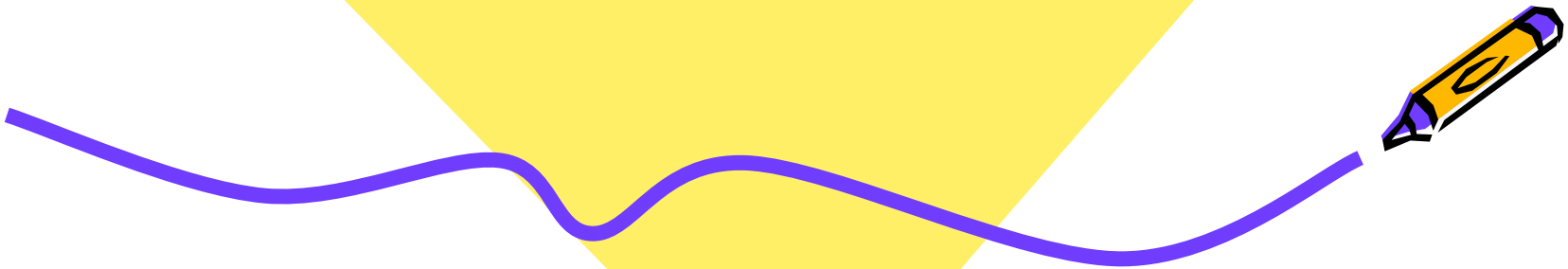
性能调优工具

Perf 介绍

philpeng



Agenda

- 性能优化必要性
- Perf简介
- 相关背景知识
- Perf使用
- Q & A

性能优化必要性

- 系统写完了，各种功能测试也没问题了，就可以上线了？
- 若排序打分库极限qps只有1，其结果精度再大也无法上线
- 公司大把机器，可以使用大量的机器来解决？但这迟早会是一个瓶颈
- 性能优化，可以在硬件资源有限的条件下，最大的发挥其潜能

Perf 简介

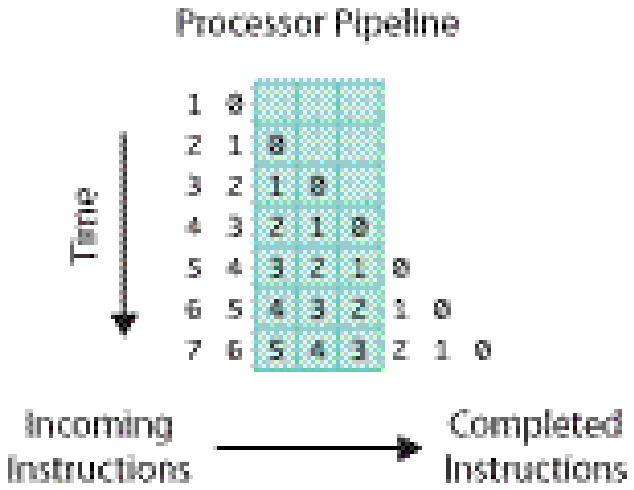
- 一款随Linux内核代码一同发布和维护的性能诊断工具
  - 内核2.6.31加入Performance Counter
  - 内核2.6.32改名为Performance Event
- 目前版本0.0.2，其对OProfile的优势是，它可以最先应用到加入内核的新特性。
- 利用tracepoint、内核中的特殊计数器、PMU(Performance Monitor Unit)进行性能统计
- 可用来分析应用程序的性能问题(per 线程)，也可用来分析内核的性能问题

相关背景知识 (1)

- 1. 硬件特性之cache
  - 处理器的指令执行速度很快
  - 内存读写速度相对较慢
  - Cache与处理器处理速度相当，把常用的数据存于cache，能较大提高性能。

相关背景知识 (2)

- 2. 硬件特性之流水线、超标量、乱序执行
  - 并行
  - 一个时钟周期可以同时处理多条指令，分别被流水线的不同部分处理



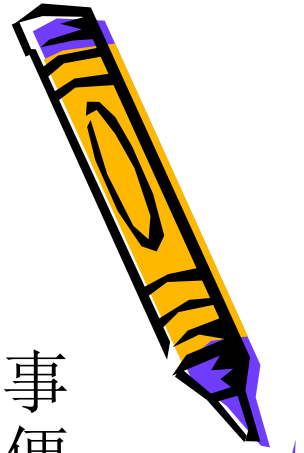
相关背景知识 (3)

- 2. 硬件特性之流水线、超标量、乱序执行 (cont.)
  - 超标量(super-scalar)是指一个时钟周期发射多条指令的流水线机器架构
  - 为充分利用处理器的流水线，不一定按照程序的执行顺序执行，其指令有可能被乱序执行
  - 基本条件
    - 相邻的指令相互没有依赖关系

相关背景知识 (4)

- 3. 硬件特性之分支预测
  - 分支指令对软件性能有较大的影响
  - 例如，在处理器采用流水线设计后，处理器顺序读取指令，并预取下几条指令，如果分支结果是跳到其他指令，则被其预取的指令被抛弃，从而影响性能。

相关背景知识 (5)



- 4. PMU
  - 允许软件对某种硬件事件设置counter，当事件发生的次数超过counter内设置的值后，便产生中断。如，cache miss次数
  - 捕获到中断，便可以分析程序对以上提到的硬件特性的利用率

相关背景知识 (6)

- 5. Tracepoint
  - 散落在内核源码中的hook，可在特定的代码被运行时被触发。
  - 例如，可利用潜伏在内存分配器中的tracepoint，来分析应用程序运行期间内存管理模块的行为

Perf使用 (1)

- perf 所有命令

```
usage: perf [--version] [--help] COMMAND [ARGS]

The most commonly used perf commands are:
  annotate        Read perf.data (created by perf record) and display annotated code
  archive         Create archive with object files with build-ids found in perf.data file
  bench           General framework for benchmark suites
  buildid-cache   Manage build-id cache.
  buildid-list    List the buildids in a perf.data file
  diff            Read two perf.data files and display the differential profile
  inject          Filter to augment the events stream with additional information
  kmem            Tool to trace/measure kernel memory(slab) properties
  kvm             Tool to trace/measure kvm guest os
  list            List all symbolic event types
  lock            Analyze lock events
  probe           Define new dynamic tracepoints
  record          Run a command and record its profile into perf.data
  report          Read perf.data (created by perf record) and display the profile
  sched           Tool to trace/measure scheduler properties (latencies)
  stat            Run a command and gather performance counter statistics
  test            Runs sanity tests.
  timechart       Tool to visualize total system behavior during a workload
  top             System profiling tool.
  trace           Read perf.data (created by perf record) and display trace output

See 'perf help COMMAND' for more information on a specific command.
```

Perf使用 (2)

- 内核版本需2.6.31以上

- 常用命令:
  - perf list
  - perf stat
  - perf top
  - perf record
  - perf report
  - perf annotate

Perf使用 (3)

- 1. perf list
  - 列出所有能够触发perf采样点的事件
  - perf/Oprofile的基本原理都是对被监测对象进行采样
  - 采样的方式很多，最简单的情形是根据tick中断进行
  - 除此之外，我们还可以cache miss事件触发进行采样

```
reltext@GaoXin_172_28_0_48:~> perf list

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

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

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

Perf使用 (4)

- 1. perf list (cont.)
- 以上触发事件可划分成三类
  - 硬件事件
    - 由PMU硬件产生的事件，如cache命中
  - 软件事件
    - 内核软件产生的事件，如进程切换，tick数等
  - tracepoint
    - 内核中的静态tracepoint触发的事件，如内存分配器的分配次数等

Perf使用 (5)

- 2. perf stat
  - 通过概括精简的方式提供被调试程序运行的整体情况和汇总数据
  - 通过perf stat，我们可以很快分析出被调试程序是CPU密集型还是IO密集型，从而可以进行下一步的优化
  - 使用方式

```
usage: perf stat [<options>] [<command>]

    -e, --event <event>   event selector. use 'perf list' to list available events
    -i, --no-inherit      child tasks do not inherit counters
    -p, --pid <n>         stat events on existing process id
    -t, --tid <n>         stat events on existing thread id
    -a, --all-cpus        system-wide collection from all CPUs
    -c, --scale           scale/normalize counters
    -v, --verbose         be more verbose (show counter open errors, etc)
    -r, --repeat <n>      repeat command and print average + stddev (max: 100)
    -n, --null            null run - dont start any counters
    -B, --big-num         print large numbers with thousands' separators
```

Perf使用 (6)

- 2. perf stat (cont.)

```
#include <iostream>

void foo()
{
    int i = 0;
    for (; i < 100000000; ++i)
        ;
}

int main()
{
    int i = 0;
    for (; i < 10; ++i)
        foo();

    return 0;
}
```

```
reltext@GaoXin_172_28_0_48:~/philpeng> perf stat ./test

 Performance counter stats for `./test`:

       3776.640743  task-clock-msecs         #      0.999 CPUs
                4  context-switches         #      0.000 M/sec
                1  CPU-migrations           #      0.000 M/sec
              260  page-faults              #      0.000 M/sec
       7527267348  cycles                   #   1993.112 M/sec  (scaled from 70.12%)
       5003986311  instructions             #      0.665 IPC    (scaled from 80.10%)
       1001274866  branches                 #    265.123 M/sec  (scaled from 80.10%)
            36067  branch-misses            #      0.004 %      (scaled from 80.10%)
            17108  cache-references         #      0.005 M/sec  (scaled from 19.90%)
             9222  cache-misses             #      0.002 M/sec  (scaled from 19.90%)

       3.779392702  seconds time elapsed
```

  - 我们从以上的例子，可以很清晰地看到程序test是CPU密集型(其CPU利用率达到99.9%)

Perf使用 (7)

- 3. perf top
  - 用于实时显示当前系统的性能统计信息
  - 使用perf stat时，往往已经找到一个调优的目标，但有时候系统性能突然下降，需弄清楚到底是哪个进程作怪
  - 此时perf top可以用来观察整个系统当前的状态，通过查看其输出可以找出当前系统最耗时的某个进程

```
PerfTop:    1235 irqs/sec  kernel:16.2%  exact:  0.0% [1000Hz cycles],  (all, 4 CPUs)
------------------------------------------------------------------------------

             samples  pcnt function                       DSO
             _______ _____ _______________________________ ___________________________

            1103.00 67.0% __GI_memcpy                     /lib64/libc-2.4.so
              40.00  2.4% schedule                       [kernel.kallsyms]
              36.00  2.2% find_busiest_group             [kernel.kallsyms]
              22.00  1.3% fput                           [kernel.kallsyms]
              22.00  1.3% thread_return                  [kernel.kallsyms]
              19.00  1.2% sched_clock                    [kernel.kallsyms]
              18.00  1.1% tick_nohz_stop_sched_tick      [kernel.kallsyms]
              16.00  1.0% fget_light                     [kernel.kallsyms]
              16.00  1.0% _spin_lock_irqsave            [kernel.kallsyms]
              13.00  0.8% perf_session__mmap_read_counter /usr/local/perf35/bin/perf
              13.00  0.8% perf_poll                     [kernel.kallsyms]
              13.00  0.8% read_tsc                      [kernel.kallsyms]
              13.00  0.8% weighted_cpuload             [kernel.kallsyms]
              11.00  0.7% _spin_lock_irq                [kernel.kallsyms]
```

Perf使用 (8)

- 4. perf record & perf report
  - 使用perf record来获取某个进程的采样信息，存到perf.data文件中，使用perf report来显示统计结果
  - perf record记录单个函数级别的统计信息，可以定位某个进程最耗时的函数
  - perf record最常用的选项是-p和-g
    - 使用-p选项，指定采样某个进程的信息
    - 还可以使用-g选项，生成函数的调用关系表

Perf使用 (9)

- 5. perf annotate
  - 读取perf.data (由前面的perf record生成)，显示注释的代码
  - 可以查看程序中哪些代码的耗时比较长

Perf使用 (10)

- 6. 其它命令
  - perf lock 分析锁性能
  - perf kmem 分析内核内存
  - perf sched 分析任务调度
  - 等等

Q&A ?

Thank you!