perf

(Linux profiling with performance counter)

Concurrent Programming

Scalable Computing Systems Laboratory
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 available events to be measured with perf

```
$ sudo perf list
```

```
mrbin2002@ubuntu:~/TA/Multicore$ sudo perf list

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

  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]

  msr/smi/                                           [Kernel PMU event]
  msr/tsc/                                           [Kernel PMU event]
  power/energy-cores/                                [Kernel PMU event]
  power/energy-gpu/                                  [Kernel PMU event]
  power/energy-pkg/                                  [Kernel PMU event]

  rNNN                                               [Raw hardware event descriptor]
  cpu/t1=v1[,t2=v2,t3 ...]/modifier                  [Raw hardware event descriptor]
   (see 'man perf-list' on how to encode it)

  mem:<addr>[/len][:access]                          [Hardware breakpoint]
```

Perf Commands – perf stat

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

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

```
mrbin2002@ubuntu:~/TA/Multicore/lab2$ sudo perf stat ./prac_mutex
thread 140504191280896, local count: 1000000
thread 140504182888192, local count: 1000000
thread 140504174495488, local count: 1000000
thread 140504166102784, local count: 1000000
thread 140504157710080, local count: 1000000
thread 140504149317376, local count: 1000000
thread 140504140924672, local count: 1000000
thread 140504132531968, local count: 1000000
thread 140504124139264, local count: 1000000
thread 140504115746560, local count: 1000000
global count: 10000000

 Performance counter stats for './prac_mutex':

       3303.401893      task-clock (msec)         #    3.393 CPUs utilized
           303,808      context-switches          #    0.092 M/sec
           196,931      cpu-migrations            #    0.060 M/sec
                82      page-faults               #    0.025 K/sec
   <not supported>      cycles
   <not supported>      stalled-cycles-frontend
   <not supported>      stalled-cycles-backend
   <not supported>      instructions
   <not supported>      branches
   <not supported>      branch-misses

       0.973515568 seconds time elapsed
```

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
```

```
mrbin2002@ubuntu:~/TA/Multicore/lab2$ sudo perf record -g ./prac_mutex
thread 139750672590592, local count: 1000000
thread 139750664197888, local count: 1000000
thread 139750655805184, local count: 1000000
thread 139750647412480, local count: 1000000
thread 139750639019776, local count: 1000000
thread 139750630627072, local count: 1000000
thread 139750622234368, local count: 1000000
thread 139750613841664, local count: 1000000
thread 139750605448960, local count: 1000000
thread 139750597056256, local count: 1000000
global count: 10000000
[ perf record: Woken up 5 times to write data ]
[ perf record: Captured and wrote 1.336 MB perf.data (13013 samples) ]
mrbin2002@ubuntu:~/TA/Multicore/lab2$ ls
Makefile  perf.data  prac_mutex  prac_mutex.cpp  prac_mutex.o
```

Perf Commands – perf record

- Sampling events on a running process

```
$ sudo perf record -g -p 7553
```

```
mrbin2002@ubuntu:~/TA/Multicore/lab2$ ps u
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
mrbin20+   2491  0.0  0.0  28056  6716 pts/1    Ss   15:15   0:00 -bash
mrbin20+   7425  0.2  0.0  28056  6720 pts/10   Ss   15:39   0:00 -bash
mrbin20+   7553  403  0.1  94812  9656 pts/1    Sl+  15:40   0:16 ./prac_mutex
mrbin20+   7576  0.0  0.1  30904  7160 pts/10   R+   15:40   0:00 ps u
mrbin2002@ubuntu:~/TA/Multicore/lab2$ sudo perf record -g -p 7553
[ perf record: Woken up 17 times to write data ]
[ perf record: Captured and wrote 2.006 MB perf.data (19992 samples) ]
```

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

Perf Commands – perf report

- Analyze the output file generated from perf record

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

```
Samples: 13K of event 'cpu-clock', Event count (approx.): 3288500000
  Overhead  Command     Shared Object        Symbol
+   16.32%  prac_mutex  [kernel.kallsyms]    [k] _raw_spin_unlock_irqrestore
+   13.25%  prac_mutex  libpthread-2.19.so   [.] pthread_mutex_lock
+    9.21%  prac_mutex  libpthread-2.19.so   [.] pthread_mutex_unlock
+    9.05%  prac_mutex  [kernel.kallsyms]    [k] native_queued_spin_lock_slowpath
+    6.53%  prac_mutex  libpthread-2.19.so   [.] __lll_lock_wait
+    6.01%  prac_mutex  [kernel.kallsyms]    [k] futex_wake
+    5.99%  prac_mutex  [kernel.kallsyms]    [k] finish_task_switch
+    5.55%  prac_mutex  [kernel.kallsyms]    [k] futex_wait_setup
+    4.85%  prac_mutex  [kernel.kallsyms]    [k] _raw_spin_lock
+    4.09%  prac_mutex  [kernel.kallsyms]    [k] entry_SYSCALL_64_after_swapgs
+    2.97%  prac_mutex  [kernel.kallsyms]    [k] get_futex_key_refs.isra.13
+    2.73%  prac_mutex  [kernel.kallsyms]    [k] hash_futex
+    1.96%  prac_mutex  [kernel.kallsyms]    [k] get_futex_key
+    1.96%  prac_mutex  [kernel.kallsyms]    [k] get_futex_value_locked
+    1.47%  prac_mutex  libpthread-2.19.so   [.] __lll_unlock_wake
+    1.34%  prac_mutex  [kernel.kallsyms]    [k] futex_wait
+    1.34%  prac_mutex  prac_mutex           [.] _Z10ThreadFuncPv
```

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-2.19.so
        |          lea    0x16(%r8),%rsi
        |          mov    %r8,%rdi
        |          and    $0x80,%edx
        |          add    $0x8,%rsp
        |        ↓ jmpq   7af0
        |          nop
        |  58:     and    $0x80,%esi
  0.40  |          mov    $0x1,%edi
  0.40  |          xor    %eax,%eax
        |          lock   cmpxchg %edi,(%r8)
 59.04  |        ↓ jne    23a
  0.40  |          mov    0x8(%r8),%eax
  9.24  |          test   %eax,%eax
        |        ↓ jne    dd
  0.40  |  78:     mov    %fs:0x2d0,%eax
  0.29  |          addl   $0x1,0xc(%r8)
  1.38  |          mov    %eax,0x8(%r8)
  0.57  |          xor    %eax,%eax
        |  8b:     add    $0x8,%rsp
        |        ← retq
        |  90:     cmp    $0x100,%eax
```

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: 13K of event 'cpu-clock', Event count (approx.): 3288500000
  Overhead  Command      Shared Object        Symbol
+   16.32%  prac_mutex   [kernel.kallsyms]    [k] _raw_spin_unlock_irqrestore
-   13.25%  prac_mutex   libpthread-2.19.so   [.] pthread_mutex_lock
     pthread_mutex_lock
     start_thread
+    9.21%  prac_mutex   libpthread-2.19.so   [.] pthread_mutex_unlock
+    9.05%  prac_mutex   [kernel.kallsyms]    [k] native_queued_spin_lock_slowpath
+    6.53%  prac_mutex   libpthread-2.19.so   [.] __lll_lock_wait
+    6.01%  prac_mutex   [kernel.kallsyms]    [k] futex_wake
+    5.99%  prac_mutex   [kernel.kallsyms]    [k] finish_task_switch
+    5.55%  prac_mutex   [kernel.kallsyms]    [k] futex_wait_setup
+    4.85%  prac_mutex   [kernel.kallsyms]    [k] _raw_spin_lock
+    4.09%  prac_mutex   [kernel.kallsyms]    [k] entry_SYSCALL_64_after_swapgs
+    2.97%  prac_mutex   [kernel.kallsyms]    [k] get_futex_key_refs.isra.13
+    2.73%  prac_mutex   [kernel.kallsyms]    [k] hash_futex
+    1.96%  prac_mutex   [kernel.kallsyms]    [k] get_futex_key
+    1.96%  prac_mutex   [kernel.kallsyms]    [k] get_futex_value_locked
+    1.47%  prac_mutex   libpthread-2.19.so   [.] __lll_unlock_wake
+    1.34%  prac_mutex   [kernel.kallsyms]    [k] futex_wait
+    1.34%  prac_mutex   prac_mutex           [.] _Z10ThreadFuncPv
+    0.97%  prac_mutex   [kernel.kallsyms]    [k] sys_futex
```

```
pthread_mutex_lock  /lib/x86_64-linux-gnu/libpthread-2.19.so
        |          lea    0x16(%r8),%rsi
        |          mov    %r8,%rdi
        |          and    $0x80,%edx
        |          add    $0x8,%rsp
        |        ↓ jmpq   7af0
        |          nop
        | 58:      and    $0x80,%esi
  0.40  |          mov    $0x1,%edi
  0.40  |          xor    %eax,%eax
        |          lock   cmpxchg %edi,(%r8)
 59.04  |        ↓ jne    23a
  0.40  |          mov    0x8(%r8),%eax
  9.24  |          test   %eax,%eax
        |        ↓ jne    dd
  0.40  | 78:      mov    %fs:0x2d0,%eax
  0.29  |          addl   $0x1,0xc(%r8)
  1.38  |          mov    %eax,0x8(%r8)
  0.57  |          xor    %eax,%eax
        | 8b:      add    $0x8,%rsp
        |        ← retq
        | 90:      cmp    $0x100,%eax
```

Choose a symbol and press ‘a’ key

More about Perf…

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

Thank You