Ftrace

Debugger, performance measurements, kernel teacher

Frédéric Weisbecker <fweisbec@gmail.com>

Introduction

Origins from the PREEMPT_RT patch.

Self-contained kernel tracing tool/framework

Set of tracers

Set of user toggable/tunable tracepoints

The Ring Buffer

Generic ring buffer for all the kernel

Per cpu write and read

Lockless write and read

Read through ftrace layer or directly splice

Ring Buffer operations

- Write side
  - Overwrite or stop in before head mode
  - Before: Lock and reserve
  - After:
    - Unlock and commit
    - Unlock and discard
- Read side
  - Iterator (local reader)
  - Read (global consumer)

Tracers

- Most basic tracing unit
- Callbacks:
  - Higher level tracing framework operations
  - Lower level fs operations
- Use of tracepoints or ad hoc captures
- Insertion to the ring buffer
- Reserved for tracing requiring low level operations.

Function tracer

Use of a gcc trick (-pg option)

- Static calls to an mcount function
- Probing on entry
- Careful choice of untraced functions

Different modes:

- Static mcount() calls
- Dynamic patching

Function trace

```
# tracer: function
#
#           TASK-PID    CPU#    TIMESTAMP  FUNCTION
#              | |       |          |         |
     soffice.bin-5363  [001]  2744.270302: raise_softirq <-run_local_timers
     soffice.bin-5363  [001]  2744.270303: rcu_pending <-update_process_times
     soffice.bin-5363  [001]  2744.270303: __rcu_pending <-rcu_pending
     soffice.bin-5363  [001]  2744.270304: __rcu_pending <-rcu_pending
     soffice.bin-5363  [001]  2744.270304: printk_tick <-update_process_times
```

Function graph tracer

Extends the function tracer by also hooking on return:

- Live hooking
- Each task has its private stack of function calls

New facilities:

- Draw a call graph
- Measure execution time of functions

Function graph trace

```
# tracer: function_graph
    #
    # CPU  DURATION        FUNCTION CALLS
    # |          |  |                 |  |  |  |

     0)   0.931 us    |    _spin_lock();
     0)                |    page_add_new_anon_rmap() {
     0)                |      __inc_zone_page_state() {
     0)   0.615 us    |        __inc_zone_state();
     0)   1.848 us    |      }
     0)   0.751 us    |      page_evictable();
     0)                |      lru_cache_add_lru() {
     0)   0.691 us    |        __lru_cache_add();
     0)   1.990 us    |      }
     0)   7.231 us    |    }
     0)   0.766 us    |    _spin_unlock();
```

Graph tracer enhancement

- Clients of entry/return hooks: save custom datas in task call graph stack
- Print return values (size? Format?)
- Print parameters values (use of dwarf infos)
- Filter by duration (manage a stack to filter? Userland post-processing?)

Syscalls tracer

- Use existing syscall definition CPP wrapper
  - Build a syscall metadata table
  - Link syscall metadata table to syscall table
- Fast retrieval of number of parameters on fast path
  - One shot registers saving (struct pt_regs)
- Fast retrieval of metadata on slow path
  - Retrieve parameter types and names, link to its value (pretty-printing)

Syscall trace

```
# tracer: syscall
    #
    #          TASK-PID    CPU#    TIMESTAMP  FUNCTION
    #             |          |          |          |                     |
               bash-5606  [000]  2404.628180: sys_dup2(oldfd: a, newfd: 1)
               bash-5606  [000]  2404.628261: sys_dup2 -> 0x1
               bash-5606  [000]  2404.628264: sys_fcntl(fd: a, cmd: 1, arg: 0)
               bash-5606  [000]  2404.628267: sys_fcntl -> 0x1
               bash-5606  [000]  2404.628270: sys_close(fd: a)
               bash-5606  [000]  2404.628273: sys_close -> 0x0
               bash-5606  [000]  2404.628290: sys_rt_sigprocmask(how: 0, set: 0, oset:
    6cf808, sigsetsize: 8)
               bash-5606  [000]  2404.628294: sys_rt_sigprocmask -> 0x0
```

Syscall tracing enhancements

- Build one ftrace event per syscall (ready)
  - Provide filters, toggling, no need of a tracer
- Build a hashlist of complex types:
  - Pointers to a structure: size?
  - Format
  - Link syscalls metadata to this hashlist of complex types. For fast path, have two new fields in the syscall metadata:
    - Bitmap of complex types for this syscall
    - Size of parameter to save from the user pointer (or callback to save in case of very complex parameters).

Some other tracers

- Latency tracing (irqsoff, preemptoff, preemptirqsoff) requires snapshot mode
- Tracers waiting for ftrace events conversion
  - Kmemtrace
  - Blktrace
  - Boot tracer
- Tracers in a middle stage
  - Power, sched, etc...
- Exceptions: mmiotrace...

Ftrace events

- Upper layer of tracepoints
- User-side toggable: the enable/set_event files
  - By event
  - By subsystem
  - All
- Can be filtered using tunable rules

Defining an event

```
TRACE_EVENT(name,
        TP_PROTO(proto),
        TP_ARGS(args),
        TP_STRUCT__entry(define fields),
        TP_fast_assign(assign_fields),
        TP_printk("fmt", fields)
    );
```

Various set of fields

- Static: __field, __array
- Dynamic: __dynamic_array, __string

Drawbacks of ftrace events

CPP is somewhat limited

Need of a specific tracer or dedictated code for (rare) low level or ad-hoc needs.

No histogram / statistical tracing

Ideas for the future

- Ftrace is bad at stat/histogram tracing
- Use perfcounter as a powerful bridge and user interface
- Your ideas!