HITACHI
Inspire the Next

Ftrace Event Tracer and Enhancement for Flight Recorder

LinuxCon Japan 2012

Yokohama Research Laboratory
Hitachi, Ltd.

Hiraku Toyooka <hiraku.toyooka.gu@hitachi.com>

Agenda

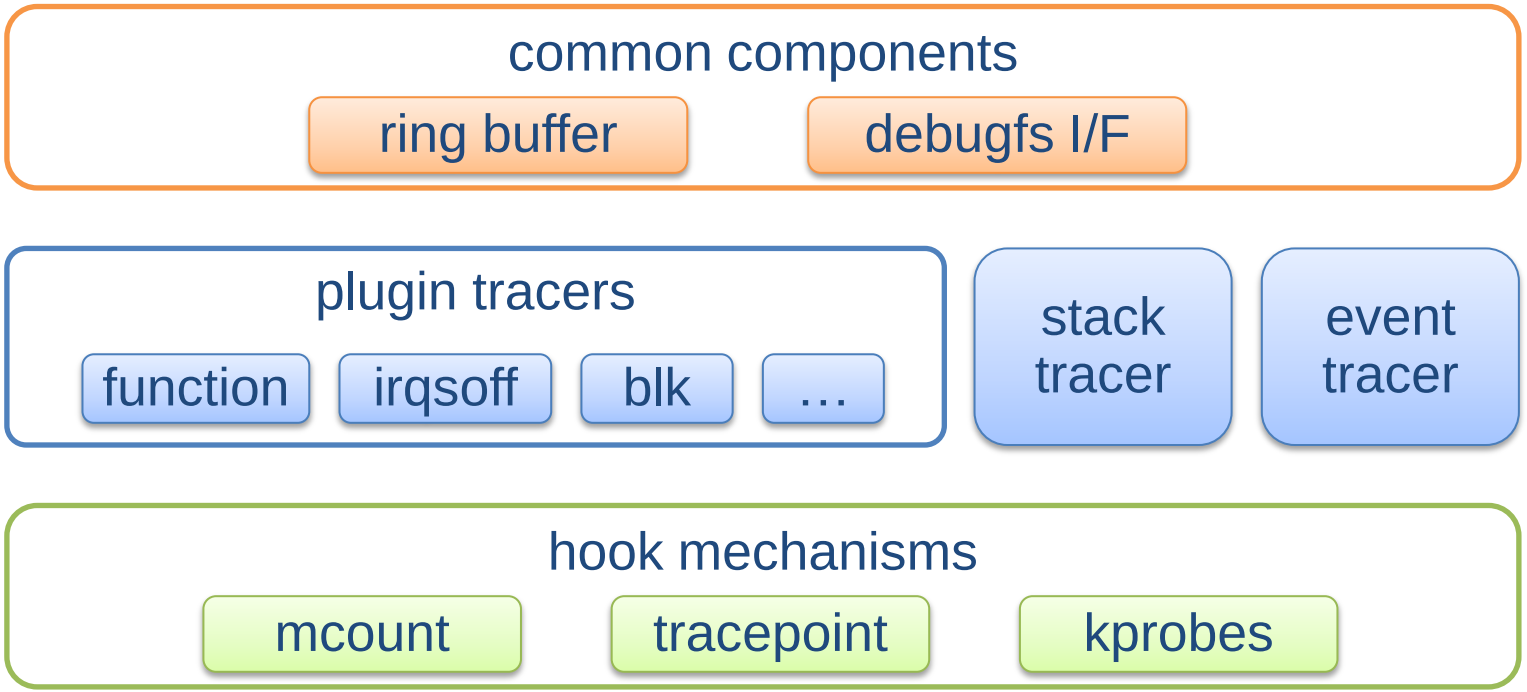
HITACHI
Inspire the Next

- Ftrace event tracer
- Event tracer as a flight recorder
- Introducing 2 features
  - Snapshot & Multiple ring buffer
  - Why these are necessary
  - Interface
- Future plan
- Conclusion

What is ftrace?

HITACHI
Inspire the Next

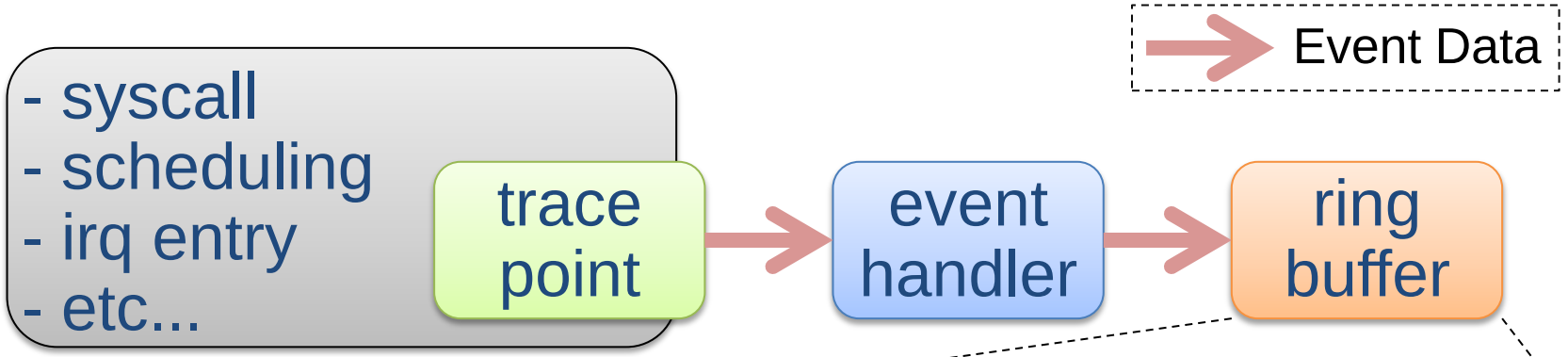
- Ftrace is a framework for kernel tracing
  - Each “tracer” performs meaningful tracing
  - (Started as a function tracer, but it’s currently one of the tracers)



What is ftrace event tracer? (a.k.a. “events”)

HITACHI
Inspire the Next

- Record events when kernel steps on “tracepoint” embedded in kernel




```
gnome-panel-1716 [001] 11970.096184: sched_stat_runtime: comm=gnome-panel pid=1716 runtime=31888 [ns]
  trace-cmd-3844 [002] 11970.096185: sched_stat_runtime: comm=trace-cmd pid=3844 runtime=230692 [ns]
  trace-cmd-3844 [002] 11970.096188: sched_switch:       trace-cmd:3844 [120] S ==> swapper/2:0 [120]
  ome-panel-1716 [001] 11970.096188: sched_switch:       gnome-panel:1716 [120] S ==> swapper/1:0 [120]
         ls-3845 [003] 11970.096192: sched_wakeup:       migration/3:17 [0] success=1 CPU:003
         ls-3845 [003] 11970.096193: sched_stat_runtime: comm=trace-cmd pid=3845 runtime=93127 [ns]
         ls-3845 [003] 11970.096194: sched_switch:       trace-cmd:3845 [120] R ==> migration/3:17 [0]
    gration/3-17 [003] 11970.096196: sched_stat_wait:    comm=trace-cmd pid=3845 delay=4131 [ns]
.....
```

Available events

HITACHI
Inspire the Next

- static events (tracepoint-based)
  - sched
  - kmem
  - irq (incl. softirq)
  - ext3, ext4, jbd, block
  - kvm, xen
  - syscall (enter/exit)
  - etc...

more than 360 events
in 3.4.0-rc4
(except syscalls)

- dynamic events
  - kprobes-based trace events (2.6.33~)

HITACHI
Inspire the Next

Debugfs files for getting event data or settings

```
tracing
├── events
│   ├── sched
│   │   ├── enable
│   .   ├── sched_switch
│   .   .   ├── enable
│
├── options
│   ├── overwrite
│   .
│   .
│
├── per_cpu
│   ├── cpu0
│   .   ├── trace
│   .   ├── trace_pipe
│
├── trace
├── trace_pipe
```

Debugfs I/F (Cont.)

HITACHI
Inspire the Next

- events/*event_class*/*event_name*/enable
  - for enabling/disabling a specific event (or event class)

```
# echo 1 > events/kmem/kmalloc/enable
```

```
# echo 1 > events/kmem/enable
```

- options/overwrite
  - for enabling/disabling overwrite mode of ring buffer
  - When the ring buffer is full,

    1: oldest events are discarded (default)

    0: newest events are discarded

Debugfs I/F (Cont.)

HITACHI
Inspire the Next

- trace
  - for reading a ring buffer (all per-cpu buffers)

```
# cat trace
```

  - Read doesn't consume event data in the buffer

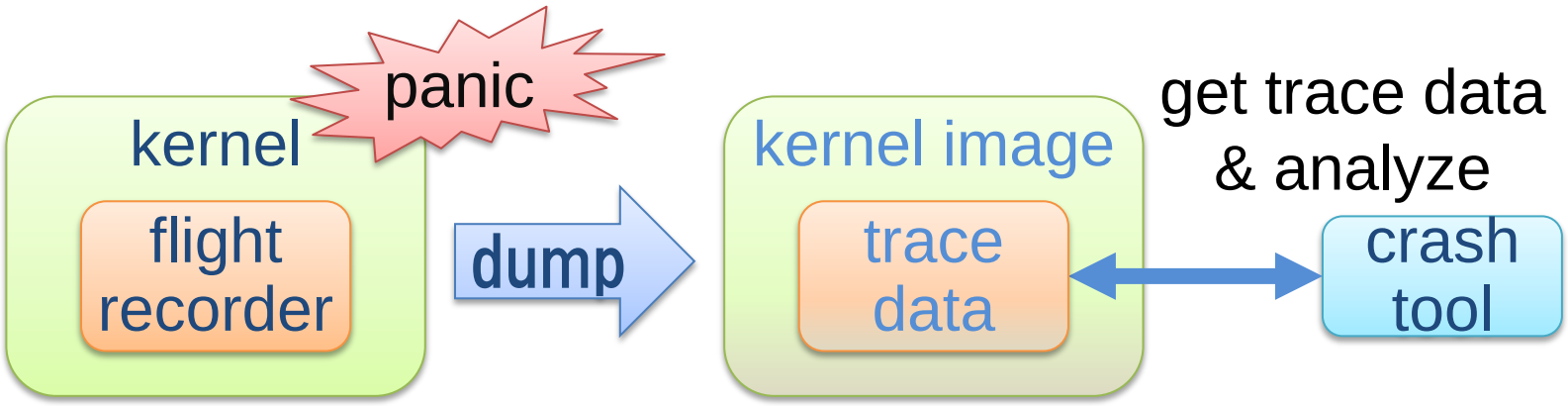
- trace_pipe
  - similar to "trace"
  - Read consumes event data in the buffer

- per_cpu/cpu*X*/trace
  - for reading each per-cpu ring buffer

HITACHI
Inspire the Next

Event tracer is available as a flight recorder

- record event data at all times system is running
- use overwrite mode buffer (= discard old events)
- stop tracing on critical errors (panic), and we can analyze failure causes



Far more requirements

HITACHI
Inspire the Next

Event tracer is useful as a flight recorder, but…

- It's difficult to handle non-critical errors (such as application's errors or fail-over of bonding driver)
  - the system has to continue to run, so the system can't stop trace
  - on the other hand, failure analysis is necessary to prevent the same errors

- It's difficult to satisfy above 2 requirements in current event tracer
  - (I'll explain in detail later)

Proposal of new features

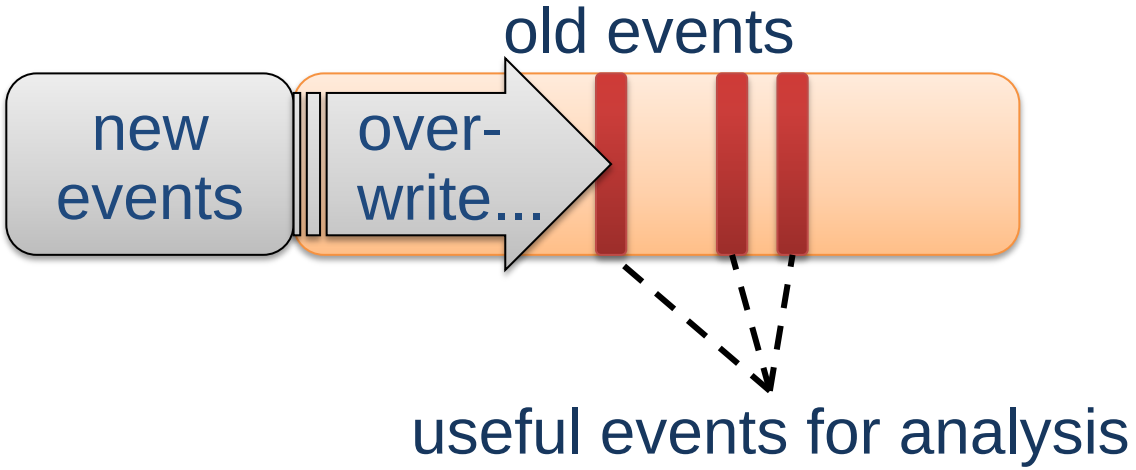
HITACHI
Inspire the Next

- In order to solve those problems, I propose following features
  - Snapshot
  - Multiple ring buffer

Snapshot – Why is it necessary?

HITACHI
Inspire the Next

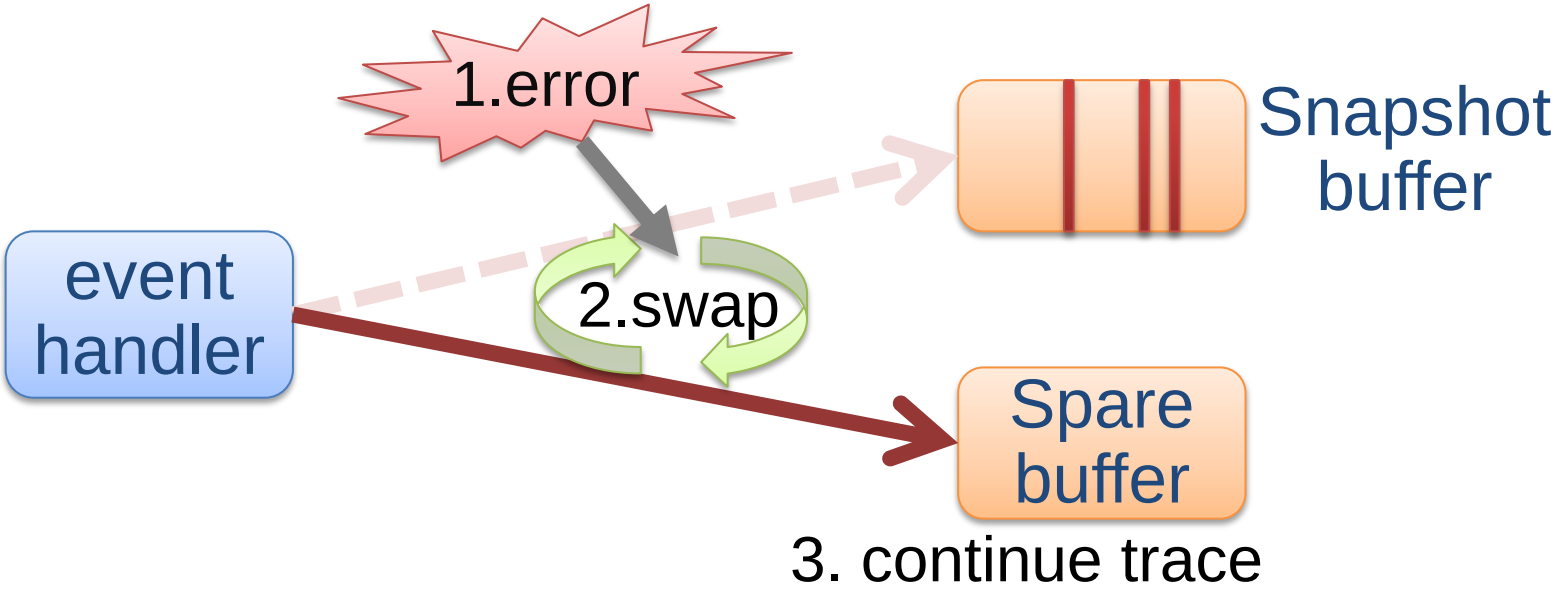
- After a recoverable error happens,
  - in case we stop trace, next error events can't be recorded
  - in case we continue trace, useful events for error analysis may be overwritten by new events



- It's necessary to save ring buffer on errors while enabling trace -> **Snapshot!**

Snapshot – What is it?

- Swapping a buffer for a spare buffer
- Snapshot buffer can be read from userspace
- We can continue trace across the swapping



- Fortunately, swapping mechanism already exists
  - irqsoff and wakeup tracers are using it

Snapshot – Interface

HITACHI
Inspire the Next

- Errors detected by application can be trigger
- I propose following 2 debugfs files

"snapshot_enabled"

- enable snapshot (prepare a spare buffer)

```
# echo 1 > snapshot_enabled
```

- disable snapshot (shrink a spare buffer)

```
# echo 0 > snapshot_enabled
```

"snapshot"

- take a snapshot

```
# echo 1 > snapshot
```

- read a snapshot

```
# cat snapshot
```

Snapshot – Remaining problem

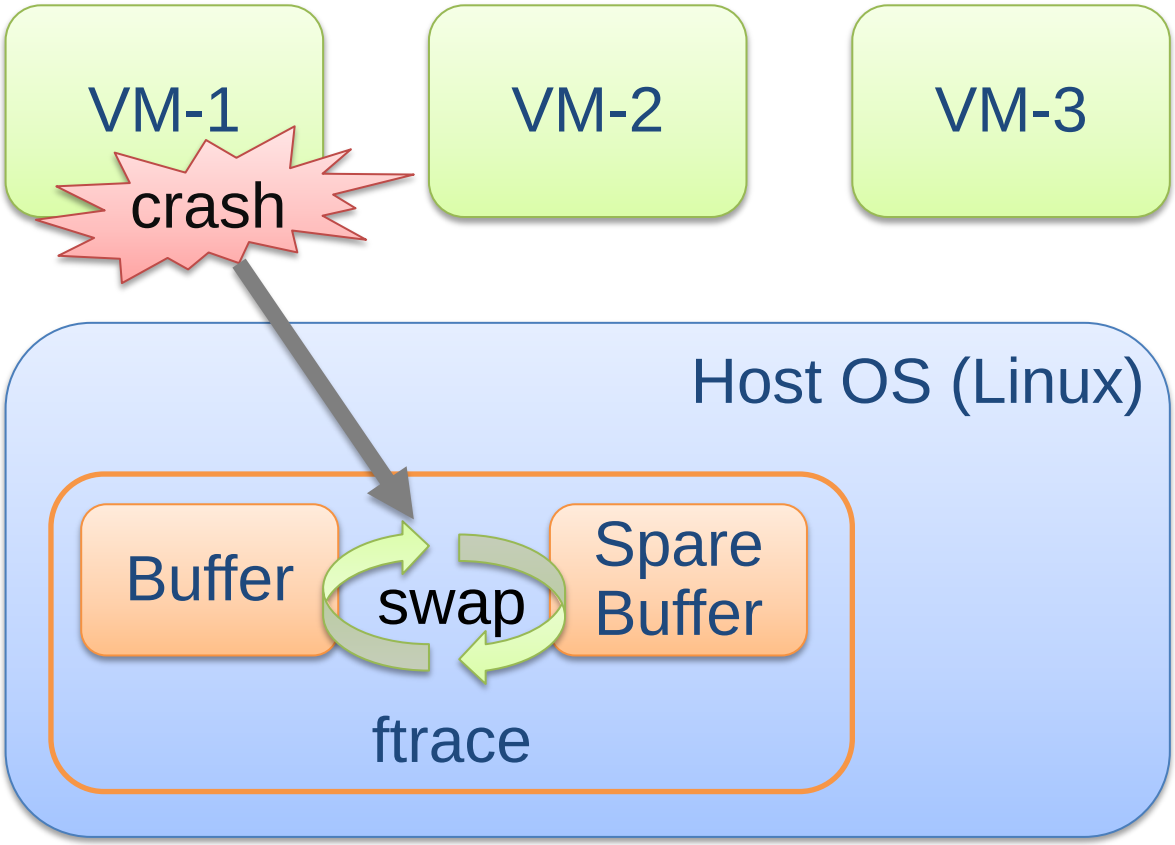
HITACHI
Inspire the Next

- Errors detected only by kernel
  - e.g. Exceptions, fail-over of bonding driver
  - How should we catch those errors?
    - add exception trace events and use them as trigger?
    - or other way?

Snapshot – Other use case

HITACHI
Inspire the Next

- Snapshot is useful in virtualization(KVM)
  - Host OS's trace data is useful for failure analysis of VMs
  - Host OS can't be stopped even in a VM's crash



Multiple ring buffer – Why is it necessary?

HITACHI
Inspire the Next

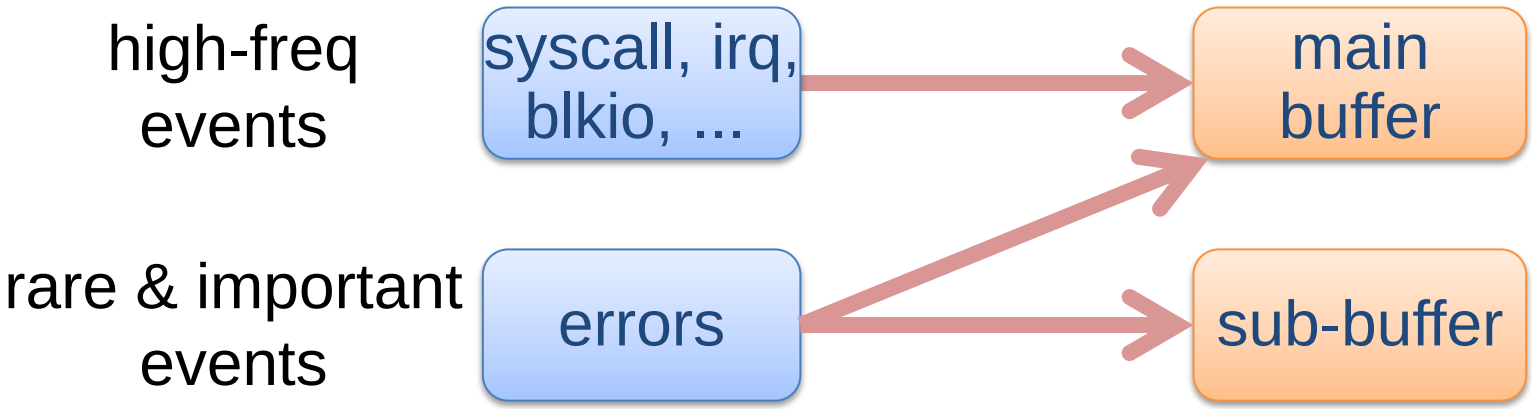
- Current event tracer can record events to only one ring buffer
- When an error event happens, the error event could be overwritten by other high-freq events
  - Error events are so rare and important that even only those events should be preserved
  - Snapshot is useful for one error, but can't deal with multiple errors

- It's necessary to protect error events from high-freq events -> **Multiple ring buffer!**

Multiple ring buffer – What is it?

HITACHI
Inspire the Next

- A mechanism to increase the number of ring buffers on demand
- We can separate (or replicate) important events into sub-buffer(s)



- Important and rare events leave in sub-buffer over a long time

Multiple trace buffer – Interface

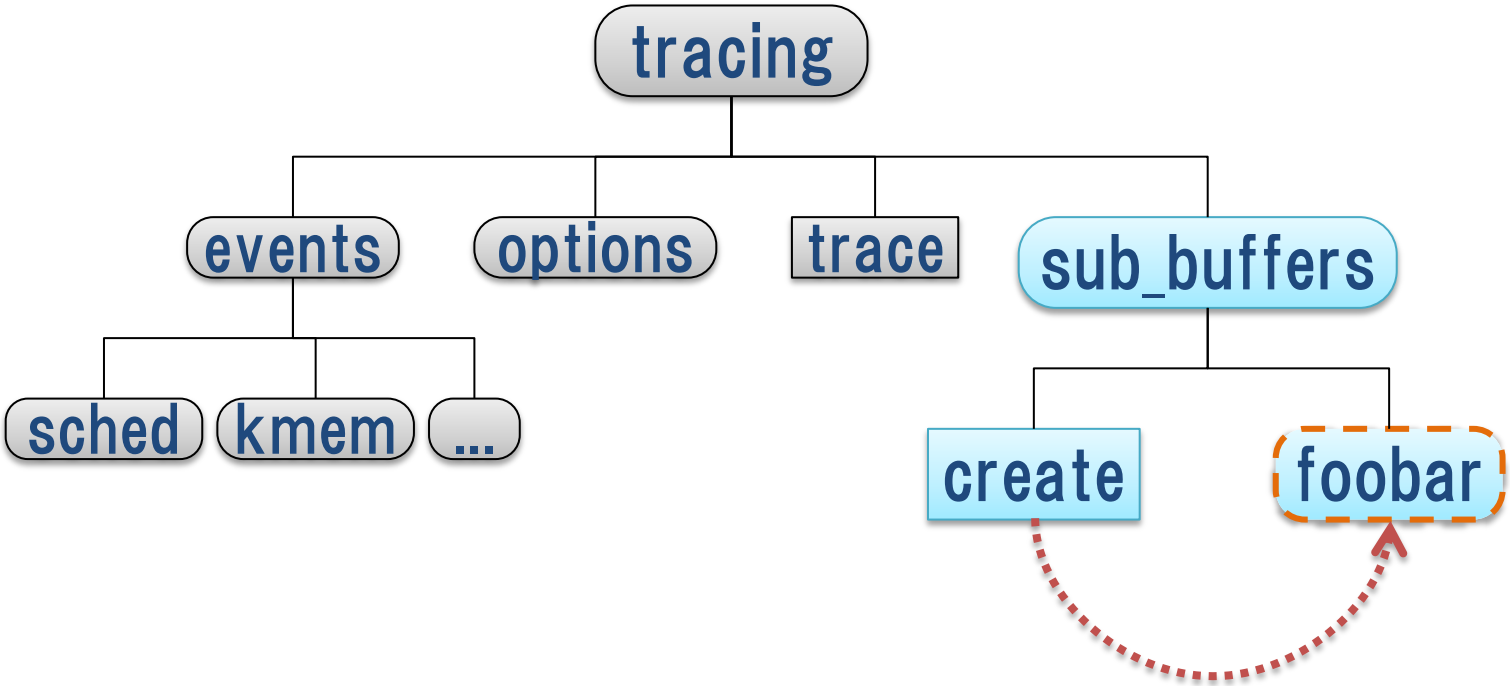
HITACHI
Inspire the Next

- Steven Rostedt told me his idea (thanks!)
  - https://lkml.org/lkml/2011/12/20/212
- create_buffer
  - a (debugfs) file that you echo a name into to create a new (sub-)buffer
  - then a directory with that name will appear

Multiple trace buffer – Interface (Cont.)

HITACHI
Inspire the Next

```
# echo foobar > tracing/sub_buffers/create
```

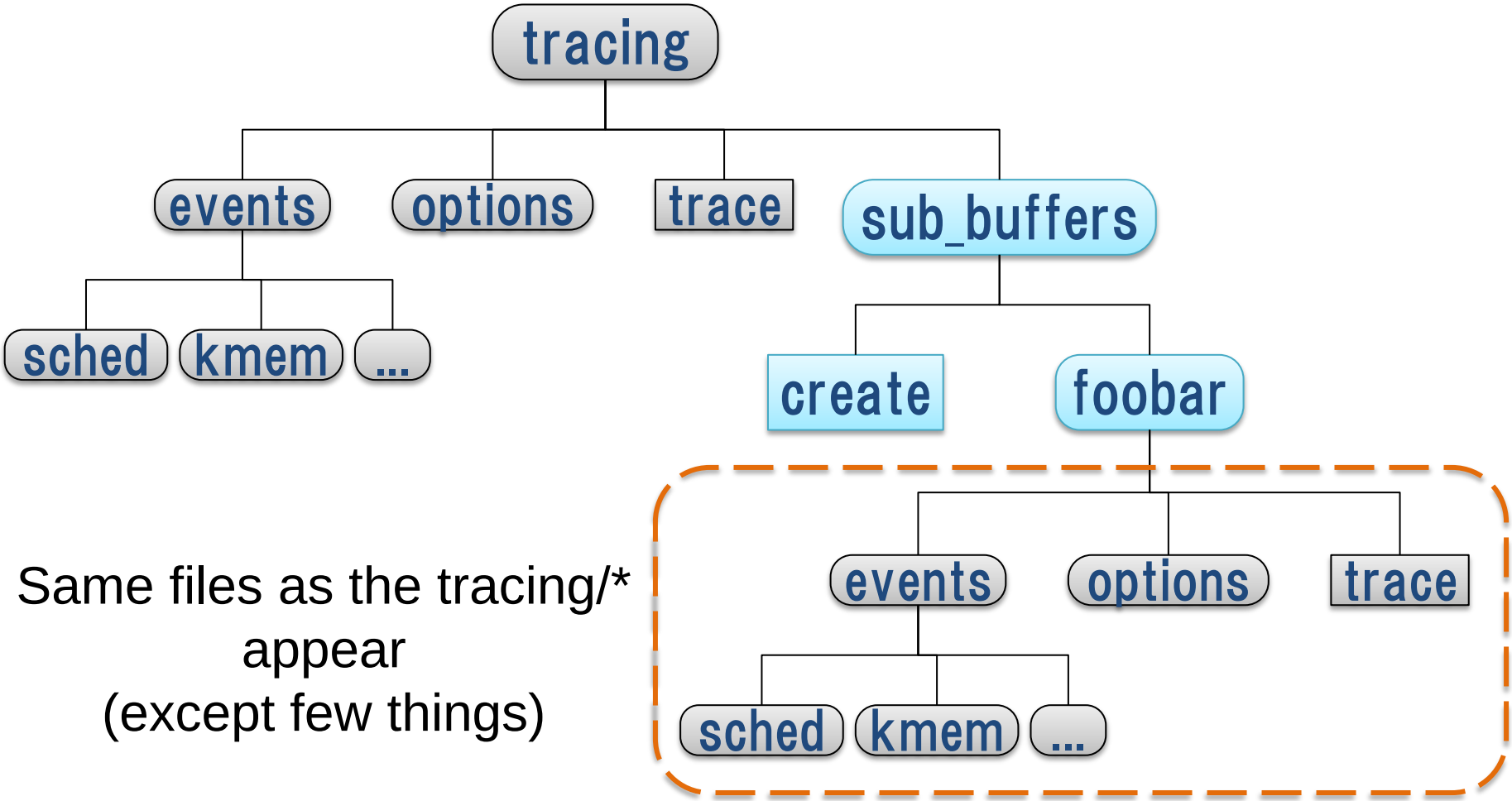


New directory “foobar” appears

Multiple trace buffer – Interface (Cont.)

HITACHI
Inspire the Next

```
# ls tracing/sub_buffers/foobar
```



Same files as the tracing/* appear
(except few things)

Future plans

HITACHI
Inspire the Next

- I will implement proposed features and submit patches to LKML
- I’d like to discuss how exceptions should be treated in snapshot.

Conclusion

HITACHI
Inspire the Next

- Ftrace event tracer is useful as a flight recorder
- For far more requirements, I proposed following features:
  - Snapshot I/F
  - Multiple ring buffer
- These are useful to preserve important events
- I’d like to discuss and solve remaining issues
  - Errors detected only in kernel

Trademarks

HITACHI
Inspire the Next

- Linux is a trademark of Linus Torvalds in the United States, other countries, or both.
- Other company, product, or service names may be trademarks or service marks of others.

Multiple trace buffer – Implementation

HITACHI
Inspire the Next

- LTTng 2.0 have already implemented multiple buffer using “tracepoint”
- Can we implement in event tracer in the same way?
  1. create buffer
  2. add a tracepoint entry corresponding to the buffer
     - It’s necessary for all enabled tracepoints