opens giving you more information about the task and offering some actions (see figure 3). The first two lines indicate the unique identification number of the task and the identification

of the thread on which the task is or was executed or for which the task is scheduled. '-1' (as seen in figure 3) means that the task has not been assigned to an execution thread yet.

The next line denotes the name of the function or a generated name of the form `function_N` where `N` is a unique number. A function name is generated when the runtime environment did not send information about the actual function name.

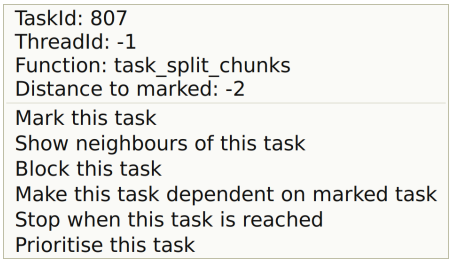

Figure 3: TEMANEJO context menu

If a task is marked (see below), an additional line above the separator is displayed indicating the distance to the marked task. Distance means number of dependencies (edges) to or from the marked task. Note, that a dependency graph is a directed acyclic graph, hence edges can only be traversed in one direction, If more than one connections are possible between two nodes, the shortest distance is displayed.

For tasks that do not depend on each other, i.e. could run in parallel potentially, the distance is denoted as `None`. A negative number means, that the marked task is going to be executed before the clicked task, i.e. the latter depends on the marked task.

The items below the separator are clickable and allow you to further analyse the task dependencies and to control the execution of the application. Which of and how these options actually work depends on whether the programming model's runtime supports these features.

The first two clickable options of the context menu allow you to mark a task and to mark the task and highlight its direct neighbours, respectively. When you right-click on another task after having marked a task, the distance to that task is displayed in the context menu as described above.

The option 'Block this task' will mark the task with a thick red cross and send the request to block this task to the runtime. The intended behaviour of the StarSs runtime is to execute every task (and all code outside of tasks) which does not depend on the blocked task but not the blocked task itself.

'Make this task dependent on marked task' is only clickable if another task is marked. If activated it will block the task until the marked task is finished.

Blocking and subsequently adding dependencies is only possible if supported by the programming model's runtime system.

The option 'Stop when this task is reached' can be used to advance the execution of the application until a certain task is about to be executed. The execution will then pause. Usually you will uncheck the 'Stop...' options in the control tab in the information and control display to the left (see section 4.2.2) after you marked a task to pause at. Then you will advance the execution using the step function (see section 4.2.3). When the marked task is reached, you will switch on one or more of the 'Stop...' options and step through the graph as usual.

Finally, 'Prioritise this task' will toggle the priority level of a task. Again, this is

available only if supported by the programming model's runtime system.

4.2.2 The information and control display

The information and control display consists of several panels organised in tabs.

Display tab : In the default tab, named "display", the meaning of the node colour, margin colour and node shape are denoted and can be changed via the drop down menu at the top of each list.

In the default settings, the node colour means "function", i.e. the colour of each node refers to the function which will be executed. In the table, the function names of the registered functions are displayed with the respective node colour as background. The counters indicate how many of each function calls (tasks) are "not queued" (`NQ`), "queued" (`Q`), "running" (`R`) and "finished" (`F`). Last column and row show the totals for each function and status, respectively.

Changing the meaning of the colours and shapes is done by choosing a different value from the drop down menus. The columns and rows will change respectively and are self-explanatory.

Control tab : In this tab you find a number of ways to control the execution and display of the graph. Topmost you can navigate directly to a task of a specific number. With the next button you can clear any markers set via the context menu in the graph view.

In the section "Breakpoints" you can specify, when the execution of the program should be paused. A click on the "Run" button (see section 4.2.3) will run the program until of the events is reached, which you checked here.

Also you can run `gdb` with a breakpoint set to a specific function.

Log tab : In the log tab the log of the incoming messages is shown.

4.2.3 The toolbar

The symbols on the toolbar will change depending on your window manager, button style and operating system. From left to right, the buttons perform the following tasks:

- ▷ **Connect**: Open the Connect dialogue.
- ▷ **Preferences**: Open the preferences dialogue.
- ▷ **Redraw**: Redraw the graph.
- ▷ **Zoom in**: Increase zoom level.
- ▷ **Zoom out**: Decrease zoom level.
- ▷ **Fit graph to window**: Adjust zoom level, so that the whole graph fits into the window. Please be aware that for large graphs this may result in a blank screen, when there are more nodes to be drawn than your screen has pixels.

- ▷ **Default zoom level**: Set zoom level to 1.
- ▷ **Run until next pause**: Use this to step through the graph.
- ▷ **Run until nth pause**: This is used for fast stepping through the graph. `n` can be adjusted in the preferences dialogue.
- ▷ **Debug**: start `gdb`.
- ▷ **Export image or animation**: Export an image or animation of the current graph.
- ▷ **About Temanejo** : Some information about Temanejo.
- ▷ **Quit**: Quit Temanejo.

4.2.4 The status bar

The status bar provides you with some additional information about the running application and the task graph. On the left the last registered incoming message is displayed. On the right, some graph characteristics are listed. In detail that is the number of nodes, the number of nodes (i.e. tasks), edges (i.e. dependencies), dependency addresses and dependency addresses before renaming. Renaming is a feature of the SMPSs runtime. For further information please refer to the SMPSs documentation. Additional warnings may be displayed in the center of the status bar.

4.3 How to invoke the gnu debugger

Temanejo can use the classical debugger `gdb` to attach to running applications and do line-based debugging of task internals. On Ubuntu Linux and derivatives thereof, attaching to running processes is not allowed for normal users.[3] In order to allow it, you need to edit the file `/etc/sysctl.d/10-ptrace.conf` as root. There is only one parameter in the file which has to be set to 0. This is only necessary once and requires a restart of the computer to take action.

To use `gdb` with Temanejo, you have to compile your application with the gcc option `-g`. For SMPSs it is also necessary to apply the option `-k`:

shell: application directory

```
# smpss-cc -g -k smpss_app.c -o smpss_app
```

These options are needed to compile with debugging symbols and to keep the temporary files that SMPSs generates. Actually it is these files that you will be debugging.

Start the application and Temanejo as usual and press the debug button or 'd' on your keyboard. A window with the gdb session opens. Type in your breakpoints (for example "b func_name") or other `gdb` commands. Then type 'c' (or 'continue'). `gdb` returns the

[3] See https://wiki.ubuntu.com/SecurityTeam/Roadmap/KernelHardening#ptrace%20Protection

control to TEMANEJO. Now you can step through the task graph with TEMANEJO, but keep an eye on the `gdb` window. When a breakpoint is reached, TEMANEJO will stop reacting. Nevertheless, when you hit the "run" button 20 times, it will apply these 20 steps later. So don't.

Step through lines of code with gdb (command: 's' or 'step' or whatever you find in `gdb`'s manual). When you're done with `gdb`, type 'c' in the `gdb` window again, and the control goes back to TEMANEJO.

References

[1] S. Brinkmann, Ch. Niethammer, J. Gracia, and R. Keller. Temanejo - a debugger for task based parallel programming models. In *ParCo2011: Proceeding of the International Conference on Parallel Computing*, 2011.

[2] TEXT - Towards EXaflop applicaTions (http://www.project-text.eu).

[3] The OmpSs programming model (http://pm.bsc.es/ompss).

[4] The StarPU open source project on Ohloh (https://www.ohloh.net/p/starpu).

[5] Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. *StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures"*. 2009.

[6] OpenMP (http://openmp.org/wp/).

[7] S. Brinkmann and J. Gracia. Cppss – A C++ Library for Efficient Task Parallelism. In *INFOCOMP 2013 Proceedings*. IARIA XPS Press, 2013.

[8] PaRSEC – Parallel Runtime Scheduling and Execution Controller (https://icl.cs.utk.edu/parsec/).