LTTng

Stable: 30.03.2021 - 10:17 / Revision: 10.02.2021 - 10:38

Contents

1 Article purpose

This article provides the basic information needed to start using the Linux® tracing tool: **LTTng**[1] (Linux Trace Toolkit, next generation).

2 Introduction

The following table provides a brief description of the tool, as well as its availability depending on the software packages:

✓: this tool is either present (ready to use or to be activated), or can be integrated and activated on the software package.

✗: this tool is not present and cannot be integrated, or it is present but cannot be activated on the software package.

Tool			STM32MPU Embedded Software distribution			STM32MPU Embedded Software distribution for Android™		
Name	Category	Purpose	Starter Package	Developer Package	Distribution Package	Starter Package	Developer Package	Distribution Package
LTTng	Tracing tools	**LTTng**[1] is an open source tracing framework for Linux kernel and user spaces. It is a powerful tool that can be used for many purposes. LTTng traces need to be processed /displayed with a	✗	✗	✓	✗	✗	✗

Tool			STM32MPU Embedded Software distribution			STM32MPU Embedded Software distribution for Android™		
Name	Category	Purpose	Starter Package	Developer Package	Distribution Package	Starter Package	Developer Package	Distribution Package
		host tool such as Trace Compass[2], based on Eclipse plugin for increased portability.						

This tool is composed of 3 parts:

- lttng-ust: user space libs
- lttng-tools: user space tools
- lttng-modules: kernel modules

3 Installing the trace and debug tool on your target board

3.1 Using STM32MPU Embedded Software distribution

OpenSTLinux embeds lttng-ust and lttng-tools by default.

The LTTng Kernel space is not installed/activated by default in OpenSTLinux to avoid increasing the roofs size.

To use LTTng in OpenSTLinux you need to both:

- activate 'Kernel Function Tracer'. the Linux kernel configuration must activate CONFIG_FUNCTION_TRACER and CONFIG_FUNCTION_GRAPH_TRACER using the Linux kernel Menuconfig tool (Menuconfig or how to configure kernel):

```
Symbol: FUNCTION_TRACER
Location:
  Kernel Hacking --->
    Tracers -->
      [*] Kernel Function Tracer

Symbol: FUNCTION_GRAPH_TRACER
Location:
  Kernel Hacking --->
    Tracers -->
      [*] Kernel Function Tracer
       [*] Kernel Function Graph Tracer
```

This trace framework is not activated by default in OpenSTLinux distributions since there is an impact on the Linux kernel size (around 1.5% increase of vmlinux), and also an impact on the overall performance, because of additional processing done to trace kernel events and function calls.

- compile and install lttng-modules

lttng is integrated in weston image distribution through openembedded-core package: *openembedded-core/meta/recipes-core/packagegroups/packagegroup-core-tools-profile.bb*.

```
RDEPENDS_${PN} = "\
    ${PROFILETOOLS} \
    ${LTTNGUST} \
    ${LTTNGTOOLS} \
    ${LTTNGMODULES} \
    ${BABELTRACE} \
    ${SYSTEMTAP} \
    ${VALGRIND} \
    "
```

however, lttng-modules is removed in meta-st package: *meta-st/meta-st-openstlinux/recipes-core/packagegroups/packagegroup-core-tools-profile.bbappend*.

```
RDEPENDS_${PN}_remove = "${LTTNGMODULES}"
```

3.1.1 Developer Package

It is not recommended to enable Linux kernel configuration for LTTng by using Developer Package because all external modules should also be recompiled (*e.g. gcnano driver for GPU STM32MP1*), and this is not possible with Developer Package (which does not necessary provide all sources).

That is the reason why this is set as 'not supported' for Developer Package.

3.1.2 Distribution Package

If you have already activated the requested configuration for ftrace, then you can go directly to "Compile lttng-modules" item.

- Activate requested Linux kernel configuration: please refer to Ftrace#Distribution_Package.
- Compile lttng-modules

  Comment line meta-st package: *meta-st/meta-st-openstlinux/recipes-core/packagegroups/packagegroup-core-tools-profile.bbappend*.

```
-RDEPENDS_${PN}_remove = "${LTTNGMODULES}"
+#RDEPENDS_${PN}_remove = "${LTTNGMODULES}"
```

- Re-build the image in order to integrate new Linux kernel uImage and LTTng modules.

```
PC $> bitbake st-image-weston
```

- Flashload partition images. See Flashing the built image.

4 Getting started

The best way to *get started* with LTTng is to have a look to the official webpage of LTTng[1].

4.1 Activate and generate traces on target side

Some useful information can be found in LTTng documentation[3].

Kernel tracing example[4]:

```
Create a tracing session:
Board $> lttng create my-kernel-session

List the available kernel tracepoints and system calls:
Board $> lttng list --kernel

Create an event rule that matches the desired event names, for example sched_switch and
sched_process_fork:
Board $> lttng enable-event --kernel sched_switch,sched_process_fork

You can also create an event rule that matches all the Linux kernel tracepoints (this
will generate a lot of data when tracing):
Board $> lttng enable-event --kernel --all

Start tracing:
Board $> lttng start

Generate activity on your system for a few seconds. For example, start an application,
load a kernel module, or list the files within a directory...

Stop tracing and destroy the tracing session:
Board $> lttng stop
Board $> lttng destroy
The destroy command does not destroy the trace data; it only destroys the state of the
tracing session.
```

4.2 Import LTTng traces in a Trace Compass trace viewer

4.2.1 Host Installation for Trace Compass

Even if LTTng are tools running on target, the analysis of the traces generated by LTTng tools should be done with host tools like **Trace Compass**[2].

So first you have to download and install/run the Trace Compass tool[5]: download the last version (or at least TraceCompass-3.2.0).

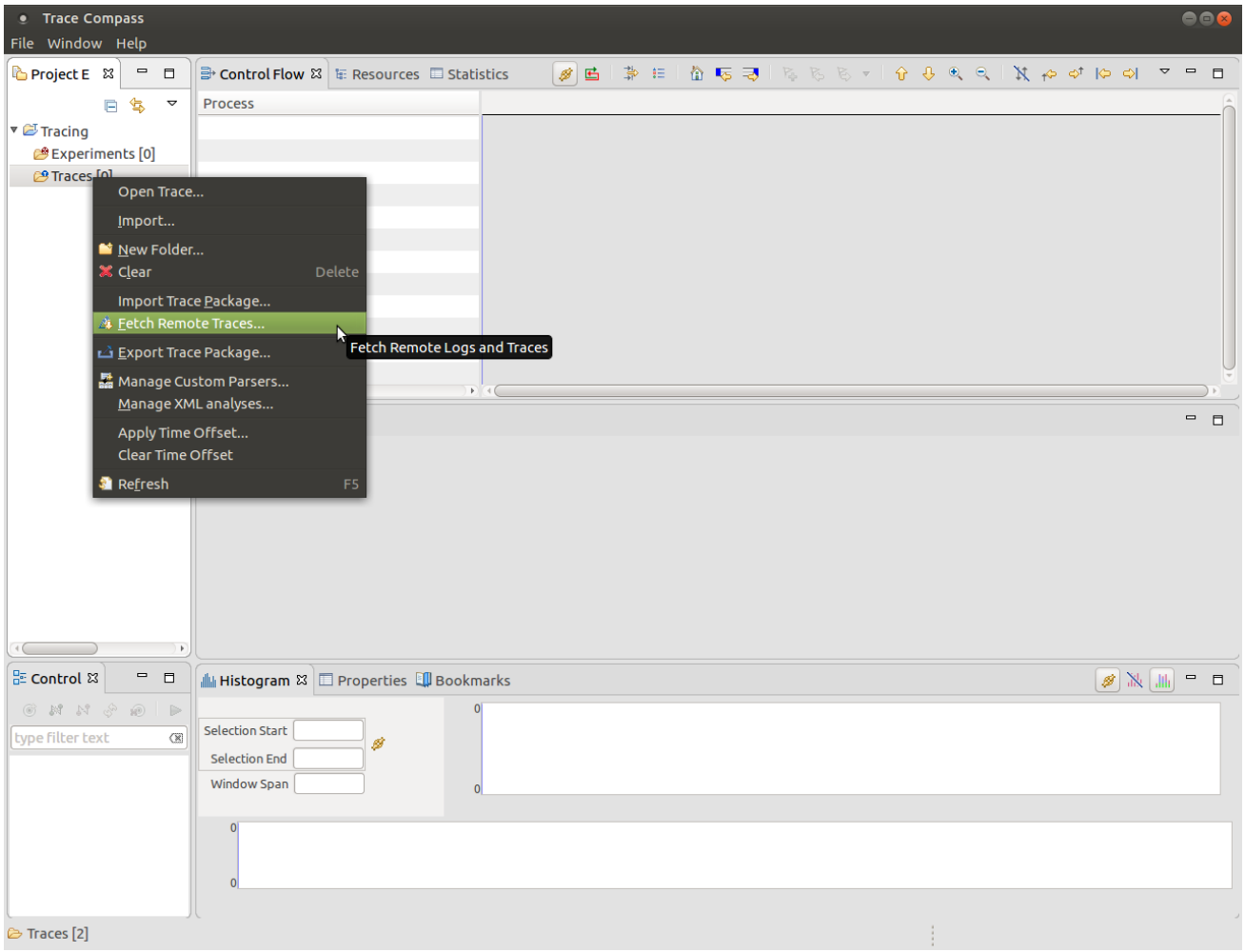
4.2.2 Visualize LTTng trace with Trace Compass

Once the trace have been generated you can use Trace Compass[2] to visualize your traces, interpreted in dedicated context view.

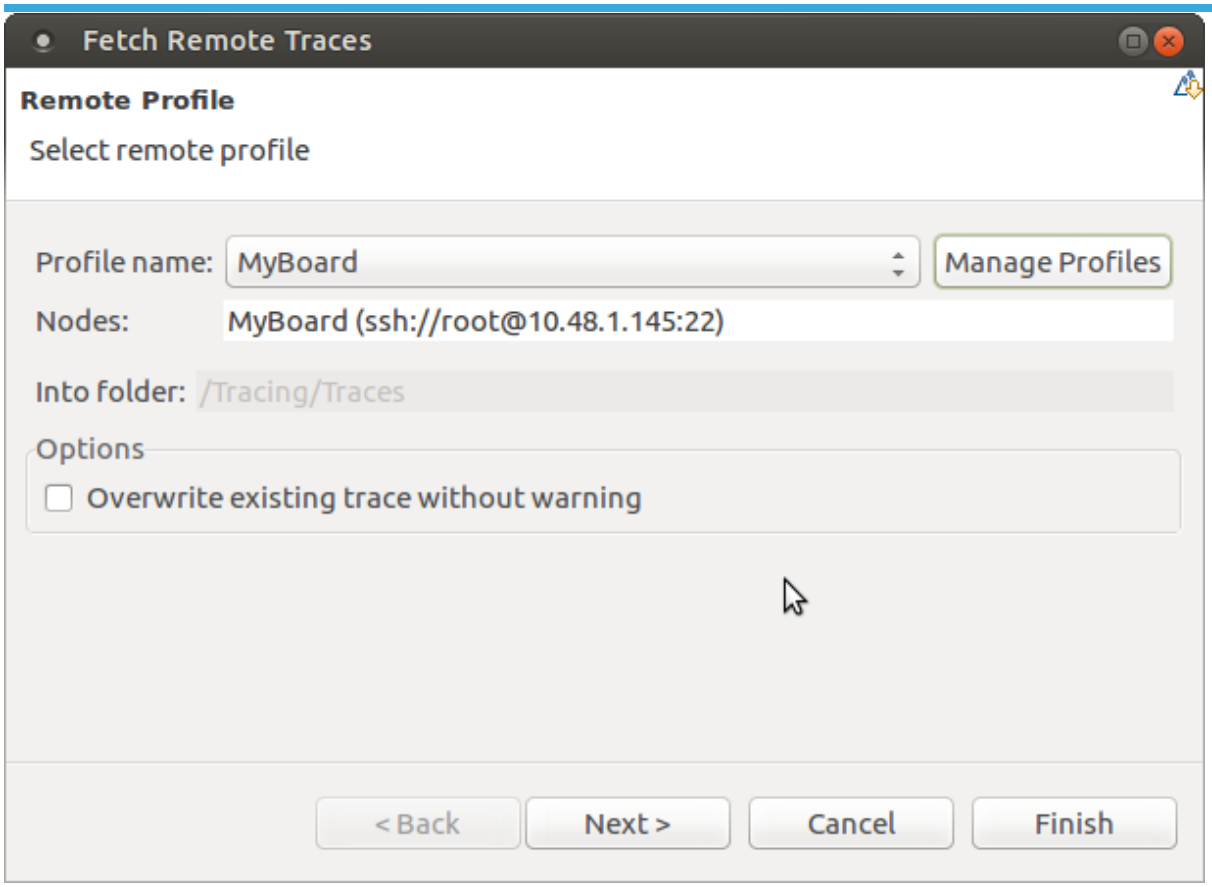
- First ensure you can open an ssh session with your board. **But close it after, because no ssh session must be open for the next steps**

```
PC $> ssh root@<ip_of_remote_board>

Board $>
Board $> exit
logout
Connection to <ip_of_remote_board> closed.

PC $>
```

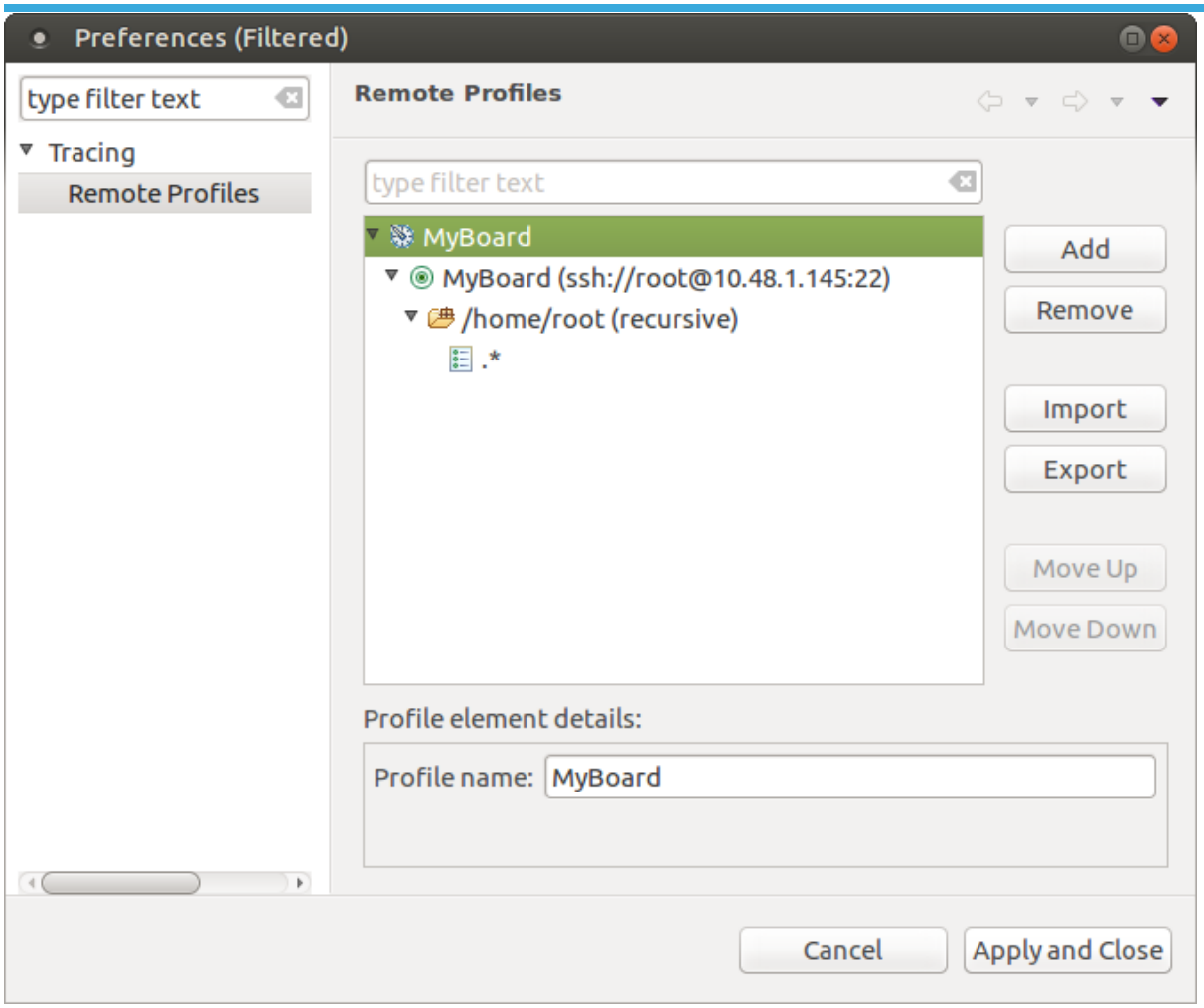
- Open Trace Compass and fetch the remote trace:



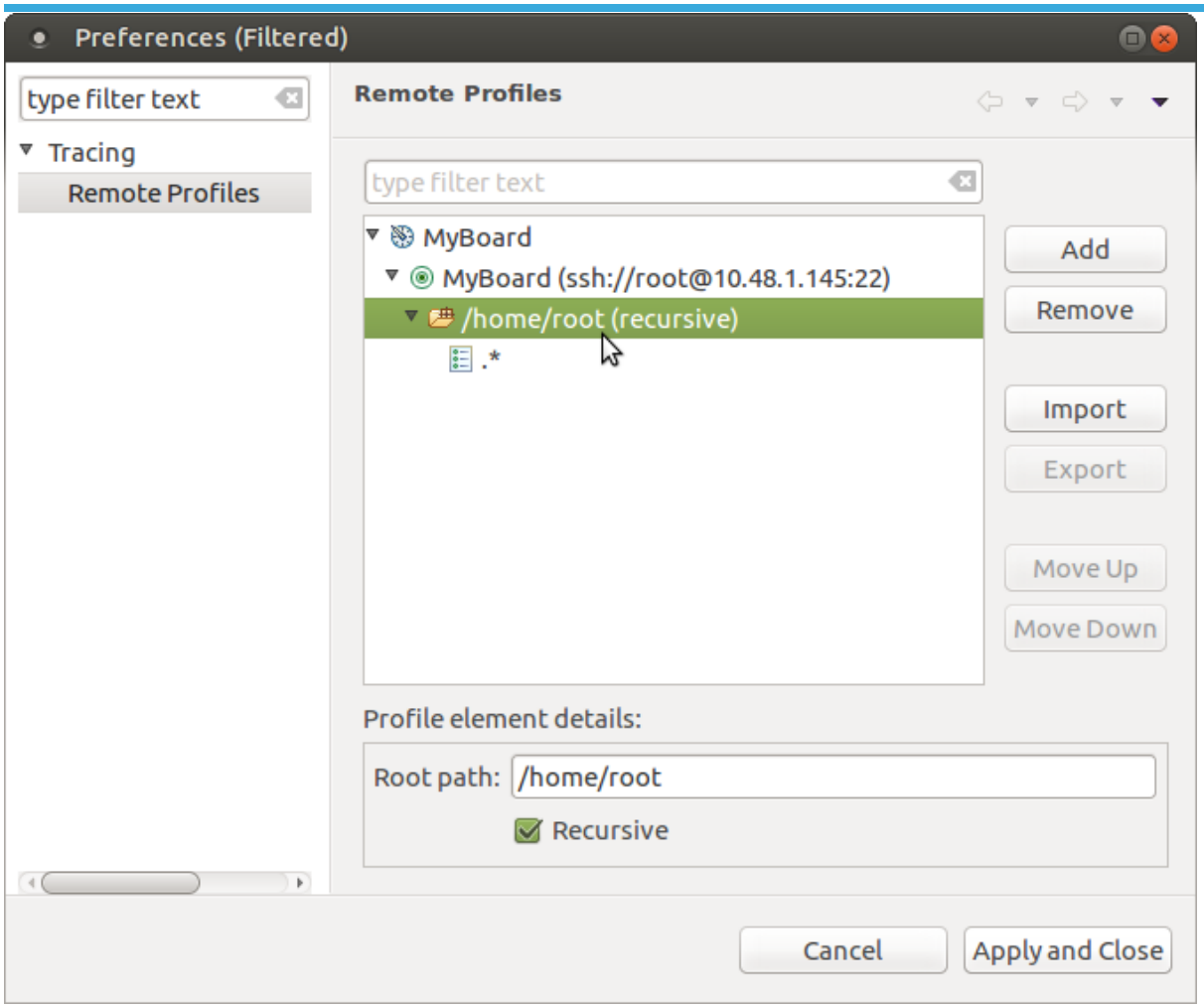
- Click on Manage profile to select/add the profile matching your setup (STBoards connection through ssh, ...)



- If needed add a new profile: in this case, a profile named STBoards with ssh connection

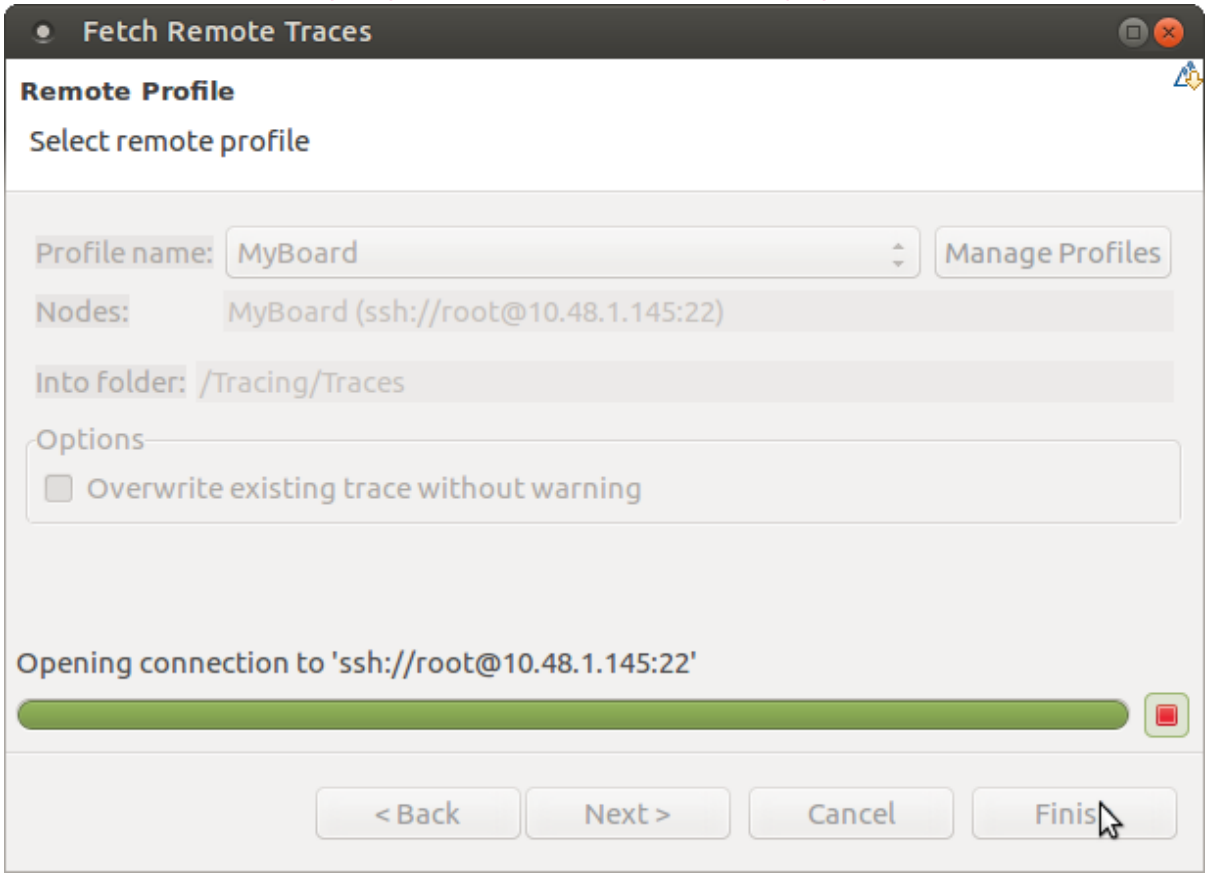


- Take care about the search path, by default Trace Compass looks for a wrong search path, please correct it



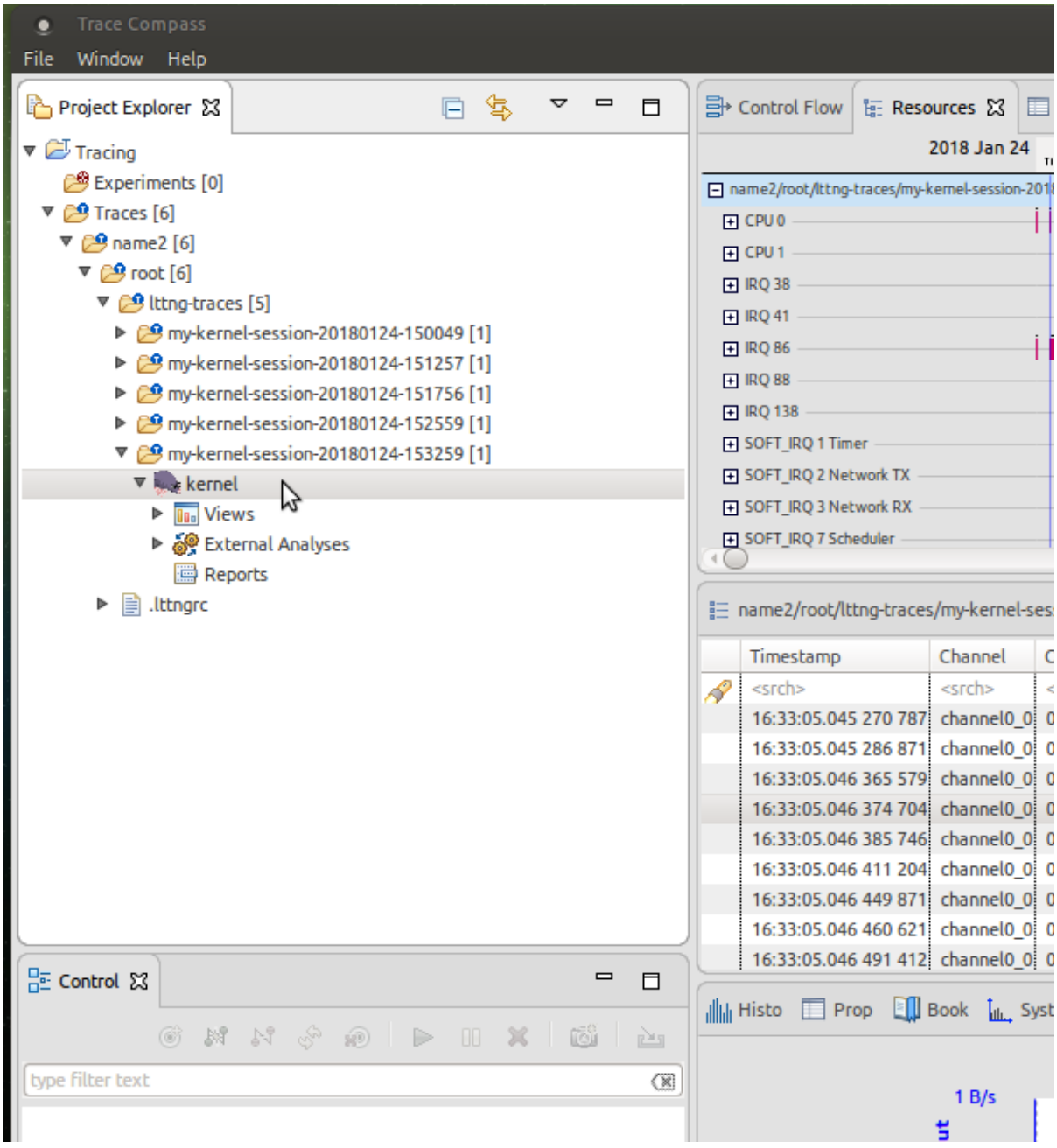
- End by clicking on finish button

Note that sometimes it fails to get logs at the first trial, do not hesitate to try again.



- Your traces will be loaded on Trace Compass, and you can see them on the Project explorer on the left.

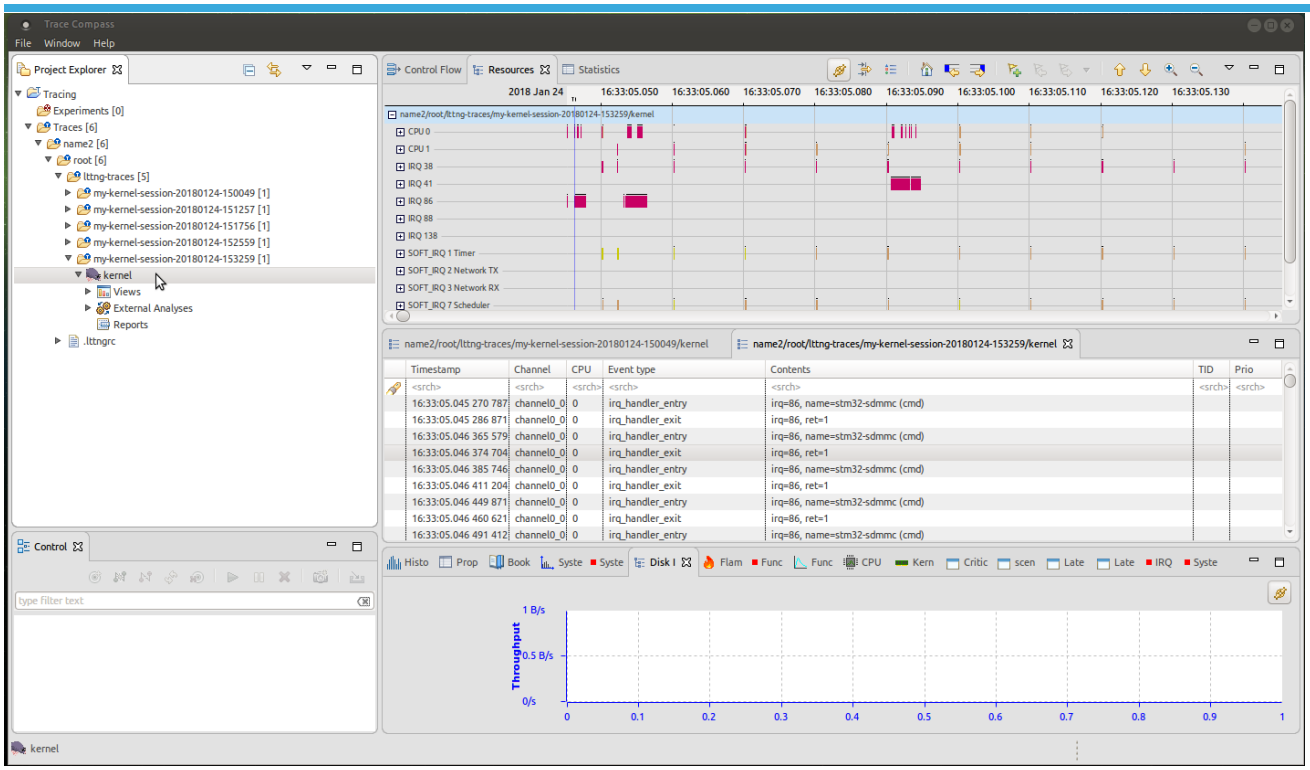
In the example below 5 traces have been loaded:



- Open the trace

To open the trace, left-click on the kernel item corresponding to your trace, then **Open**.

- Check for trace on the right panels


Trace Compass
File Window Help
Project Explorer
Tracing
Experiments [0]
Traces [6]
name2 [6]
root [6]
lttng-traces [5]
my-kernel-session-20180124-150049 [1]
my-kernel-session-20180124-151257 [1]
my-kernel-session-20180124-151756 [1]
my-kernel-session-20180124-152559 [1]
my-kernel-session-20180124-153259 [1]
kernel
Views
External Analyses
Reports
.lttngrc
Control Flow
Resources
Statistics
2018 Jan 24
16:33:05.050
16:33:05.060
16:33:05.070
16:33:05.080
16:33:05.090
16:33:05.100
16:33:05.110
16:33:05.120
16:33:05.130
CPU 0
CPU 1
IRQ 38
IRQ 41
IRQ 86
IRQ 88
IRQ 138
SOFT_IRQ 1 Timer
SOFT_IRQ 2 Network TX
SOFT_IRQ 3 Network RX
SOFT_IRQ 7 Scheduler
Timestamp
Channel
CPU
Event type
Contents
TID
Prio
16:33:05.045 270 787 channel0_0 0 irq_handler_entry irq=86, name=stm32-sdmmc (cmd)
16:33:05.045 286 871 channel0_0 0 irq_handler_exit irq=86, ret=1
16:33:05.046 365 579 channel0_0 0 irq_handler_entry irq=86, name=stm32-sdmmc (cmd)
16:33:05.046 374 704 channel0_0 0 irq_handler_exit irq=86, ret=1
16:33:05.046 385 746 channel0_0 0 irq_handler_entry irq=86, name=stm32-sdmmc (cmd)
16:33:05.046 411 204 channel0_0 0 irq_handler_exit irq=86, ret=1
16:33:05.046 449 871 channel0_0 0 irq_handler_entry irq=86, name=stm32-sdmmc (cmd)
16:33:05.046 460 621 channel0_0 0 irq_handler_exit irq=86, ret=1
16:33:05.046 491 412 channel0_0 0 irq_handler_entry irq=86, name=stm32-sdmmc (cmd)
Control
type filter text
Throughput
1 B/s
0.5 B/s
0/s
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
kernel

5 References

- [1.01.11.2] http://lttng.org
- [2.02.12.2] https://www.eclipse.org/tracecompass
- http://lttng.org/docs/v2.10/
- http://lttng.org/docs/v2.10/#doc-tracing-the-linux-kernel
- https://www.eclipse.org/tracecompass/#getting

Graphics Processing Units