Performance Monitoring and Tuning with OProfile

William Cohen
Performance Tools Engineer
Red Hat, Inc.

Abstract

- The complexity of hardware and software makes it difficult to identify performance problems on computer systems. There can be unexpected interactions between the various software and hardware components that developers are unable to predict. Thus, performance monitoring tools in Linux such as OProfile are essential for identifying these performance problems, allowing the programmers to directly address the problems.

Performance Tuning Issues: Software

- Applications not single-threaded, batch-oriented benchmarks.
- Current applications:
  - Very large
  - Shared libraries
  - Multiple executables/threads
  - Indefinite runtimes (daemons)
  - Interactive (GUIs)

Performance Tuning Issue: Hardware

- Processors:
  - Many different ways to hurt performance, e.g. cache misses, serializing instructions, and branch mispredictions
  - Seemingly minor changes can have large effect on performance
- I/O devices

OProfile Introduction

- Started as John Levon's Master Thesis work
- Originally modelled after DEC Continuous Profiling Infrastructure (DCPI)
- System-wide profiler (both kernel and user code)
- Sample-based profiler
- SMP machine support
- Performance monitoring hardware support
- Relatively low overhead, typically <10%
- Designed to run for long times
- Provides flat profiles of executable files

Profiling Requirements (Red Hat)

- Hardware:
  - Intel PII, PIII, P4, P4HT, Core, and Core 2
  - AMD Athlon, Duron, and AMD64
  - Intel ia64
  - PPC64
- Software:
  - RHEL 3, RHEL 4, RHEL 5, and Fedora Core
  - oprofile rpm
- User:
  - Root access

OProfile Database

- All the data in /var/lib/oprofile/samples
- Samples files code executable name and event
- Collects samples in /var/lib/oprofile/samples/current
- Can package samples and related executables off-line analysis with oparchive

Events to Measure

- Processor specific
- Many events to choose from
- Can sample between 2 and 8 events concurrently
- ophelp lists available events
- Select events in the “opcontrol --setup”
- Initially time-based samples most useful:
  - PPro/PII/PIII/AMD: CPU_CLK_UNHALTED
  - P4: GLOBAL_POWER_EVENTS
  - IA64: CPU_CYCLES
  - TIMER_INT (fall-back profiling mechanism) default

Events to Measure (cont.)

- Performance monitoring hardware can provide additional kinds of sampling
- Pipeline stalls/serializing instructions
- Branch mis-predictions
- Cache misses
- TLB misses

Selecting Count Between Samples

- Higher count -> fewer samples and lower overhead
- Lower count -> more samples and higher overhead
- Possible to select count too low so that sampling dominates processor
- Event sample rates vary:
  - ia64 number of unused dispersal slots (max 6 per cycle)
  - cache misses from L2/L3 cache
- Default event may be too frequent sample rate, e.g. 100,000 clock cycles per sample

Shared Library and Kernel Samples

- OProfile can associate the samples for shared library with application
- Gives more complete picture of where application spends time
  - opcontrol --separate=library option
- Newer version of oprofile associate kernel samples with application, changes in user space daemon not kernel

Mapping OProfile Data to Source

- If gdb can debug code, e.g. get line numbers, then OProfile can map information back to source.
- "-g" used information to map back to lines
- Per function view (opreport -l) doesn't use debug symbols
- Debuginfo RPMs now being built by default, allow characterization of executables in stock RPMs

Data Collection from OProfile RPM Build

- Find out where the process spends time during a build of the OProfile RPM
- Environment:
  - Processor: Intel Pentium M 1.6GHz
  - Memory: 1536MB RAM
  - Disc:
    - hda: Hitachi Travelstar 5K80 family, ATA DISK drive
  - Running RHEL5.3

Procedure

- Install the OProfile source RPM via:

```
rpm -Uvh oprofile-0.9.3-18.el5.src.rpm
```

- Start oprofile with:

```
opcontrol --setup \
  --vmlinux=/usr/lib/debug/lib/modules/`uname -r`\
  /vmlinux \
  --event=CPU_CLK_UNHALTED:1600000:0:1:1 \
  --separate=library
opcontrol --start
```

- Start RPM build with:

```
rpmbuild -ba oprofile.spec >& oprof.probs
```


Procedure (cont)

- When OProfile build complete stop oprofile:
  - `opcontrol --shutdown`
  - `opcontrol --save=oprofile_build`

Overall Oprofile RPM build Data

```
$ opreport -t 15 session:oprofile_build
CPU: Pentium M (P6 core), speed 600 MHz (estimated)
Counted CPU_CLK_UNHALTED events (clocks processor is not halted, and
not in a thermal trip) with a unit mask of 0x00 (No unit mask) count
1600000
CPU_CLK_UNHALT...|
  samples|      %|
------------------
    44501 36.7996 cc1plus
   CPU_CLK_UNHALT...|
     samples|      %|
   ------------------
       38701 86.9666 cc1plus
        5794 13.0199 libc-2.5.so
           5  0.0112 ld-2.5.so
           1  0.0022 [vdso] (tgid:597 range:0x690000-0x691000)
```


More Detailed Look at Executables

- Find out which RPM executable associated with:
  - `opreport --long-filenames`
  - `rpm -qf` *filename*
- Get the associated debuginfo RPMs
- Install debuginfo RPMs
- Now map samples back to source
- Some files do not have proper debuginfo files, e.g. X

Look at Kernel

- Need kernel debuginfo installed before “opcontrol –setup”

```
$ opreport /usr/lib/debug/lib/modules/`uname -r`/vmlinux \
 -l session:oprofile_build -t 2
CPU: Pentium M (P6 core), speed 600 MHz (estimated)
Counted CPU_CLK_UNHALTED events (clocks processor is not halted, and
not in a thermal trip) with a unit mask of 0x00 (No unit mask) count
1600000
samples  %        symbol name
2584     19.8739  get_page_from_freelist
638       4.9069  __copy_to_user_ll
467       3.5918  do_wp_page
455       3.4995  __handle_mm_fault
367       2.8226  mask_and_ack_8259A
```


Look at Compiler executable

```
$  opreport -l -t 1 session:oprofile_build \
/usr/libexec/gcc/i386-redhat-linux/4.1.1/cc1plus
warning: [vdso] (tgid:597 range:0x690000-0x691000) could not be
found.
CPU: Pentium M (P6 core), speed 600 MHz (estimated)
Counted CPU_CLK_UNHALTED events (clocks processor is not halted,
and not in a thermal trip) with a unit mask of 0x00 (No unit mask)
count 1600000
samples  %        image name               symbol name
1514      3.4022  libc-2.5.so              memset
962       2.1617  cc1plus                  _cpp_lex_direct
916       2.0584  cc1plus                  ggc_alloc_stat
860       1.9325  cc1plus                  ht_lookup_with_hash
685       1.5393  cc1plus                  _cpp_clean_line
681       1.5303  libc-2.5.so              memcpy
646       1.4517  cc1plus                  walk_tree
...
```


Oprofile Limitations

- Requires root access to set up and control oprofile
- Complicated set of architecture-specific events
- Sampling imprecision
- Simple-minded analysis tools

Events Available to Monitor

- OProfile uses time based events at the default event
- Time-based sampling focuses attentions correctly
- Would like to narrow down the cause of slowdown:
  - Processors have specific hardware events
  - Hardware events vary between processors
  - `ophelp` will list the available events with short descriptions

Additional OProfile Information

- OProfile:
  - http://oprofile.sourceforge.net/
  - http://people.redhat.com/wcohen

Analyzing Kernel Modules

- OProfile setup allows Oprofile to find the kernel
- Oprofile does not know where kernel modules located
- Oprofile puts samples as module name
- Opreport “-p” option can help

opreport_module output

```
# opreport -t 5 -l /ext3 -p /lib/modules/`uname -r` \
session:oprofile_build
CPU: Pentium M (P6 core), speed 600 MHz (estimated)
Counted CPU_CLK_UNHALTED events (clocks processor is not halted, and
not in a thermal trip) with a unit mask of 0x00 (No unit mask) count
1600000
warning: could not check that the binary file /lib/modules/2.6.18-
128.1.1.el5/kernel/fs/ext3/ext3.ko has not been modified since the
profile was taken. Results may be inaccurate.
samples  %        image name               symbol name
54       15.1685  ext3.ko                  ext3_get_acl
39       10.9551  ext3.ko                  __ext3_get_inode_loc
37       10.3933  ext3.ko                  ext3_mark_iloc_dirty
21        5.8989  ext3.ko                  ext3_get_blocks_handle
```


Processor Hardware Optimizations

- Many techniques to improve performance:
  - Cache memory
  - Pipelining, superscalar, and out-of-order (OOO) execution
  - Branch prediction
- The techniques usually help, but make some assumptions and can hurt performance in some cases

Cache

- Instruction and data caches
- Cache smaller but faster than main memory
- Reduce average memory access time
- Make assumptions about locality:
  - Spatial
  - Temporal

Cache Problems

- Poor spatial locality:
  - Cache line has multiple memory locations
  - Only using small part of cache line
  - Non-unit stride through memory
- Poor temporal locality
  - No reuse of data in cache
- False sharing
  - Cache line bounces between processor

Instruction Cache Example

- Same set up as before
  - Start oprofile with:

```
opcontrol --setup \
  --vmlinux=/usr/lib/debug/lib/modules/`uname -r`\
  /vmlinux \
  --event=INST_RETIRED:100000:0:1:1 \
  --event=L2_IFETCH:100000:0xf:1:1 \
  --separate=library
opcontrol --start
```

  - Start RPM build with:

```
rpmbuild -ba oprofile.spec >& oprof.probs
```

  - `opcontrol --shutdown`
  - `opcontrol --save=oprofile_build_ic`

I-Cache Output

```
$  opreport  -l session:oprofile_build_ic -t 2 /usr/libexec/gcc/i386-redhat-
linux/4.1.1/cc1pluswarning: [vdso] (tgid:4733 range:0xb82000-0xb83000) could not
be found.
CPU: Pentium M (P6 core), speed 600 MHz (estimated)
Counted INST_RETIRED events (number of instructions retired) with a unit mask of
0x00 (No unit mask) count 100000
Counted L2_IFETCH events (number of L2 instruction fetches) with a unit mask of
0x0f (All cache states) count 100000
samples  %        samples  %        symbol name
17604     2.7898  14        0.0965  _cpp_lex_direct
16869     2.6733  1         0.0069  _cpp_clean_line
16732     2.6516  191       1.3165  ggc_alloc_stat
12740     2.0189  105       0.7237  walk_tree
```


Data Cache Example

- Same set up as before
  - Start oprofile with:

```
opcontrol --setup \
  --vmlinux=/usr/lib/debug/lib/modules/`uname -r`\
  /vmlinux \
  --event=L2_RQSTS:100000:0xf:1:1 \
  --event=INST_RETIRED:100000:0:1:1 \
  --separate=library
opcontrol --start
```

  - Start RPM build with:

```
rpmbuild -ba oprofile.spec >& oprof.probs
```

  - `opcontrol --shutdown`
  - `opcontrol -save=oprofile_build_dc`

Dcache Output

```
$ opreport -l session:oprofile_build_dc -t 2 \
 /usr/libexec/gcc/i386-redhat-linux/4.1.1/cc1plus
warning: [vdso] (tgid:12283 range:0xb7b000-0xb7c000) could not be
found.
CPU: Pentium M (P6 core), speed 600 MHz (estimated)
Counted INST_RETIRED events (number of instructions retired) with a
unit mask of 0x00 (No unit mask) count 100000
Counted L2_RQSTS events (number of L2 requests) with a unit mask of
0x0f (All cache states) count 100000
samples  %        samples  %        symbol name
17854     2.8282  23        0.1274  _cpp_lex_direct
16798     2.6609  32        0.1772  _cpp_clean_line
16702     2.6457  303       1.6780  ggc_alloc_stat
12635     2.0015  136       0.7532  walk_tree
```


Pipelined, Superscalar, and OOO execution

- Most processor take multiple steps to complete the execution of an instruction
- Pipelined execution overlaps the processing stages of the instructions
- Superscalar execution allow multiple instructions to be processed concurrently by separate hardware units
- Out-of-Order execution allows instructions to be executed in the order that their input is available rather than the order specified in the code

Pipelined, Superscalar, and OOO Execution Problems

- Some instructions interfere with the parallel execution and require instructions to be serialized, e.g. floating point mode change instruction on x86
- Data dependencies may cause stalls
- Branches can influence which instruction executed
- The information about branch may not always be available
- Limits on instruction combinations, e.g. Pentium II/III instruction decoder

Branch Prediction

- Processor attempts to guess path through code
- Starts executing instructions speculatively based on information in instruction or past history
- If prediction correct, performance improvement

Branch Prediction Problems

- Branch prediction is not aways correct
- Corrective action may be needed and slow things down
- Some branches are hard to predict

Branches

- Same set up as before
  - Start oprofile with:
    - `opcontrol --setup \`
      `--vmlinux=/usr/lib/debug/lib/modules/`uname -r`\`
      `/vmlinux \`
      `--event=BR_MISS_PRED_RETIRED:100000:0:1:1 \`
      `--event=BR_INST_RETIRED:100000:0:1:1 \`
      `--separate=library`
    - opcontrol --start
  - Start RPM build with:
    - `rpmbuild -ba oprofile.spec >& oprof.probs`
  - `opcontrol --shutdown`
  - `opcontrol --save=oprofile_build_br`

Branch Results

```
$ opreport  -l session:oprofile_build_br -t 2 \
/usr/libexec/gcc/i386-redhat-linux/4.1.1/cc1plus
CPU: Pentium M (P6 core), speed 600 MHz (estimated)
Counted BR_INST_RETIRED events (number of branch instructions retired) with a
unit mask of 0x00 (No unit mask) count 100000
Counted BR_MISS_PRED_RETIRED events (number of mispredicted branches retired)
with a unit mask of 0x00 (No unit mask) count 100000
samples  %        samples  %        image name               symbol name
4661      3.5914  60        1.0629  cc1plus                  _cpp_clean_line
3477      2.6791  152       2.6926  cc1plus                  _cpp_lex_direct
2930      2.2576  37        0.6554  libc-2.5.so              strcmp
2849      2.1952  172       3.0469  cc1plus                  walk_tree
2689      2.0720  29        0.5137  cc1plus                  ggc_alloc_stat
```