LiTL

Lightweight Trace Library

User Manual

Roman Iakymchuk and François Trahay

August 9, 2013

Contents

Chapter 1

License of LiTL

LiTL is developed and distributed under the GNU General Public License.

This library is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

LiTL is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with LiTL; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Chapter 2

LiTL

LiTL [1] is a lightweight binary trace library that aims at providing performance analysis tools with a scalable event recording mechanism that utilizes minimum resources of the CPU and memory. In order to efficiently analyze modern HPC applications that combine OpenMP (or Pthreads) threads and MPI processes, we design and implement various mechanisms to ensure the scalability of LiTL for a large number of both threads and processes.

LiTL is designed in order to resolve the following performance tracing issues:

- Scalability and the number of threads;
- Scalability and the number of recorded traces;
- Optimization in the storage capacity usage.

As a results, LiTL provides similar functionality to standard event recording libraries and records events only from user-space. LiTL minimizes the usage of the CPU time and memory space in order to avoid disturbing the application that is being analyzed. Also, LiTL is fully thread-safe that allows to record events from multi-threaded applications. Finally, LiTL is a generic library that can be used in conjunction with many performance analysis tools.

Chapter 3

Installation

3.1 Requirements

In order to use LiTL, the following software is needed:

1. autoconf 2.63;
2. libelf or libbfd. On Debian, libelf can be installed from command line by the following command:
   `apt-get install libelf-dev`

3.2 Getting LiTL

Current development version of LiTL is available via Git

```
git clone git+ssh://fusionforge.int-evry.fr//var/lib/
gforge/chroot/scmrepos/git/litl/litl.git
```

After getting the latest development version from Git,

```
./bootstrap
```

should be run in the root directory and only then the tool can be built.

3.3 Building EZTrace

At first, to configure LiTL the following configure script should be invoked:

```
./configure -prefix=<LITL_INSTALL_DIR>
```

The configuration script contains many different options that can be set. However, during the first try we recommend to use the default settings.

Once LiTL is configured, the next two commands should be executed:

```
make
make install
```

In order to check whether LiTL was installed correctly, a set of tests can be run as

```
make check
```

Chapter 4

LiTL in Details

Chapter 5

How to Use LiTL

5.1 Reading Events

After the application was traced and events were recorded into binary trace files, those traces can be analyzed using `litl_read` as

```
litl_read -f trace.file
```

This utility shows the recorded events in the following format:

- Time since last probe record on the same CPU;
- ID of the current thread on this CPU;
- Event type;
- Code of the probe;
- Number of parameters of the probe;
- List of parameters of the probe, if any.

5.2 Merging Traces

Once the traces were recorded, they can be merged into an archive of traces for further processing by the following command

```
litl_read -o archive.trace trace.0 trace.1 ...  trace.n
```

5.3 Splitting Traces

If there is a need for a detail analysis of a particular trace files, an archive of traces can be split back into separate trace files by

```
litl_read -f archive.trace -d output.dir
```

Chapter 6

Environment Variables

For a more flexible and comfortable usage of LiTL, we provide three environment variables:

- `LITL_BUFFER_FLUSH` specifies the behavior of LiTL when the event buffer is full. If it is set to one, which is a default option, the buffer is flushed. This permits to record traces that are larger than the buffer size. Otherwise, if it is set to zero any additional event will be recorded. The trace is, thus, truncated and there is no impact on the application performance;
- `LITL_THREAD_SAFETY` specifies the behavior of LiTL while tracing multi-threaded applications. If it is set to one, which is a default value, the thread safety is enabled. Otherwise, when it is set to zero, the event recording is not thread safe;
- `LITL_TIMING_METHOD` specifies the timing method that will be used during the recording phase. The LiTL timing methods can be divided into two groups: those that measure time in clock ticks and those that rely on `clock_gettime()` function. The first has one method:
  - `ticks` that uses the CPU specific register, e.g. rdtsc on X86 and X86_64 architectures.

  The second has five different possibilities:
  - `monotonic` that corresponds to `CLOCK_MONOTONIC`;
  - `monotonic_raw` – `CLOCK_MONOTONIC_RAW`;
  - `realtime` – `CLOCK_REALTIME`;
  - `thread_cputime` – `CLOCK_THREAD_CPUTIME_ID`;
  - `process_cputime` – `CLOCK_PROCESS_CPUTIME_ID`.

  User can also define its own timing method and set the environment variable accordingly.

Chapter 7

Troubleshooting

If you encounter a bug or want some explanation about LiTL, please contact and ask our development team on the development mailing list

- `litl-devel@fusionforge.int-evry.fr`.

Bibliography

[1] Roman Iakymchuk and François Trahay. LiTL: Lightweight trace library. In *Proceedings of the 25th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'2013), Porto de Galinhas, Pernambuco, Brazil, October 23-26*, 2013.