

perf

(Linux profiling with performance counter)

Multicore Programming

Introduction

- What is Perf?
- Perf Commands
- Example

What is Perf?

- 리눅스의 시스템 성능 측정 도구
- Kernel, User mode 모두 측정 가능
- 여러가지 종류의 Event(Hardware, Software, Hardware cache, ...)에 대한 측정 가능
- 최신 리눅스 커널에 포함

Perf Commands

- perf list
perf를 통해 측정 가능한 event의 종류를 확인한다.
- perf stat
Event count를 얻는다.
- perf record
Events를 sampling하여 output 파일을 생성한다.
- perf report
perf record를 통해 생성된 file을 분석한다.
- perf annotate
perf record를 통해 생성된 file을 instruction level에서 분석한다.
- more Commands, but not today

Perf Commands – perf list

perf를 통해 측정 가능한 event의 종류를 확인한다.

```
$ sudo perf list
```

```
mrbin2002@ubuntu:~/TA_multicore/prac_mutex$ sudo perf list

List of pre-defined events (to be used in -e):
  cpu-clock                [Software event]
  task-clock               [Software event]
  page-faults OR faults   [Software event]
  context-switches OR cs  [Software event]
  cpu-migrations OR migrations [Software event]
  minor-faults            [Software event]
  major-faults            [Software event]
  alignment-faults        [Software event]
  emulation-faults        [Software event]
  dummy                   [Software 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>[:access]      [Hardware breakpoint]
```

Perf Commands – perf stat

타겟 프로세스가 실행되는 동안 발생하는 event의 count를 측정한다.

```
$ sudo perf stat ./a.out
```

```
mrbin2002@ubuntu:~/TA_multicore/prac_mutex$ sudo perf stat ./a.out
thread 30770944, local count: 1250000
thread 22378240, local count: 1250000
thread 13985536, local count: 1250000
thread 5592832, local count: 1250000
thread -2799872, local count: 1250000
thread -11192576, local count: 1250000
thread -19585280, local count: 1250000
thread -27977984, local count: 1250000
thread -36370688, local count: 1250000
global count: 11250000

Performance counter stats for './a.out':

    11574.015841      task-clock (msec)    #    6.143 CPUs utilized
      358,797        context-switches          #    0.031 M/sec
        5,814        cpu-migrations            #    0.502 K/sec
           80        page-faults              #    0.007 K/sec
            0         cycles                  #    0.000 GHz
            0      stalled-cycles-frontend    #    0.00% frontend cycles idle
            0      stalled-cycles-backend     #    0.00% backend  cycles idle
            0      instructions               #
            0      branches                   #    0.000 K/sec
            0      branch-misses              #    0.000 K/sec

    1.884165349 seconds time elapsed
```

Perf Commands – perf record

Event를 sampling하여 output file을 생성한다.

```
$ sudo perf record -g ./a.out
```

```
mrbin2002@ubuntu:~/TA_multicore/prac_mutex$ sudo perf record -g ./a.out
thread -554502400, local count: 1250000
thread -562895104, local count: 1250000
thread -571287808, local count: 1250000
thread -579680512, local count: 1250000
thread -588073216, local count: 1250000
thread -596465920, local count: 1250000
thread -604858624, local count: 1250000
thread -613251328, local count: 1250000
thread -621644032, local count: 1250000
global count: 11250000
[ perf record: Woken up 8 times to write data ]
[ perf record: Captured and wrote 3.018 MB perf.data (~131878 samples) ]
```

(process를 실행시키고 종료될 때 까지의 event sampling)

Perf Commands – perf record

Event를 sampling하여 output file을 생성한다.

```
$ sudo perf record -g -p 24123
```

```
mrbin2002@ubuntu:~/TA_multicore/prac_mutex$ ps u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
mrbin20+  4962  0.0  0.1  29288  6788 pts/7    Ss   Sep21   0:00 /bin/bash
mrbin20+ 10971  0.5  0.2  57844 10416 pts/7    S    07:23   5:40 vi project
mrbin20+ 11292  0.0  0.1  29308  6980 pts/7    S    07:25   0:00 /bin/bash
mrbin20+ 15421  0.0  0.1  29276  6776 pts/8    Ss+  Sep21   0:00 /bin/bash
mrbin20+ 16771  0.0  0.1  29332  7000 pts/7    S    Sep21   0:00 /bin/bash
mrbin20+ 24123  758  0.0 826096  1996 pts/7    Sl+  23:11   1:53 ./a.out
mrbin20+ 24271  0.0  0.0  23868  2540 pts/0    R+   23:11   0:00 ps u
mrbin20+ 31826  0.0  0.1  29276  6716 pts/0    Ss   Sep21   0:00 bash
mrbin20+ 32161  0.6  0.2  58428 11112 pts/7    S    Sep21  14:19 vi project
mrbin2002@ubuntu:~/TA_multicore/prac_mutex$ sudo perf record -g -p 24123
[sudo] password for mrbin2002:
[ perf record: Woken up 115 times to write data ]
[ perf record: Captured and wrote 19.230 MB perf.data (~840175 samples) ]
```

(이미 실행중인 process에 대한 event sampling)

Perf Commands – perf report

perf record를 통해 생성된 file을 분석한다.

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

```
Samples: 32K of event 'cpu-clock', Event count (approx.): 8086500000
Overhead Command Shared Object Symbol
- 74.04% a.out [kernel.kallsyms] [k] _raw_spin_lock
- _raw_spin_lock
- 42.96% do_futex
  sys_futex
  system_call_fastpath
  __lll_unlock_wake
  start_thread
- 31.07% futex_wait
  do_futex
  sys_futex
  system_call_fastpath
  __lll_lock_wait
  start_thread
+ 0.01% futex_wait_setup
+ 0.01% futex_wake
+ 12.01% a.out [kernel.kallsyms] [k] _raw_spin_unlock_irqrestore
+ 4.00% a.out [kernel.kallsyms] [k] finish_task_switch
+ 1.99% a.out libpthread-2.19.so [.] pthread_mutex_lock
+ 1.86% a.out libpthread-2.19.so [.] pthread_mutex_unlock
```

Perf Commands – perf annotate

perf record를 통해 생성된 file을 instruction level에서 분석한다.

\$ sudo perf annotate _raw_spin_lock

```
raw_spin_lock /proc/kcore

Disassembly of section load0:

ffffff817b6260 <load0>:
  nop
0.01  mov     $0x20000,%eax
      lock  xadd  %eax,(%rdi)
1.08  mov     %eax,%edx
      shr   $0x10,%edx
0.01  cmp     %ax,%dx
      ↓ jne   19
      ← retq
0.01  19:  mov     %edx,%ecx
0.01  movzwl %dx,%esi
      1e:  mov     $0x8000,%eax
0.01  ↓ jmp     2f
      nop
5.87  28:  pause
34.02  sub     $0x1,%eax
0.01  ↓ je      4f
33.16  2f:  movzwl (%rdi),%edx
22.78  mov     %edx,%r8d
0.11  xor     %ecx,%r8d
2.71  and     $0xfffe,%r8d
```

Perf Commands – perf annotate

perf record를 통해 생성된 file을 instruction level에서 분석한다.

perf report 화면에서 annotate command 수행 가능

Samples: 18K of event 'cpu-clock', Event count (approx.): 4603000000

Overhead	Command	Shared Object	Symbol
+ 72.84%	a.out	[kernel.kallsyms]	[k] _raw_spin_lock
+ 12.25%	a.out	[kernel.kallsyms]	[k] _raw_spin_unlock_irqrestore
+ 4.05%	a.out	[kernel.kallsyms]	[k] finish_task_switch
+ 2.49%	a.out	libpthread-2.19.so	[.] pthread_mutex_lock
+ 2.07%	a.out	libpthread-2.19.so	[.] pthread_mutex_unlock
+ 0.84%	a.out	[kernel.kallsyms]	[k] futex_wait_setup
+ 0.74%	a.out	libpthread-2.19.so	[.] __lll_lock_wait
+ 0.68%	a.out	[kernel.kallsyms]	[k] futex_wake
+ 0.47%	a.out	[kernel.kallsyms]	[k] system_call_after_swapgs
+ 0.32%	a.out	libpthread-2.19.so	[.] __lll_unlock_wake
+ 0.32%	a.out	[kernel.kallsyms]	[k] get_futex_key_refs.isra.13
+ 0.31%	a.out	[kernel.kallsyms]	[k] hash_futex
+ 0.30%	a.out	[kernel.kallsyms]	[k] do_futex
+ 0.27%	a.out	[kernel.kallsyms]	[k] sys_futex
+ 0.25%	a.out	[kernel.kallsyms]	[k] get_futex_key
+ 0.24%	a.out	[kernel.kallsyms]	[k] get_futex_value_locked
+ 0.24%	a.out	a.out	[.] _Z10ThreadFuncPv
+ 0.21%	a.out	[kernel.kallsyms]	[k] futex_wait
+ 0.13%	a.out	[kernel.kallsyms]	[k] _raw_spin_unlock
+ 0.12%	a.out	[kernel.kallsyms]	[k] __unqueue_futex
+ 0.10%	a.out	[kernel.kallsyms]	[k] futex_wait_queue_me
+ 0.10%	a.out	[kernel.kallsyms]	[k] rcu_note_context_switch
+ 0.09%	a.out	[kernel.kallsyms]	[k] schedule



```
Annotate raw spin lock
Zoom into a.out(27415) thread
Zoom into the Kernel DSO
Browse map details
Run scripts for samples of thread [a.out]
Run scripts for samples of symbol [_raw_spin_lock]
Run scripts for all samples
Switch to another data file in PWD
Exit
```



raw_spin_lock /proc/kcore

Disassembly of section load0:

fffffffb17b6260 <load0>:

0.01	nop
	mov \$0x20000,%eax
	lock xadd %eax,(%rdi)
1.08	mov %eax,%edx
	shr \$0x10,%edx
0.01	cmp %ax,%dx
	↓ jne 19
	← retq
0.01	19: mov %edx,%ecx
0.01	movzwl %dx,%esi
	1e: mov \$0x8000,%eax
0.01	↓ jmp 2f
	nop
5.87	28: pause
34.02	sub \$0x1,%eax
0.01	↓ je 4f
33.10	2f: movzwl (%rdi),%edx
22.78	mov %edx,%r8d
0.11	xor %ecx,%r8d
2.71	and \$0xffff,%r8d

오른쪽 화살표키(->) 누름

엔터키 누름

More about Perf...

그 밖의 다양한 방식의 측정이 가능하다.

Perf Tutorial

<https://perf.wiki.kernel.org/index.php/Tutorial>

Thank You
