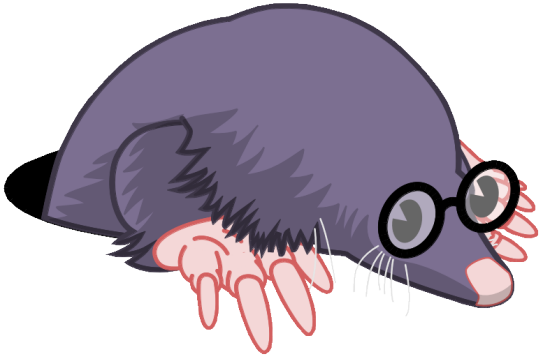
Polytechnique Montréal – May 2018



What’s new at EfficiOS ?

EfficiOS

Outline

- Kernel Contributions,
- LTTng Scope,
- LTTng 2.11,
- Babeltrace.

Kernel Contributions

- Memory Barriers (membarrier(2)),
- Restartable sequence (rseq),
- CPU operation vectors (cpu_opv).

membarrier

- Membarrier is a system call issuing a memory barrier on a set of threads,
- New commands:
  - *Private expedited* (4.14),
  - *Shared* renamed to *Global* (4.16),
  - *Global expedited* (4.16),
  - *Private expedited sync core* for JIT reclaim (4.16).
- Provide LTTng-UST ring buffer and liburcu read-side performance enhancement.

EfficiOS

Restartable Sequences

- Restartable Sequences (rseq) is a newly proposed system call which accelerates user-space operations on per-cpu data, e.g.:
  - LTTng-UST ring buffer,
  - Liburcu per-cpu flavor (for multi-process RCU over shared memory),
  - Facebook's jemalloc,
  - Performance Monitoring Unit counters on arm64,
- A TLS area is registered for each thread, and then shared between kernel and user-space. It allows restarting user-space assembly instruction sequences if preempted, migrated, or interrupted by signal delivery. It also provides a copy of the current CPU number which is always kept up to date by the kernel, readable from user-space.

Restartable Sequences Limitations

- Limitations:
  - Debugger instruction and line-level single-stepping triggers restarts (infinite retry loops),
    - Can be mitigated with __rseq_table section if used by future debugger implementation to skip rseq critical sections,
  - Unable to target a specific CPU: executes on the current CPU,
  - Requires application to provide a fallback using separate data, e.g.:
    - Split-counters,
    - memory allocation from a memory pool based on another synchronization mechanism,
    - use a system call to read performance counters.

CPU operation vectors

- CPU operation vector (cpu_opv) is a small interpreter in the kernel,
- Get references to pages backing all memory targeted by operations first, taking page faults if needed,
- Migrates current thread to specific CPU if needed,
- Disables preemption, and executes the sequences of operations “atomically” with respect to scheduler preemption and migration,
- Handles CPU hotplug (off-line CPU) by disabling CPU hotplug temporarily (read-side hotplug lock), and using a mutex providing mutual exclusion,
- Can be used as slow-path fallback taking care of rseq fast-path limitations,
- rseq is planned to be proposed independendly first (4.18 ?) and then cpu_opv may be proposed in the future to fulfill the missing requirements.

LTTng Scope

- First official release (LTTng Scope 0.3),
- Base feature set:
  - Detailed event record list,
  - Thread and CPU timeline views,
  - Project-wide event highlighting,
  - Easy installer bundle.

EfficiOS

LTTng Scope

File View

auto-20180226-12084317604
Traces
auto-20180226-120843
Filters

Event count

4,000
3,000
2,000
1,000
0

Threads | Arrow Series | Sorting Modes | Filter Modes

auto-20180226-12084317
0/2 - swapper/2
0/0 - swapper/0
0/3 - swapper/3
0/1 - swapper/1
1 - systemd
2 - kthreadd
3 - kworker/0:0
4 - kworker/0:0H

CPU | Arrow Series | Sorting Modes | Filter Modes

auto-20180226-12084317
CPU 0
CPU 1
CPU 2
CPU 3

Timestamp	Trace	CPU	Event Type	Event Fields
12:08:49.483336240	auto-20180...	1	kmem_kmalloc	[call_site=0xffffffffc0cee329, ptr=0xffffa07d24958400, bytes_req=739, bytes_alloc=1024, gfp_f
12:08:49.483342238	auto-20180...	1	kmem_kfree	[call_site=0xffffffffc0cee396, ptr=0xffffa07d24958400]
12:08:49.483343462	auto-20180...	1	sched_waking	[comm=lttng-consumerd, tid=3781, prio=20, target_cpu=1]
12:08:49.483344366	auto-20180...	1	sched_migrate_task	[comm=lttng-consumerd, tid=3781, prio=20, orig_cpu=1, dest_cpu=2]
12:08:49.483345850	auto-20180...	1	sched_wakeup	[comm=lttng-consumerd, tid=3781, prio=20, target_cpu=2]
12:08:49.483346620	auto-20180...	2	power_cpu_idle	[state=4294967295, cpu_id=2]
12:08:49.483347514	auto-20180...	1	kmem_kmalloc	[call_site=0xffffffffc0cee329, ptr=0xffffa07d24bbf800, bytes_req=273, bytes_alloc=512, gfp_fla

	Start	End	Span (s)
Visible Time Range	12:08:49.483336240	12:08:49.583336240	0.100000000
Selection Time Range	12:08:49.483336240	12:08:49.483336240	0.000000000
Trace Project Time Range	2018-02-26 12:08:49.483336240 -05:00	2018-02-26 12:09:07.012245821 -05:00	17.528909581

Ready

EfficiOS

To be expected for LTTng Scope

- LTTng Scope 0.4:
  - Multi-trace projects,
  - Project state persistence.
- LTTng Scope 0.5:
  - Callstack view for LTTng-UST traces,
  - Adding and removing widgets in the main view.

EfficiOS

LTTng 2.11

- New features:
  - Session rotation,
  - Dynamic instrumentation,
  - Filtering on array and sequence integers in LTTng-UST and LTTng-modules.

EfficiOS

LTTng 2.11 – Session Rotation

- Allow processing of portion of the trace without stopping tracing,
- Split trace in self contained-traces on the fly,
- Allows for pipelining and/or sharding of analyses,
- Encryption, compression, cleanup of old chunks, integration with external message bus tools.

EfficiOS

LTTng 2.11 – Dynamic instrumentation

- Adding tracepoints without having to recompile or restart a process,
- Using the uprobe interface,
- Tracing userspace using the **kernel tracer**,
- Supported instrumentation point types:
  - ELF symbols,
  - SystemTap/SDT probe points (without semaphore).

```
lttng enable-event --kernel
  --userspace-probe=elf:/path/to/binary:symbol
   event_name
```

EfficiOS

LTTng 2.11 – Dynamic instrumentation

- Limitations:
  - Slower than LTTng-UST, because of context-switches to the kernel,
  - No tracepoint payload recorded at the moment.

Filtering on array and sequence of integers

- Filter out event based on the content of arrays and sequence

```
[14:32:57.03] host lttng_ust_prov:event : { _field_length = 4, field = [ [0]
= 121, [1] = 55, [2] = 23, [3] = 42 ] }
```

- Define filter using indexes in sequence:

```
lttng enable-event --userspace lttng_ust_prov:event
            --filter='field[0]<100 && field[3]==42'
```

EfficiOS

Babeltrace 2.0

- Flexible trace processing framework:
  - Graph-structured processing,
  - Customizable components,
  - API for out-of-tree components.
- Targeting comparable performance with Babeltrace 1.x before freezing APIs.

Babeltrace - Performance

- Reducing object allocation:
  - Object pooling.
- Removing precondition checks:
  - Introducing “Developer Mode”.

EfficiOS

Thank you!

Any questions?

EfficiOS