perf

(Linux profiling with performance counter)

Concurrent Programming

Scalable Computing Systems Laboratory
Hanyang University

HYU 한양대학교 HANYANG UNIVERSITY

Introduction

- What is Perf?
- Installing Perf
- Perf Commands
- Example

What is Perf?

- Performance analyzing tool in Linux
  - available from Linux kernel version 2.6.31
- Capable of statistical profiling of the entire system
  - (both kernel and userland code)
- Available to measure many types of event
  - (Hardware events, Software events, …)

Perf Commands

- perf list

  Show the list of available events to be measured with perf

- perf stat

  Obtain aggregated event counts

- perf record

  Sampling events on per-thread, per-process, per-cpu basis and generate the output file

- perf report

  Analyze the output file generated from *perf record*

- perf annotate

  Analyze the output file generated from *perf record* in the instruction level

- more Commands, but not today

Perf Commands – perf list

- Show the list of events that can be measured by perf

```
$ perf list
```

```
jongbin@multicore-96:~/TA/Multicore/lab02$ perf list

List of pre-defined events (to be used in -e):

  branch-instructions OR branches                    [Hardware event]
  branch-misses                                      [Hardware event]
  bus-cycles                                         [Hardware event]
  cache-misses                                       [Hardware event]
  cache-references                                   [Hardware event]
  cpu-cycles OR cycles                               [Hardware event]
  instructions                                       [Hardware event]
  ref-cycles                                         [Hardware event]

  alignment-faults                                   [Software event]
  bpf-output                                         [Software event]
  context-switches OR cs                             [Software event]
  cpu-clock                                          [Software event]
  cpu-migrations OR migrations                       [Software event]
  dummy                                              [Software event]
  emulation-faults                                   [Software event]
  major-faults                                       [Software event]
  minor-faults                                       [Software event]
  page-faults OR faults                              [Software event]
  task-clock                                         [Software event]

  L1-dcache-load-misses                              [Hardware cache event]
  L1-dcache-loads                                    [Hardware cache event]
```


HYU 한양대학교 HANYANG UNIVERSITY

Perf Commands – perf stat

- Generate the statistics of the events that are occurred during process execution

```
$ perf stat ./prac_mutex
```

```
jongbin@multicore-96:~/TA/Multicore/lab02$ perf stat ./prac_mutex
thread 140530927269632: local count -> 1000000
thread 140530910492416: local count -> 1000000
thread 140530902099712: local count -> 1000000
thread 140530893707008: local count -> 1000000
thread 140530885314304: local count -> 1000000
thread 140530876921600: local count -> 1000000
thread 140530868528896: local count -> 1000000
thread 140530860136192: local count -> 1000000
thread 140530851743488: local count -> 1000000
thread 140530843350784: local count -> 1000000
global count -> 10000000

 Performance counter stats for './prac_mutex':

      15981.957836      task-clock:u (msec)       #    8.397 CPUs utilized
                 0      context-switches:u        #    0.000 K/sec
                 0      cpu-migrations:u          #    0.000 K/sec
               126      page-faults:u             #    0.008 K/sec
     3,999,649,184      cycles:u                  #    0.250 GHz
       660,814,873      instructions:u            #    0.17  insn per cycle
       171,686,540      branches:u                #   10.743 M/sec
         3,501,680      branch-misses:u           #    2.04% of all branches

       1.903217149 seconds time elapsed
```


HYU 한양대학교 HANYANG UNIVERSITY

Perf Commands – perf record

- Sampling events on per-thread, per-process, per-cpu basis and generate the output file

```
$ sudo perf record -g ./prac_mutex
```

```
thread 140498458892032: local count -> 1000000
thread 140498450499328: local count -> 1000000
thread 140498442106624: local count -> 1000000
thread 140498433713920: local count -> 1000000
thread 140498425321216: local count -> 1000000
thread 140498416928512: local count -> 1000000
thread 140498408535808: local count -> 1000000
thread 140498400143104: local count -> 1000000
thread 140498391750400: local count -> 1000000
thread 140498383357696: local count -> 1000000
global count -> 10000000
[ perf record: Woken up 18 times to write data ]
[ perf record: Captured and wrote 6.830 MB perf.data (57555 samples) ]
jongbin@multicore-96:~/TA/Multicore/lab02$ ls
perf.data  prac_mutex  prac_mutex.cpp
```


HYU 한양대학교 HANYANG UNIVERSITY

Perf Commands – perf record

- Sampling events on a running process

```
$ sudo perf record -g -p 54487
```

```
jongbin@multicore-96:~/TA/Multicore/lab02$ ps u
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
jongbin   53728  0.0  0.0  22916  5600 pts/0    Ss   01:12   0:00 -bash
jongbin   54465  0.5  0.0  22880  5608 pts/4    Ss   03:33   0:00 -bash
jongbin   54487  0.0  0.0  98812  9580 pts/0    Sl+  03:33   0:05 ./prac_mutex
jongbin   54498  0.0  0.0  47428  7512 pts/4    R+   03:33   0:00 ps u
```

```
$ sudo perf record -g -p `pidof prac_mutex`
```


HYU 한양대학교 HANYANG UNIVERSITY

Perf Commands – perf report

- Analyze the output file generated from perf record

```
$ sudo perf report -g graph --no-children
```

```
Samples: 57K of event 'cycles:ppp', Event count (approx.): 715903980698041
  Overhead  Command     Shared Object       Symbol
+   30.00%  prac_mutex  [kernel.vmlinux]    [k] native_queued_spin_lock_slowpath
+   17.82%  prac_mutex  [kernel.vmlinux]    [k] futex_wake
+    8.91%  prac_mutex  libpthread-2.23.so  [.] pthread_mutex_unlock
+    8.60%  prac_mutex  [kernel.vmlinux]    [k] sys_futex
+    8.26%  prac_mutex  [kernel.vmlinux]    [k] try_to_wake_up
+    5.84%  prac_mutex  libpthread-2.23.so  [.] pthread_mutex_lock
+    5.53%  prac_mutex  [kernel.vmlinux]    [k] __unqueue_futex
+    4.30%  prac_mutex  [kernel.vmlinux]    [k] futex_wait_setup
+    3.99%  prac_mutex  [kernel.vmlinux]    [k] update_numa_stats
+    3.38%  prac_mutex  [kernel.vmlinux]    [k] __perf_event_task_sched_in
+    3.07%  prac_mutex  [kernel.vmlinux]    [k] update_cfs_rq_h_load
     0.31%  prac_mutex  [kernel.vmlinux]    [k] hash_futex
     0.00%  prac_mutex  [kernel.vmlinux]    [k] get_futex_value_locked
     0.00%  prac_mutex  [kernel.vmlinux]    [k] _raw_spin_lock
     0.00%  prac_mutex  libpthread-2.23.so  [.] __lll_lock_wait
     0.00%  prac_mutex  prac_mutex          [.] thread_func
```

Perf Commands – perf annotate

- Analyze the output file generated from perf record in the instruction level

```
$ sudo perf annotate pthread_mutex_lock
```

```
pthread_mutex_lock  /lib/x86_64-linux-gnu/libpthread-2.23.so
        58:    and    $0x80,%edx
  0.07         mov    $0x1,%edi
  2.02         xor    %eax,%eax
  0.03         mov    %edx,%esi
 40.92         lock   cmpxchg %edi,(%r8)
             ↓ je     84
  0.79         lea    (%r8),%rdi
               sub    $0x80,%rsp
  0.40       → callq  __lll_lock_wait
  0.63         add    $0x80,%rsp
  6.42   84:   mov    0x8(%r8),%edx
               test   %edx,%edx
             ↓ jne    209
  0.30   90:   mov    %fs:0x2d0,%eax
  0.13         mov    %eax,0x8(%r8)
  2.12         addl   $0x1,0xc(%r8)
```

Perf Commands – perf annotate

- Analyze the output file generated from perf record in the instruction level

  You can run the *annotate* command simply in the report screen

```
Samples: 57K of event 'cycles:ppp', Event count (approx.): 715903980698041
  Overhead  Command     Shared Object       Symbol
+   30.00%  prac_mutex  [kernel.vmlinux]    [k] native_queued_spin_lock_slowpath
+   17.82%  prac_mutex  [kernel.vmlinux]    [k] futex_wake
-    8.91%  prac_mutex  libpthread-2.23.so  [.] pthread_mutex_unlock
     pthread_mutex_unlock
     start_thread
+    8.60%  prac_mutex  [kernel.vmlinux]    [k] sys_futex
+    8.26%  prac_mutex  [kernel.vmlinux]    [k] try_to_wake_up
+    5.84%  prac_mutex  libpthread-2.23.so  [.] pthread_mutex_lock
+    5.53%  prac_mutex  [kernel.vmlinux]    [k] __unqueue_futex
+    4.30%  prac_mutex  [kernel.vmlinux]    [k] futex_wait_setup
+    3.99%  prac_mutex  [kernel.vmlinux]    [k] update_numa_stats
+    3.38%  prac_mutex  [kernel.vmlinux]    [k] __perf_event_task_sched_in
+    3.07%  prac_mutex  [kernel.vmlinux]    [k] update_cfs_rq_h_load
     0.31%  prac_mutex  [kernel.vmlinux]    [k] hash_futex
     0.00%  prac_mutex  [kernel.vmlinux]    [k] get_futex_value_locked
     0.00%  prac_mutex  [kernel.vmlinux]    [k] _raw_spin_lock
     0.00%  prac_mutex  libpthread-2.23.so  [.] __lll_lock_wait
     0.00%  prac_mutex  prac_mutex          [.] thread_func
```

```
pthread_mutex_unlock  /lib/x86_64-linux-gnu/libpthread-2.23.so

        Disassembly of section .text:

        000000000000b4c0 <__pthread_mutex_unlock>:
        __GI___pthread_mutex_unlock():
 15.09    mov    0x10(%rdi),%esi
          mov    %rdi,%rdx
  2.13    mov    %esi,%eax
  0.27    mov    %esi,%r8d
  0.11    and    $0x17f,%eax
          and    $0x7c,%r8d
        ↓ jne    50
  2.35    test   %eax,%eax
        ↓ jne    60
  0.21 1a:  movl   $0x0,0x8(%rdx)
  0.53      subl   $0x1,0xc(%rdx)
  4.32      and    $0x80,%esi
 72.05      lock   decl   (%rdx)
        ↓ je     46
  1.12      lea    (%rdx),%rdi
```

Choose a symbol and press ‘a’ key

More about Perf…

Perf Tutorial
https://perf.wiki.kernel.org/index.php/Tutorial

Thank You