Another timestamp may be required to make synchronization possible between traces

08/08/2013 12:08 PM - Daniel U. Thibault

---

### Description

In order to preserve the assertion that successive events in an LTTng buffer are in strictly increasing timestamp order, it turns out the 'timestamp' of each event record is assigned by LTTng at a “late” stage, specifically when the buffer slot reservation is done.

This has the counter-intuitive effect that a single event to which several trace sessions subscribe will bear a different timestamp in each trace, because each event record instance may reserve space in each buffer at a different moment (this could occur with multiple sessions under a single session daemon, or when user-space events are captured by both a root session daemon and a user-space session daemon). The difference between timestamps can be quite large (up to 3 milliseconds in one test). This makes reconciliation/synchronization between different traces of the same system time slice quite difficult, if not impossible. (See lttng-dev Digest, Vol. 61, Iss. 46, Message 2, 13 May 2013 and follow-ups)

A possible solution would be to add another timestamp field to all LTTng events, named maybe 'event_timestamp' to distinguish it from the 'timestamp' field. This timestamp would be set as early as possible in the trace provider's processing of the event, well before the event record gets split into multiple instances. Trace viewers could then offer the option of sorting event using this field instead of the current 'timestamp' one, and the key assertion that "multiple instances of the same event will all bear the same event_timestamp" would make synchronization of traces a lot easier and more reliable.

---

### History

#### #1 - 08/08/2013 12:14 PM - Mathieu Desnoyers

- Status changed from New to Won't fix

this added timestamps would:

a) waste space/bandwidth at trace collection,
b) increase the overhead of the tracer, due to added time-stamp read,
c) not be monotonically increasing across a given stream (or would require very heavy synchronization against interrupt/signals/concurrent threads, generating even more overhead).

So I don't think this idea will ever be doable in practice. This should be handled by the viewer, not the tracer.

Thanks,

Mathieu

#### #2 - 08/08/2013 12:31 PM - Daniel U. Thibault

Mathieu Desnoyers wrote:

this added timestamps would:

a) waste space/bandwidth at trace collection,
b) increase the overhead of the tracer, due to added time-stamp read,
c) not be monotonically increasing across a given stream (or would require very heavy synchronization against interrupt/signals/concurrent threads, generating even more overhead).

So I don't think this idea will ever be doable in practice. This should be handled by the viewer, not the tracer.

Points a and b are quite true. Point c is irrelevant: you should think of the 'event_timestamp' as being just another payload field. As for the viewer handling the problem, that is impossible because the information required (the event_timestamp) is not available.

A user-space workaround would be to manually include 'event_timestamp' in one's definition of one's events, but this would work only for those events (and the viewers would be unaware of the significance of the additional timestamp field).

Maybe the 'event_timestamp' field could be an option one could turn on at trace session creation time?
It may be that the use cases that highlight this problem are so unusual that there will be no demand for this feature. But what worries me the most is what the experiment reveals: the sequence of events captured by LTTng may not be the true sequence. Analysts hunting for race conditions need to be aware of this. On the other hand, it may be that within an event stream the true event sequence is much better preserved, and it may even be that the only thing that is being lost here is the notion of simultaneity between trace sessions, so analysts need only be warned that they should not rely on synthetic traces (i.e. traces constructed by merging different trace sessions).

#3 - 08/08/2013 12:56 PM - Mathieu Desnoyers

• bugs@lttnq.org (bugs@lttnq.org) wrote:

Issue #614 has been updated by Daniel U. Thibault.

[...]

Points a and b are quite true. Point c is irrelevant: you should think of the 'event_timestamp' as being just another payload field.

The viewers need to use time for "seek" functions, which heavily rely on those being monotonically increasing within a stream. Timestamps are not "just another field".

As for the viewer handling the problem, that is impossible because the information required (the event_timestamp) is not available.

The viewer could have an upper-bound (+/- number of nanoseconds, sometimes extended by knowledge of the trace, e.g. that interrupts and scheduling can cause large delays).

So it's not "impossible" to handle.

[...]

It may be that the use cases that highlight this problem are so unusual that there will be no demand for this feature. But what worries me the most is what the experiment reveals: the sequence of events captured by LTTng may not be the true sequence.

I've got news for you: there is no true sequence. At least, not when we don't "stop the entire world" when we do tracing. And we don't want to have such a large impact. An event can nest on top of tracing due to signals, interruption, preemption, and making this fully ordered would just kill performances, so we don't.

So we only get a relative order of events, that's about it.

Analysts hunting for race conditions need to be aware of this.

Yes. They should.

Thanks,

Mathieu

--

Mathieu Desnoyers
EfficiOS Inc.
http://www.efficios.com

#4 - 08/08/2013 01:15 PM - Daniel U. Thibault

Mathieu Desnoyers wrote:

• bugs@lttnq.org (bugs@lttnq.org) wrote:

Issue #614 has been updated by Daniel U. Thibault.

[...]
Points a and b are quite true. Point c is irrelevant: you should think of the 'event_timestamp' as being just another payload field.

The viewers need to use time for "seek" functions, which heavily rely on those being monotonically increasing within a stream. Timestamps are not "just another field".

I meant that it's "just another field" for LTTng, because it has no consequences for the management of the buffers, unlike the 'timestamp'. The proposed field would have come into play only at the viewer level. You're quite right that any viewer which chose to switch its trace time sort key from one timestamp to the other would face a large transient work load. For a closed trace (meaning a non-live one), this processing would need to be done just once, thankfuly.

#5 - 08/08/2013 01:56 PM - Anonymous

Daniel brings up a good point. Having the "same" event in separate traces but with different timestamps could indeed be a bit confusing at the analysis level. But I agree that it would add complexity in the tracer that would not be needed in the majority of cases.

Perhaps we can deal with it at the viewer level by using some kind of heuristics to recognize that some events are the same (ie, "their timestamps are very close, and their source cpu and payload are exactly the same: it must be the same event." etc.)