Linux Trace Toolkit Next Generation Manual

Author : Mathieu Desnoyers, September 2005
Last update : January 30, 2010
(originally known as the LTTng QUICKSTART guide)

Table of Contents

Introduction

This document is made of five parts : the first one explains how to install LTTng and LTTV from sources, the second one describes the steps to follow to trace a system and view it. The third part explains briefly how to add a new trace point to the kernel and to user space applications. The fourth and last part explains how to create Debian or RPM packages from the LTTng and LTTV sources.

These operations are made for installing the LTTng 0.86 tracer on a linux 2.6.X kernel. You will also find instructions for installation of LTTV 0.12.x : the Linux Trace Toolkit Viewer. To see the list of compatibilities between LTTng, ltt-control, LTTV, please refer to : LTTng+LTTV versions compatibility The ongoing work had the Linux Kernel Markers integrated in the mainline Linux kernel since Linux 2.6.24 and the Tracepoints since 2.6.28. In its current state, the lttng patchset is necessary to have the trace clocksource, the instrumentation and the LTTng high-speed data extraction mechanism added to the kernel.

Licenses

LTTng, UST and LTTV are developed by an open community. LTTng is released under a dual Gnu LGPLv2.1/GPLv2 license, except for very few kernel-specific files which are derived work from the Linux kernel.

LTTV is available under the Gnu GPLv2. The low-level LTTV trace reading library is released under Gnu LGPLv2.1.

The Eclipse LTTng trace analysis tool is released under the EPL and uses the LTTV trace reading library (LGPLv2.1).

The UST (Userspace Tracing) and the Userspace RCU libraries are released under the LGPLv2.1 license, which allows linking with non-GPL (BSD, proprietary...) applications. The associated headers are released under MIT-style/BSD-style licenses.

Please refer to each particular file licensing for details.

Supported architectures

LTTng :

- x86 32/64 bits
- PowerPC 32 and 64 bits
- ARMv7 OMAP3
- Other ARM (with limited timestamping precision, e.g. 1HZ. Need architecture-specific support for better precision)
- MIPS
- sh (partial architecture-specific instrumentation)
- sparc64 (partial architecture-specific instrumentation)
- s390 (partial architecture-specific instrumentation)
- Other architectures supported without architecture-specific instrumentation and with low-resolution timestamps.

LTTV :

- Intel 32/64 bits
- PowerPC 32 and 64 bits
- Possibly others. Takes care of endianness and type size difference between the LTTng traces and the LTTV analysis tool.

Installation from sources

Prerequisites

Tools needed to follow the package download steps :

- wget
- bzip2
- gzip
- tar

You have to install the standard development libraries and programs necessary to compile a kernel :

```
(from Documentation/Changes in the Linux kernel tree)
```

```
Gnu C                2.95.3              # gcc --version
Gnu make             3.79.1              # make --version
binutils             2.12                # ld -v
util-linux           2.10o               # fdformat --version
module-init-tools    0.9.10              # depmod -V
```

You might also want to have libncurses5 to have the text mode kernel configuration menu, but there are alternatives.

Prerequisites for LTTV 0.x.x installation are :

```
gcc 3.2 or better
gtk 2.4 or better development libraries
  (Debian : libgtk2.0, libgtk2.0-dev)
  (Fedora : gtk2, gtk2-devel)
  note : For Fedora users : this might require at least core 3 from Fedora,
  or you might have to compile your own GTK2 library.
glib 2.4 or better development libraries
  (Debian : libglib2.0-0, libglib2.0-dev)
  (Fedora : glib2, glib2-devel)
libpopt development libraries
  (Debian : libpopt0, libpopt-dev)
  (Fedora : popt)
libpango development libraries
  (Debian : libpango1.0, libpango1.0-dev)
  (Fedora : pango, pango-devel)
libc6 development librairies
  (Debian : libc6, libc6-dev)
  (Fedora : glibc, glibc)
```

- Reminder

See the list of compatibilities between LTTng, ltt-control and LTTV at : LTTng+LTTV versions compatibility.

Getting the LTTng packages

```
su -
mkdir /usr/src/lttng
cd /usr/src/lttng
(see http://ltt.polymtl.ca/lttng for package listing)
wget http://ltt.polymtl.ca/lttng/patch-2.6.X-lttng-0.x.xx.tar.bz2
bzip2 -cd patch-2.6.X-lttng-0.x.xx.tar.bz2 | tar xvof -
```

Getting LTTng kernel sources

```
su -
cd /usr/src
wget http://kernel.org/pub/linux/kernel/v2.6/linux-2.6.X.tar.bz2
bzip2 -cd linux-2.6.X.tar.bz2 | tar xvof -
cd linux-2.6.X
- For LTTng 0.9.4- cat /usr/src/lttng/patch*-2.6.X-lttng-0.x.xx* | patch -p1
- For LTTng 0.9.5+ apply the patches in the order specified in the series file,
  or use quilt
cd ..
mv linux-2.6.X linux-2.6.X-lttng-0.x.xx
```

Installing a LTTng kernel

```
su -
cd /usr/src/linux-2.6.X-lttng-0.x.xx
make menuconfig (or make xconfig or make config)
    Select the < Help > button if you are not familiar with kernel
    configuration.
    Items preceded by [*] means they has to be built into the kernel.
    Items preceded by [M] means they has to be built as modules.
    Items preceded by [ ] means they should be removed.
  go to the "General setup" section
    Select the following options :
    [*] Prompt for development and/or incomplete code/drivers
    [*] Activate markers
    [*] Activate userspace markers ABI (experimental, optional)
```

```
    [*] Immediate value optimization (optional)
    [*] Linux Trace Toolkit Next Generation (LTTng) --->
    <M> or <*> Compile lttng tracing probes
    <M> or <*> Linux Trace Toolkit High-speed Lockless Data Relay
    <M> or <*> Linux Trace Toolkit Lock-Protected Data Relay
    <M> or <*> Linux Trace Toolkit Serializer
    <M> or <*>   Linux Trace Toolkit Marker Control
    <M> or <*>     Linux Trace Toolkit Tracer
    [*] Align Linux Trace Toolkit Traces
    <M> or <*> Support logging events from userspace
    [*] Support trace extraction from crash dump
    <M> or <*> Linux Trace Toolkit Trace Controller
    <M> or <*> Linux Trace Toolkit State Dump
  Select <Exit>
  Select <Exit>
  Select <Yes>
make
make modules_install
(if necessary, create a initrd with mkinitrd or your preferate alternative)
(mkinitrd -o /boot/initrd.img-2.6.X-lttng-0.x.xx 2.6.X-lttng-0.x.xx)

-- on X86, X86_64
make install
reboot
Select the Linux 2.6.X-lttng-0.x.xx kernel in your boot loader.

-- on PowerPC
cp vmlinux.strip /boot/vmlinux-2.6.X-lttng-0.x.xx
cp System.map /boot/System.map-2.6.X-lttng-0.x.xx
cp .config /boot/config-2.6.X-lttng-0.x.xx
depmod -ae -F /boot/System.map-2.6.X-lttng-0.x.xx 2.6.X-lttng-0.x.xx
mkinitrd /boot/initrd.img-2.6.X-lttng-0.x.xx 2.6.X-lttng-0.x.xx
(edit /etc/yaboot.conf to add a new entry pointing to your kernel : the entry
that comes first is the default kernel)
ybin
select the right entry at the yaboot prompt (see choices : tab, select : type
the kernel name followed by enter)
Select the Linux 2.6.X-lttng-0.x.xx kernel in your boot loader.
--
```

Editing the system wide configuration

You must activate debugfs and specify a mount point. This is typically done in fstab such that it happens at boot time. If you have never used DebugFS before, these operation would do this for you :

```
mkdir /mnt/debugfs
cp /etc/fstab /etc/fstab.lttng.bkp
echo "debugfs         /mnt/debugfs    debugfs rw              0       0"  >> /etc/fstab
```

then, rebooting or issuing the following command will activate debugfs :

```
mount /mnt/debugfs
```

You need to load the LTT modules to be able to control tracing from user space. This is done by issuing the following commands. Note however these commands load all LTT modules. Depending on what options you chose to compile statically, you may not need to issue all these commands.

```
modprobe ltt-trace-control
modprobe ltt-marker-control
modprobe ltt-tracer
modprobe ltt-serialize
modprobe ltt-relay
modprobe ipc-trace
modprobe kernel-trace
modprobe mm-trace
modprobe net-trace
modprobe fs-trace
modprobe jbd2-trace
modprobe ext4-trace
modprobe syscall-trace
modprobe trap-trace
#if locking tracing is wanted, uncomment the following
#modprobe lockdep-trace
```

If you want to have complete information about the kernel state (including all the process names), you need to load the ltt-statedump module. This is done by issuing the command :

```
modprobe ltt-statedump
```

You can automate at boot time loading the ltt-control module by :

```
cp /etc/modules /etc/modules.bkp
echo ltt-trace-control >> /etc/modules
echo ltt-marker-control >> /etc/modules
echo ltt-tracer >> /etc/modules
echo ltt-serialize >> /etc/modules
echo ltt-relay >> /etc/modules
echo ipc-trace >> /etc/modules
echo kernel-trace >> /etc/modules
echo mm-trace >> /etc/modules
echo net-trace >> /etc/modules
echo fs-trace >> /etc/modules
echo jbd2-trace >> /etc/modules
echo ext4-trace >> /etc/modules
echo syscall-trace >> /etc/modules
echo trap-trace >> /etc/modules
#if locking tracing is wanted, uncomment the following
#echo lockdep-trace >> /etc/modules
```

Getting and installing the ltt-control package (on the traced machine)

(note : the ltt-control package contains lttd and lttctl. Although it has the same name as the ltt-control kernel module, they are *not* the same thing.)

```
su -
cd /usr/src
wget http://ltt.polymtl.ca/lttng/ltt-control-0.x-xxxx2006.tar.gz
gzip -cd ltt-control-0.x-xxxx2008.tar.gz | tar xvof -
cd ltt-control-0.x-xxxx2006
(refer to README to see the development libraries that must be installed on you
system)
./configure
make
make install
# (run ldconfig to ensure new shared objects are taken into account)
ldconfig
```

Userspace tracing

```
Make sure you selected the kernel menuconfig option :
    <M> or <*> Support logging events from userspace
And that the ltt-userspace-event kernel module is loaded if selected as a
module.

Simple userspace tracing is available through
echo "some text to record" > /mnt/debugfs/ltt/write_event

It will appear in the trace under event :
channel : userspace
event name : event
```

Getting and installing the LTTV package (on the visualisation machine, same or different from the visualisation machine)

```
su -
cd /usr/src
wget http://ltt.polymtl.ca/packages/lttv-0.x.xx-xxxx2008.tar.gz
gzip -cd lttv-0.x.xx-xxxx2008.tar.gz | tar xvof -
cd lttv-0.x.xx-xxxx2008
(refer to README to see the development libraries that must be installed on your
system)
./configure
make
make install
```

```
# (run ldconfig to ensure new shared objects are taken into account)
ldconfig
```

Using LTTng and LTTV

- **IMPORTANT : Arm Linux Kernel Markers after each boot before tracing**

```
ltt-armall
```

Use graphical LTTV to control tracing and analyse traces

```
lttv-gui (or /usr/local/bin/lttv-gui)
  - Spot the "Tracing Control" icon : click on it
      (it's a traffic light icon)
    - enter the root password
    - click "start"
    - click "stop"
    - Yes
      * You should now see a trace
```

Use text mode LTTng to control tracing

```
The tracing can be controlled from a terminal by using the lttctl command (as
root).

Start tracing :

lttctl -C -w /tmp/trace1 trace1

Stop tracing and destroy trace channels :

lttctl -D trace1

see lttctl --help for details.
```

(note : to see if the buffers has been filled, look at the dmesg output after lttctl -D or after stopping tracing from the GUI, it will show an event lost count. If it is the case, try using larger buffers. See lttctl --help to learn how. lttv now also shows event lost messages in the console when loading a trace with missing events or lost subbuffers.)

Use text mode LTTV

Feel free to look in /usr/local/lib/lttv/plugins to see all the text and graphical plugins available.

For example, a simple trace dump in text format is available with :

```
lttv -m textDump -t /tmp/trace
```

See lttv -m textDump --help for detailed command line options of textDump.

It is, in the current state of the project, very useful to use "grep" on the text output to filter by specific event fields. You can later copy the timestamp of the events to the clipboard and paste them in the GUI by clicking on the bottom right label "Current time". Support for this type of filtering should be added to the filter module soon.

Tracing in "Hybrid" mode

Starting from LTTng 0.5.105 and ltt-control 0.20, a new mode can be used : hybrid. It can be especially useful when studying big workloads on a long period of time.

When using this mode, the most important, low rate control information will be recorded during all the trace by lttd (i.e. process creation/exit). The high rate information (i.e. interrupt/traps/syscall entry/exit) will be kept in a flight recorder buffer (now named flight-channelname_X).

The following lttctl commands take an hybrid trace :

Create trace channel, start lttd on normal channels, start tracing:

```
lttctl -C -w /tmp/trace2 -o channel.kernel.overwrite=1 trace2
```

Stop tracing, start lttd on flight recorder channels, destroy trace channels :

```
lttctl -D -w /tmp/trace2 trace2
```

Each "overwrite" channel is flight recorder channel.

Tracing in flight recorder mode

- Flight recorder mode

The flight recorder mode writes data into overwritten buffers for all channels, including control channels, except for the facilities tracefiles. It consists of setting all channels to "overwrite".

The following lttctl commands take a flight recorder trace :

```
lttctl -C -w /tmp/trace3 -o channel.all.overwrite=1 trace3
...
lttctl -D -w /tmp/trace3 trace3
```

Adding new instrumentations with the markers

Adding kernel instrumentation

See Documentation/markers.txt and Documentation/trace/tracepoints.txt in your kernel tree.

Also see ltt/probes/ for LTTng probe examples.

Adding userspace instrumentation

Add new events to userspace programs with userspace markers packages. Get the latest markers-userspace-*.tar.bz2 and see the Makefile and examples. It allows inserting markers in executables and libraries, currently only on x86_32 and x86_64. See markers-userspace-0.5.tar.bz2 or more recent.

Note that a new design document for a 3rd generation of tracepoint/marker-based userspace tracing is available at LTTng User-space Tracing Design. This new infrastructure is not yet implemented.

The easy quick-and-dirty way to perform userspace tracing is currently to write an string to /mnt/debugfs/ltt/write_event. See Userspace tracing in the installation for sources section of this document.

Creating Debian or RPM packages

Create custom LTTV Debian packages

```
Use : dpkg-buildpackage -rfakeroot
```

You should then have your LTTV .deb files created for your architecture.

Create custom LTTng packages

For building LTTng Debian packages : get the build tree with patches applies as explained in section 2.

```
make menuconfig (or xconfig or config) (customize your configuration)
make-kpkg kernel_image
```

You will then see your freshly created .deb in /usr/src. Install it with

```
dpkg -i /usr/src/(image-name).deb
```

Then, follow the section "Editing the system wide configuration" in section 2.
