Learning the Kernel and Finding Performance Problems with KFI

Tim Bird
Sony Electronics, Inc.

Introduction to KFI

- KFI = Kernel Function Instrumentation
  - KFI uses gcc “–finstrument-functions” to insert callout code for every function entry and exit
- Provides for trace collection, including:
  - timing, filtering, triggers for starting and stopping
- Provides tool for trace display and analysis

KFI vs. Profiling

- OPROFILE takes samples of kernel (and application) execution location over time
  - Doesn’t give complete trace coverage
  - Good for statistical analysis of execution paths
- KFI gives exact trace of execution for a single instance
  - Measures all executed routines
  - Good for analyzing special cases (e.g. bootup)

KFI vs. Event tracers

- Event tracer (e.g. LTT or LKST)
  - Measures small number of discreet events
  - Much lower overhead
    - Good for long trace (even continous)
  - Shows interactions between threads and processes – e.g. state transitions
    - Carefully chosen event list
- KFI traces every kernel function
  - High overhead, but very high detail

Finding Performance Problems

- KFI measures function duration time by default!!
  - Trace log has entry time and duration
  - KFI keeps a stack of entered functions which is searches on function exit
    - Normal overhead is low, because function being exited is first item on stack
- Duration is wall time from entry to exit
  - Does NOT take into account (subtract) interrupts and thread switches
- Good for straightline code that does not block or get stuck on locks or semaphores

KFI Modes

- Dynamic configuration
  - Configure and start a trace from user-space
  - Configuration written to /proc/kfi
    - cat kficonf >/proc/kfi
- Static configuration
  - Configuration is compiled into kernel, and started automatically on boot
    - Configuration in kernel/kfistatic.conf
- Results are collected from /proc/kfi_trace in either case

Configuration

- Triggers
  - start, stop
  - function entry, function exit, time
- Filters
  - function duration (mintime, maxtime)
  - interrupt context (noints, onlyints)
  - specific functions only
- Size of trace buffer
  - logentries

Function Names vs. Addresses (in configuration file)

- Can use function name in kfistatic.conf
- Must use function address when writing config to /proc/kfi
  - Provide sym2addr to convert
    - sym2addr kficonf System.map >kficonf.addr
    - cat kficonf.addr >/proc/kfi

Function Names vs. Addresses (in trace output)

- Output from /proc/kfi_trace has addresses
- Can convert to function names with addr2sym:
  - linux/scripts/addr2sym kfi.log -m System.map >kfi.lst

Dynamic usage:

- Compile kernel with KFI enabled
- Boot kernel
- Configure KFI
  - vi kficonf
  - sym2addr kficonf System.map >kficonf.addrmap
  - cat kficonf.addrmap >/proc/kfi
- Prime the trace
  - echo “prime” >/proc/kfi
- Start trigger fires, Trace is collected
- Stop trigger fires, or user stops trace, or buffer is full
  - e.g. echo “stop” >/proc/kfi

Get trace from kernel

- Read trace from kernel, and write it to a file
  - cat /proc/kfi_trace >/tmp/trace.log
- Resolve addresses on output
  - linux/scripts/addr2sym <trace.log -m System.map >trace.lst

Analyze trace

- Look at log directly for individual call events and function durations
- Use ‘kd’ for aggregate function counts and times
- Use ‘kd -c’ to show ascii call graphs

Main Uses

- Learn kernel execution paths
- Measure timing of kernel routines
  - Find problem areas – routines with long execution

Learning Code Paths

- Linux kernel source is very large, and code paths are often very difficult to follow in source

Issue	Problem
Conditional code	Sometimes can't tell if code is compiled in or not.
Jump tables and calls through function pointers	Actual function executed determined at runtime.
Preemption points and interrupts	Cause unexpected execution flow

- KFI can show "gory details" and the real story

Example of hard-to-follow routine: serial8250_init (extract)

```
serial8250_init
|  uart_register_driver
|  serial8250_register_ports
|  |  uart_add_one_port
|  |  |  uart_configure_port
|  |  |  serial8250_config_port
|  |  |  |  serial8250_request_std_resource
|  |  |  tty_register_device
|  |  |  register_console
|  |  |  |  serial8250_console_setup
|  |  |  |  |  uart_set_options
|  |  |  |  |  |  serial8250_set_options
|  |  |  |  |  |  |  uart_get_baud_rate
. . .
```

Bouncing back and forth between serial_core code and driver code. Difficult to follow in source due to jumps through call tables.

Using KD to answer questions

- What functions are called?
- How long do the functions take?
- How many times are the functions called?
- What are the “problem” functions (where most time is spent)?

Example 1 – Find delays in bootup

- Configuration
- Bootup log (unresolved)
- Bootup log (with symbols resolved)
- Summaries of the data using ‘kd’

Bootup Configuration

```
begin
   trigger start entry start_kernel
   trigger stop entry to_userspace
   filter mintime 500
   filter maxtime 0
   filter noints
end
```

Start trace on entry to kernel, and stop just before entering user space

(exec of /sbin/init)

Only keep functions lasting 500 microseconds or longer

(Eliminate shorter ones)

Bootup Log (unresolved)

```
Kernel Instrumentation Run ID 0

Logging started at 0 usec by entry to function 0xc000877c
Logging stopped at 13199877 usec by exit from function 0xc00240dc

Filter Counters:
Execution time filter count = 155574
Total entries filtered = 155574
Entries not found = 3929
Number of entries after filters = 14628

 Entry    Delta     PID        Function                  Caller
-------- -------- -------- ----------------          ------------
       0       -1        0  0xc000877c                0x10008094
       0      958        0  0xc000ddd0                0xc000882c
       0      948        0  0xc000fe14                0xc000de10
       0      938        0  0xc000fd20
      10      919        0  0xc0044c9c
```

All times are in microseconds.

Trace lasted 13 seconds

Large number of short functions eliminated from trace (>90%)

Function addresses are reported

Bootup Log (resolved)

```
Kernel Instrumentation Run ID 0

Logging started at 13762527 usec by entry to function start_kernel
Logging stopped at 23905279 usec by log full

Filter Counters:
Execution time filter count = 754501
Total entries filtered = 754501
Entries not found = 7216
Number of entries after filters = 20000

Entry    Delta     PID        Function                  Caller
-------- -------- -------- ----------------          ------------
       2       -1        0  start_kernel              skpinv+0x190
     392    25317        0  setup_arch                start_kernel+0x3c
     400     1390           init_bootmem              setup_arch+0xec
     437     1352        0           tmem             do_init_bootmem+0x110
     439     1348                                     free_bootmem+0x40
```

Function names and callpoints are resolved

-1 indicates function where the exit was not seen during the trace

Bootup [illegible] d by duration

Show “top” 15 functions

Lots of time in timer_interrupt (8 sec)

```
$ kd -n 15 ebony-start.lst
Function                          Count  Time          ge  Local
--------------------------------- -----  ----- -------- --------
timer_interrupt                    3252  7898788     2428  7898788
do_softirq                         2870  3804507     1325    14035
__do_softirq                       2863  3790472     1323  3790472
release_console_sem                  36  1817877    50496    53868
call_console_drivers                 36  1764009    49000      264
_call_console_drivers                44  1763745    40085      144
__call_console_drivers               44  1763601    40081        1
serial8250_console_write             44  1763450    40078  1763450
create_dir                          402  1579736     3929    80801
tty_register_device                 339  1219851     3598    15384
class_device_register               290  1182139     4076    58236
class_simple_device_add             327  1179817     3608   -97058
class_device_add                    268  1085643     4050    64904
printk                               35  1082977    30942      122
vprintk                              35  1082855    30938     4385
```

Lots of time processing softirqs (4 sec)

Bootup funct…ion

Functions are nested – KFI shows wall-time duration of each

Can detect nesting when functions have similar times and call counts.

			Average	Local
[illegible]	[illegible]	[illegible]788	2428	7898788
do_softirq	2870	38[illegible]507	1325	14035
__do_softirq	2863	379[illegible]472	1323	3790472
release_console_sem	36	1817877	50496	53868
call_console_drivers	36	1764009	49000	264
_call_console_drivers	44	1763745	40085	144
__call_console_drivers	44	1763601	40081	151
serial8250_console_write	44	1763450	40078	1763450
create_dir	402	1579736	3929	80801
tty_register_device	339	1219851	3598	15384
class_device_register	290	1182139	4076	58236
class_simple_device_add	327	1179817	3608	-97058
class_device_add	268	1085643	4050	64904
printk	35	1082977	30942	122
vprintk	35	1082855	30938	4385

Large local time indicates bottom of call chain – where time was spent.

Bootup functions sorted by duration

```
$ kd -n 15 ebony-start.lst
Function                              Count Time     Average  Local
----------------------------------- ----- -------- -------- --------
timer_interrupt                      3252  7898788     2428  7898788
do_softirq                           2870  3804507     1325    14035
__do_softirq                         2863  3790472     1323  3790472
release_console_sem                    36  1817877    50496    53868
call_console_drivers                   36  1764009    49000      264
_call_console_drivers                  44  1763745    40085      144
__call_console_drivers                 44  1763601    40081      151
serial8250_console_write               44  1763450    40078  1763450
create_dir                            402  1579736      392    80801
tty_register_device                   339  1219851              15384
class_device_register                 290  1182139       76    58236
class_simple_device_add               327  1179          608   -97058
class_device_add                      268                4050    64904
printk                                                          122
vprintk                                                        4385
```

In this case, there were sub-routines called by serial8260_console_write, but they were eliminated by filtering

Functions Sorted by Local Time

Sort by local time with “-s l”

```
$ kd -n 15 -s l ebony-start.lst
Function                          Count Time     Average  Local
--------------------------------- ----- -------- -------- --------
timer_interrupt                    3252  7898788     2428  7898788
__do_softirq                       2863  3790472     1323  3790472
serial8250_console_write             44  1763450    40078  1763450
__switch_to                          26   364138    14005   364138
sysfs_lookup                         57   187251     3285   187251
__might_sleep                        13   107389     8260   107389
create_dir                          402  15797[illegible]     3929    80801
kmem_cache_alloc                     51  [illegible]1     3511    70421
preempt_schedule                    [illegible]6924     3186    66375
class_device_add                    [illegible]1085643     4050    64904
class_device_register               [illegible]0  1182139     4076    58236
kobject_add                         213   873241     4099    56017
                                    [illegible]105991     3654    55613
                                    [illegible]1817877    50496    53868
                                     47227    11806    47227
```

Shows routines where time was most spent, and either no sub-functions were called, or the sub-functions were not included in the trace.

Functio

Specify fields to show. ‘m’=max subroutine (child sub-routine where most time was spent)

```
$ kd -n 15 -f Fctlmn ebony-start.lst
```

Function	Count	Time	Local	Max-sub	Ms count
timer_interrupt	3252	7898788	7898788	update_process_times	1
do_softirq	2870	3804507	14035	__do_softirq	2863
__do_softirq	2863	3790472	3790472	sub_preempt_count	2881
release_console_sem	36	1817877	53868	call_console_drivers	36
call_console_drivers	36	1764009	264	_call_console_drivers	44
_call_console_drivers	44	1763745	144	__call_console_drivers	44
__call_console_drivers	44	1763601	151	serial8250_console_write	44
serial8250_console_write	44	1763450	1763450		0
create_dir	402	1579736	80801	sysfs_create_dir	199
tty_register_device	339	1219851	15384	class_simple_device_add	326
class_device_register	290	1182139	58236	class_device_add	268
class_simple_device_add	327	1179817	-97058	class_device_register	290
class_device_add	268	1085643	64904	kobject_add	209
printk	35	1082977	122	vprintk	35
vprintk	35	1082855	4385	release_console_sem	35

Sub-system init functions

```
$ kd ebony-start.lst | grep "_init[^a-z_]"
console_init                          1   740406   740406       25
serial8250_console_init               1   740381   740381        7
mem_init                              1   145839   145839     5113
vfs_caches_init                       1    75011    75011      394
mnt_init                              1    74025    74025      362
usb_init                              1    46636    46636     1211
usb_hub_init                          1    44810    44810      384
netlink_proto_init                    1    44141    44141      539
pcibios_init                          1    36910    36910      477
kobject_init                         11    35315     3210     8097
kref_init                             1    16821    16821    16821
paging_init                           1    14019    14019        3
free_area_init                        1    14016    14016        5
sysctl_init                           1    13017    13017        7
ocp_driver_init                       1    12577    12577     3020
param_sysfs_init                      1     6410     6410      409
kmem_cache_init                       1     5084     5084     5084
```

Example 2 – Tracing do_fork

- Configuration
- Summary data using ‘kd’
- Trace graph using ‘kd –c’
  - Interlace mode vs. non-interlace mode

Trace Configuration for do_fork

```
new
begin
   trigger start entry do_fork
   trigger stop exit do_fork
   filter mintime 0
   filter maxtime 0
   filter noints
end
```

Trace a single routine from entry to exit

No filtering by time

do_fork Functions Sorted by Duration

```
$ kd -n 15  ebony-do_fork.lst
Function                              Count Time     Average  Local
----------------------------------- ----- -------- -------- --------
do_fork                                 1    11100    11100       57
preempt_schedule                       26     9086      349       66
wake_up_new_task                        1     9079     9079       30
schedule                                2     9017     4508       39
__switch_to                             2     8961     4480     8961
__nfs_revalidate_inode                  3     5827     1942       54
link_path_walk                          3     5788     1929      -81
nfs_proc_getattr                        3     5687     1895       21
rpc_call_sync                           3     5666     1888       52
rpc_execute                             3     5097     1699       19
__rpc_execute                           3     5078     1692      173
open_exec                               2     4250     2125       19
path_lookup                             2     3929     1964       11
call_transmit                           3     3831     1277       31
do_lookup                               5     3739      747       23
```

do_fo[illegible]d by Local Time

Anomaly caused by context switching – should ignore __switch_to

```
$ kd -n 15 -s l ebony-do_fork.lst
Function                          Cou[illegible]me      Average  Local
--------------------------------- ----- -------- -------- --------
__switch_to                           2     8961     4480     8961
timer_interrupt                      10     1537      153     1537
dev_queue_xmit                        3     1474      491     1474
__do_softirq                         12     1092       91     1092
xprt_transmit                         3     3510     1170      590
sub_preempt_count                   237      449        1      449
kmem_cache_alloc                     35      433       12      372
do_IRQ                                1      369      369      369
neigh_resolve_output                  3     1849      616      367
copy_mm                               1     1702     1702      352
xprt_release                          3      506      168      268
lock_kernel                          39      228        5      228
copy_user_page                        6      216       36      216
copy_pte_range                       17      475       27      209
ip_finish_output                      3     2050      683      193
```

Functions Sorted by Call Count

```
$ kd -n 15 -s c ebony-do_fork.lst
```

Function	Count	Time	Average	Local
sub_preempt_count	237	449	1	449
__might_sleep	78	149	1	149
unlock_kernel	39	114	2	92
lock_kernel	39	228	5	228
kmem_cache_alloc	35	433	12	372
__mod_page_state	28	56	2	56
preempt_schedule	26	9086	349	66
pte_alloc_map	23	212	9	62
nfs_attribute_timeout	22	163	7	163
anon_vma_link	17	82	4	60
__vma_link_rb	17	225	13	65
copy_pud_range	17	611	35	66
cond_resched_lock	17	33	1	33
copy_page_range	17	681	40	70
copy_pmd_range	17	545	32	70

Call Trace for do_fork

```
$kd -c -l do_fork.lst
 Entry      Duration   Local      Pid   Trace
---------- ---------- ---------- ----- -------
         3      11100         57    19 do_fork
         6          5          5    19 |  alloc_pidmap
        14       1959         70    19 |  copy_process
        17         80         16    19 |  |  dup_task_struct
        19          6          4    19 |  |  |  prepare_to_copy
        21          2          2    19 |  |  |  |  sub_preempt_count
        28          6          4    19 |  |  |  kmem_cache_alloc
        30          2          2    19 |  |  |  |  __might_sleep
        37         52          4    19 |  |  |  __get_free_pages
        39         48         14    19 |  |  |  |  __alloc_pages
        41          2          2    19 |  |  |  |  |  __might_sleep
        46          2          2    19 |  |  |  |  |  zone_watermark_ok
        51         29         14    19 |  |  |  |  |  buffered_rmqueue
        54          3          3    19 |  |  |  |  |  |  __rmqueue
        59          2          2    19 |  |  |  |  |  |  sub_preempt_count
        63          2          2    19 |  |  |  |  |  |  bad_range
        68          2          2    19 |  |  |  |  |  |  __mod_page_state
        72          6          4    19 |  |  |  |  |  |  prep_new_page
        74          2          2    19 |  |  |  |  |  |  |  set_page_refs
        83          1          1    19 |  |  |  |  |  zone_statistics
       102         17          6    19 |  |  do_posix_clock_monotonic_gettime
```

or do_fork – with Interrupt

"Interlace" mode shows call graphs for interrupting code nested inside other graphs

```
      2286           1           2    41 {  {  {  {  {  {  set_page_refs
      2294           2           2    41 {  {  {  {  zone_statistics
      2300         130         130    41 {  {  {  copy_user_page
---------- %%%% start --------------
      2312         106         106    41 timer_interrupt
---------- %%%% end ----------------
---------- ]]]] start --------------
      2383          29           4    41 do_softirq
      2385          25          25    41 ]  __do_softirq
---------- ]]]] end ----------------
---------- **** start --------------
      2414           2           2    41 sub_preempt_count
---------- **** end ----------------
      2432           2           2    41 {  {  {  page_remove_rmap
      2436          16          16    41 {  {  {  update_mmu_cache
      2454           5           3    41 {  {  {  lru_cache_add_active
      2455           2           2    41 {  {  {  {  sub_preempt_count
```

Uses different punctuation to show different threads of execution

Call Trace for do_fork – with Thread Switch

```
1976       9079          30    19 |  wake_up_new_task
1978          3           3    19 |  |  task_rq_lock
1984          2           2    19 |  |  effective_prio
1990          1           1    19 |  |  sub_preempt_count
1994       9043          25    19 |  |  preempt_schedule
1996       9018          52    19 |  |  |  schedule
1998          2           2    19 |  |  |  |  profile_hit
2002          2           2    19 |  |  |  |  sched_clock
2011       8962        8962    19 |  |  |  |  __switch_to
2015         11           4    41 schedule_tail
2017          7           6    41 $  finish_task_switch
2020          1           1    41 $  $  sub_preempt_count
2033        173          11    41 do_page_fault
2035          2           2    41 /  __might_sleep
2040          2           2    41 /  find_vma
2045        158          10    41 /  handle_mm_fault
2047          2           2    41 /  /  __mod_page_state
```

Different punctuation shows different thread of execution

Call Trace for do_fork – with Thread Switch

```
1976       9079         30    19 |  wake_up_new_task
1978          3          3    19 |  |  task_rq_lock
1984          2          2    19 |  |  effective_prio
1990          1          1    19 |  |  sub_preempt_count
1994       9043         25    19 |  |  preempt_schedule
1996       9018         52    19 |  |  |  schedule
1998          2          2    19 |  |  |  |  profile_hit
2002          2          2    19 |  |  |  |  sched_clock
2011       8962       8962    19 |  |  |  |  __switch_to
2015         11          4    41 schedule_tail
2017          7          6    41 $  finish_task_switch
2020          1          1    41 $  $  sub_preempt_count
2033        173         11    41 do_page_fault
2035          2          2    41 /  __might_sleep
2040          2          2    41 /  find_vma
2045        158         10    41 /  handle_mm_fault
2047          2          2    41 /  /  __mod_page_state
```

Call Trace for do_fork – another Thread Switch

```
2160          16          16    41 /  /  /  update_mmu_cache
2178           7           5    41 /  /  /  lru_cache_add_active
2181           2           2    41 /  /  /  /  sub_preempt_count
2187           6           4    41 /  /  /  page_add_anon_rmap
2189           2           2    41 /  /  /  /  __mod_page_state
2196           2           2    41 /  /  /  sub_preempt_count
2211         267           8    41 do_page_fault
2213           2           2    41 {  __might_sleep
2217           2           2    41 {  find_vma
2221         255           9    41 {  handle_mm_fault
2223           2           2    41 {  {  __mod_page_state
2227           2           2    41 {  {  pte_alloc_map
2231         242          24    41 {  {  do_wp_page
2233           2           2    41 {  {  {  can_share_swap_page
2237          10           6    41 {  {  {  unlock_page
2239           2           2    41 {  {  {  {  page_waitqueue
2243           2           2    41 {  {  {  {  __wake_up_bit
```

Example 3 – Tracing serial8250_init

- Configuration
- Log
- Examination of call trace

Trace Configuration for serial8250_init

```
new
begin
   trigger start entry serial8250_init
   trigger stop exit serial8250_init
   filter mintime 0
   filter maxtime 0
end
```

serial8250_init Log (resolved)

```
Kernel Instrumentation Run ID 0

Logging started at 1285942 usec by entry to function     ial8250_init
Logging stopped at 1796103 usec by log full

Filter Counters:
Total entries filtered = 0
Entries not found = 14
Number of entries after filters = 20000

 Entry    Delta     PID        Function                  Caller
-------- -------- -------- -------------------- ------------
      11       -1        1  serial8250_init           do_initcalls+0x90
      24     2245        1  printk                    serial8250_init+0x40
      36     2214        1  vprintk                   printk+0x2c
      49      123        1  vscnprintf                vprintk+0x64
```

Problem: could not finish routine because log filled up

serial8250_init Trace

- See omap-serial_init.trace
- Here are highlights:
  - at 11 usec - ‘-1’ duration means exit was not seen during trace.
    - This is bad, I expected routine to complete quickly. Something else happened during init
  - at 4096 usec - interrupt occurred
    - 640 usec for hard portion, 192 usec for soft portion
    - at 5702 usec - back to main thread (interrupt complete)

serial8250_init Trace (cont.)

- at 10161 usec - call to create_dir
  - at 10190 usec - another call to a different create_dir
  - Be careful of namespace (all functions are in trace namespace, even if they are static)
  - This means timing data for different routines may be combined, even if they are different
    - (ie kd is somewhat stupid)
- at 51580 usec - finally see some interaction with the serial port
  - It takes more than 50 milliseconds just to do device driver setup unrelated to the serial port

serial8250_init Trace (cont. 2)

- at 81457 usec - here's another much longer timer interrupt
  - 4 milliseconds to perform interrupt
  - hit a timer tick where more processing was performed (updating system time)
- at 97724 usec - call to 'call_console_drivers' from
  - from printk code
  - results from wakeup via console semaphore
  - at 97279 usec - serial8250_console_write
    - starting to output all the data from the print log buffer

serial8250_init Trace (cont. 3)

- Whole rest of trace is outputting printk data
  - data was pent up waiting for console to be initialized
- at 97719 usec - serial_out followed by 4 serial_in calls
  - basic pattern for outputting a single char is to write char and wait for buffer to clear
  - 11 usec per operation = 55 usec per character
- Polling results in long delays. Why isn't buffering and interrupts being used here?

Issues With Timing Data

- "Local time" is calculated by subtracting wall time of children from wall time of function
  - Careful – filtering removes functions – local time can be inaccurate
  - Also a problem with asynchronous routines
- Warning – system effectively ignores calls that don't return (e.g. schedule, switch_to)

Resources

- Wiki pages:
  - http://tree.celinuxforum.org/CelfPubWiki/KernelFunctionInstrumentation
  - http://tree.celinuxforum.org/CelfPutWiki/PatchArchive
- Mailing list:
  - celinux-dev@tree.celinuxforum.org

Questions and Answers