Embedded Linux kernel解析ノウハウ
Linux kernel traceの詳細 part2

Yoichi Yuasa

OSAKA NDS Embedded Linux Cross Forum #5

自己紹介

- 湯浅陽一
- 1999年よりLinux kernel開発に参加
- MIPSアーキテクチャのいくつかのCPUへLinux kernelを移植

Linux kernel tracerとは (前回おさらい)

- Linux kernel内蔵の内部追跡システム
- 用途に応じて複数追跡方法から選択可能
- 時間計測が可能
- システム動作中にON/OFF可能
- メモリ上のリングバッファに記録
- フィルタ機能があり選択記録が可能
- インタフェースはメモリファイルシステム上のファイル
- ARMアーキテクチャなど組込みでも利用可能
- kernelのデバッグ、検証、分析やチューニングに利用

Tracerの種類

- Kernel Function Tracer(前回説明)
  - Kernel Function Graph Tracer(前回説明)
- Interrupts-off Latency Tracer
- Preemption-off Latency Tracer
- Scheduling Latency Tracer
- Trace Events(デフォルトで組込まれている)
  - Trace gpio events

Kernel configuration

Kernel hacking --->

[*] Tracers --->

    [*] Kernel Function Tracer

    [*]   Kernel Function Graph Tracer

    [*] Interrupts-off Latency Tracer

    [*] Preemption-off Latency Tracer

    [ ] Scheduling Latency Tracer

    [ ] Trace gpio events

Interrupts-off Latency Tracer

- trace内容
  - 実行プロセス(コマンド名)
  - プロセスID
  - 実行CPU番号
  - 割込み禁止/許可(d/・)
  - スケジュール要求(N/n/p/・)
  - ハードIRQ/ソフトIRQ(H/h/s/・)
  - プリエンプト深さ(disableカウント)
  - 割込みオフ期間
  - 遅延レベル($/@/*/#/!/+/)
  - 処理区間関数名

Interrupts-off Latency Trace表示

```
# tracer: irqsoff
#
# irqsoff latency trace v1.1.5 on 4.9.30
# --------------------------------------------------------------------
# latency: 116 us, #120/120, CPU#3 | (M:preempt VP:0, KP:0, SP:0 HP:0 #P:4)
#    -----------------
#    | task: systemd-1 (uid:0 nice:0 policy:0 rt_prio:0)
#    -----------------
#  => started at: proc_cgroup_show
#  => ended at:   proc_cgroup_show
#
#
#                  _------=> CPU#
#                 / _-----=> irqs-off
#                | / _----=> need-resched
#                || / _---=> hardirq/softirq
#                ||| / _--=> preempt-depth
#                |||| /     delay
#  cmd     pid   ||||| time  |   caller
#     \   /      |||||  \    |   /
 systemd-1       3d...    0us : _raw_spin_lock_irq <-proc_cgroup_show
 systemd-1       3d..1    3us : preempt_count_add <-_raw_spin_lock_irq
```

Interrupts-off Latency Trace表示

```
 systemd-1       3d..3  111us : _raw_spin_unlock_irqrestore < kernfs_path_from_node
 systemd-1       3d..3  112us : preempt_count_sub <-_raw_spin_unlock_irqrestore
 systemd-1       3d..2  113us : seq_puts <-proc_cgroup_show
 systemd-1       3d..2  113us : seq_putc <-proc_cgroup_show
 systemd-1       3d..2  115us : _raw_spin_unlock_irq <-proc_cgroup_show
 systemd-1       3d..1  116us : _raw_spin_unlock_irq <-proc_cgroup_show
 systemd-1       3d..1  118us : trace_hardirqs_on <-proc_cgroup_show
 systemd-1       3d..1  121us : <stack trace>
=> proc_cgroup_show
=> proc_single_show
=> seq_read
=> __vfs_read
=> vfs_read
=> SyS_read
=> el0_svc_naked
```

Preemption-off Latency Tracer

- trace内容(Interrupts-off Latency Tracerと同一)
  - 実行プロセス(コマンド名)
  - プロセスID
  - 実行CPU番号
  - 割込み禁止/許可(d/・)
  - スケジュール要求(N/n/p/・)
  - ハードIRQ/ソフトIRQ(H/h/s/・)
  - プリエンプト深さ(disableカウント)
  - preempt禁止期間
  - 遅延レベル($/@/*/#/!/+/)
  - 処理区間関数名

Preemption-off Latency Trace表示

```
# tracer: preemptoff
#
# preemptoff latency trace v1.1.5 on 4.9.30
# --------------------------------------------------------------------
# latency: 723 us, #1590/1590, CPU#0 | (M:preempt VP:0, KP:0, SP:0 HP:0 #P:4)
#    -----------------
#    | task: cat-2070 (uid:0 nice:0 policy:0 rt_prio:0)
#    -----------------
#  => started at: unmap_page_range
#  => ended at:   unmap_page_range
#
#
#                  _------=> CPU#
#                 / _-----=> irqs-off
#                | / _----=> need-resched
#                || / _---=> hardirq/softirq
#                ||| / _--=> preempt-depth
#                |||| /     delay
#  cmd     pid   ||||| time  |   caller
#     \   /      |||||  \    |   /
     cat-2070    0...1    0us : _raw_spin_lock <-unmap_page_range
     cat-2070    0...2    1us : vm_normal_page <-unmap_page_range
     cat-2070    0...2    1us : mark_page_accessed <-unmap_page_range
```

Preemption-off Latency Trace表示

```
 cat-2070    0...2  718us : unlock_page_memcg <-page_remove_rmap
 cat-2070    0...2  719us : __rcu_read_unlock <-unlock_page_memcg
 cat-2070    0...2  720us : __tlb_remove_page_size <-unmap_page_range
 cat-2070    0...2  720us : __tlb_remove_page_size.part.9 <-__tlb_remove_page_size
 cat-2070    0...2  722us : _raw_spin_unlock <-unmap_page_range
 Cat-2070    0...2  722us : preempt_count_sub <-_raw_spin_unlock
 cat-2070    0...1  723us : _raw_spin_unlock <-unmap_page_range
 cat-2070    0...1  724us : trace_preempt_on <-unmap_page_range
 cat-2070    0...1  727us : <stack trace>
=> _raw_spin_unlock
=> unmap_page_range
=> unmap_single_vma
=> unmap_vmas
=> exit_mmap
=> mmput
=> do_exit
=> do_group_exit
=> __wake_up_parent
=> el0_svc_naked
```

Preempt IRQs-off Latency Tracer

- Interrupts-off Latency Tracerと
  Preemption-off Latency Tracerを同時に行う
- 表示形式は同一

Interrupts-off Latency Tracerの仕組み

- local_irq_enable/disable/save/restore()にtrace関数を挿入
- trace関数で割込み禁止開始時間と終了時間および呼び出し元アドレスを保存
- spin_lock/rwlock/seqlockなど割込み禁止を含むkernelインタフェースを利用していれば上記local_irq_*を利用しているため自動的に測定
- local_irq_*より低位の関数を利用すると測定されない
  - raw_local_irq_enable/disable/save/restore()

Preemption-off Latency Tracerの仕組み

- スケジューラ内にtrace関数を挿入
  - preempt_count_add/sub(), preempt_schedule_common/notrace()
- trace関数でpreempt禁止開始時間と終了時間および呼び出し元アドレスを保存
- premmpt_enable/disable()やlocal_bh_enable/disable()などのkernelインタフェースを利用していてば上記処理を利用しているため自動的に測定

Latency Tracerはどんな時に利用する?

- アプリケーションパフォーマンスの初期調査
  - Kernel内に原因があってパフォーマンスが低下していないか?
    - 操作しているデバイスに長い割込み禁止区間が無いか?
    - 長いスケジュール禁止区間が無いか?
- 短周期処理で対応可能な範囲の目安調査
  - 頻繁に発生する割込みの割込みハンドラによる処理
  - アプリケーションによるデバイスなどのポーリング処理

trace準備

- Debug Filesystem(debugfs)のmount
  - /etc/fstabに記述する場合

    debugfs /sys/kernel/debug debugfs defaults 0 0
  - mount -t debugfs nodev /sys/kernel/debug
  - 多くの場合はmount処理は自動で処理済み
- Debugfsでmountした /sys/kernel/debug/tracing以下のファイルがインタフェース

Tracerインタフェース(1)

```
~# cd /sys/kernel/debug/
/sys/kernel/debug# ls
asoc                gpio          pm_qos              suspend_stats
bdi                 hid           pvr                 tee
clk                 kprobes       pwm                 tracing
debug_enabled       memblock      ras                 usb
dma_buf             mmc0          regmap              wakeup_sources
dri                 mmc1          regulator           xf-dbg-trace
dynamic_debug       mmc2          sched_features
extfrag             opp           sleep_time
fault_around_bytes  pinctrl       split_huge_pages
```

Tracerインタフェース(2)

```
/sys/kernel/debug# cd tracing
/sys/kernel/debug/tracing# ls
README                      free_buffer         set_event          trace_marker
available_events            instances           set_event_pid      trace_options
available_filter_functions  kprobe_events       set_ftrace_filter  trace_pipe
available_tracers           kprobe_profile      set_ftrace_notrace tracing_cpumask
buffer_size_kb              max_graph_depth     set_ftrace_pid     tracing_max_latency
buffer_total_size_kb        options             set_graph_function tracing_on
current_tracer              per_cpu             set_graph_notrace  tracing_thresh
dyn_ftrace_total_info       printk_formats      snapshot           uprobe_events
enabled_functions           saved_cmdlines      trace              uprobe_profile
events                      saved_cmdlines_size trace_clock
```

tracing_on

- トレース機能のON/OFF

  # cat tracing_on

  1　　ON状態

  # echo 0 > tracing_on　OFFに設定

  # cat tracing_on

  0　　OFF状態

- 初期はONなのでまずOFFにしておく

available_tracers

- 利用できるTracerの表示

  # cat available_tracers

  function_graph preemptirqsoff preemptoff irqsoff function nop

current_tracer

- 利用するTracerの設定/表示

  # cat current_tracer

  nop　　デフォルトはnop

  # echo irqsoff > current_tracer

  # cat current_tracer

  irqsoff

  # echo preemptoff > current_tracer

  preemptoff

trace

- Trace結果出力

```
/sys/kernel/debug/tracing# cat trace
# tracer: irqsoff
#
```

  - まだOFFなのでヘッダーのみ

trace実行

- 実行方法

  # echo 1 > tracing_on

  # run_test

  # echo 0 > tracing_on

  # cat trace

- Latency TracerはFunction Tracerと違って長時間実行するのでtrace実行期間を短くする必要性はない

Latency Tracer実行時のポイント

- 負荷を高くするだけではなく多くの処理を試す
  - どこに長い禁止期間があるかは分からない
  - コードから長い禁止時間を推測することは容易ではない
  - 測定時にコードカバレッジを上げることが重要
- 長時間測定行う
  - 同じ処理でも徐々に禁止期間が伸びていく処理がある
    - リスト処理

trace実行例

- Preemption IRQs-off Latency Tracer
- Root filesystemはNFS
- ファイルシステムへのアクセスで負荷を発生
- trace結果は6704エントリ

```
/sys/kernel/debug/tracing# cat trace
# tracer: preemptirqsoff
# preemptirqsoff latency trace v1.1.5 on 4.9.30
# --------------------------------------------------------------------
# latency: 3917 us, #6705/6705, CPU#3 | (M:preempt VP:0, KP:0, SP:0 HP:0 #P:4)
#    -----------------
#    | task: kswapd0-531 (uid:0 nice:0 policy:0 rt_prio:0)
#    -----------------
#  => started at: __list_lru_walk_one.isra.2
#  => ended at:   __list_lru_walk_one.isra.2
#                  _------=> CPU#
#                 / _-----=> irqs-off
#                | / _----=> need-resched
#                || / _---=> hardirq/softirq
#                ||| / _--=> preempt-depth
#                |||| /     delay
#  cmd     pid   ||||| time  |   caller
#     \   /      |||||  \    |   /
 kswapd0-531     3...1    0us : _raw_spin_lock <-__list_lru_walk_one.isra.2
 kswapd0-531     3...2    1us : dentry_lru_isolate <-__list_lru_walk_one.isra.2
 kswapd0-531     3...3    4us : _raw_spin_unlock <-dentry_lru_isolate
```

```
kswapd0-531     3...2 3905us : _raw_spin_trylock <-dentry_lru_isolate
kswapd0-531     3...2 3905us : preempt_count_add <-_raw_spin_trylock
kswapd0-531     3...3 3906us : preempt_count_add <-dentry_lru_isolate
kswapd0-531     3...4 3906us : preempt_count_sub <-dentry_lru_isolate
kswapd0-531     3...3 3907us : list_lru_isolate <-dentry_lru_isolate
kswapd0-531     3...3 3908us : _raw_spin_unlock <-dentry_lru_isolate
kswapd0-531     3...3 3908us : preempt_count_sub <-_raw_spin_unlock
kswapd0-531     3...2 3909us : dentry_lru_isolate <-__list_lru_walk_one.isra.2
kswapd0-531     3...2 3909us : _raw_spin_trylock <-dentry_lru_isolate
kswapd0-531     3...2 3910us : preempt_count_add <-_raw_spin_trylock
kswapd0-531     3...3 3910us : _raw_spin_unlock <-dentry_lru_isolate
kswapd0-531     3...3 3911us : preempt_count_sub <-_raw_spin_unlock
kswapd0-531     3...2 3912us : dentry_lru_isolate <-__list_lru_walk_one.isra.2
kswapd0-531     3...2 3912us : _raw_spin_trylock <-dentry_lru_isolate
kswapd0-531     3...2 3913us : preempt_count_add <-_raw_spin_trylock
kswapd0-531     3...3 3914us : _raw_spin_unlock <-dentry_lru_isolate
kswapd0-531     3...3 3914us : preempt_count_sub <-_raw_spin_unlock
kswapd0-531     3...2 3915us : _raw_spin_unlock <-__list_lru_walk_one.isra.2
kswapd0-531     3...2 3915us : preempt_count_sub <-_raw_spin_unlock
kswapd0-531     3...1 3917us : _raw_spin_unlock <-__list_lru_walk_one.isra.2
kswapd0-531     3...1 3918us : trace_preempt_on <-__list_lru_walk_one.isra.2
kswapd0-531     3...1 3922us : <stack trace>
```

Tracerオプション

- /sys/kernel/debug/tracing/options以下にファイルがある
- 1を書き込むとオプションがON、0を書き込むとオフになる

Tracerオプション

```
/sys/kernel/debug/tracing/options# ls
annotate            funcgraph-duration  latency-format      sym-offset
bin                 funcgraph-irqs      markers             sym-userobj
block               funcgraph-overhead  overwrite           test_nop_accept
context-info        funcgraph-overrun   print-parent        test_nop_refuse
disable_on_free     funcgraph-proc      printk-msg-only     trace_printk
display-graph       funcgraph-tail      raw                 userstacktrace
event-fork          function-trace      record-cmd          verbose
func_stack_trace    graph-time          sleep-time
funcgraph-abstime   hex                 stacktrace
funcgraph-cpu       irq-info            sym-addr
/sys/kernel/debug/tracing/options# cat function-trace
1
/sys/kernel/debug/tracing/options# echo 0 > function-trace
/sys/kernel/debug/tracing/options# cat function-trace
0
```

参考

- Documentation/trace/ftrace.txt
- Documentation/trace/ftrace-design.txt
- kernel/trace/trace_irqsoff.c
- kernel/sched/core.c
- include/linux/irqflags.h
- include/linux/preempt.h