Performance analysis using the perf suite

Mans Rullgard

Profiling

- Computers are slow
- Premature optimization is the root of all evil

Understanding perf

- Sample-based profiling
- Non-invasive
- Low overhead

Sample-based profiling

- Hardware event counters
- Periodic interrupts
- Produces heat map of events

Sample-based profiling

- Does not provide
  - Call counts for functions
  - Cycles needed for specific code sequences
- Statistical variations
  - Accuracy improves with increased runtime

Exact steps

- Record samples
- Analyse
- Optimise
- Synergy

Recording samples

- perf record <command>
- Counts CPU cycles
- Automatic interrupt period

Analysing samples

- High-level per DSO, function
  - perf report
- Instruction level
  - perf annotate

Example: flac encoding

- Overview

```
$ perf stat ./flac -f -o /dev/null test.wav
[...]
 Performance counter stats for './flac -f -o /dev/null test.wav':

      26821.345510    task-clock (msec)         #    0.999 CPUs utilized
               718    context-switches          #    0.027 K/sec
                 1    cpu-migrations            #    0.000 K/sec
               178    page-faults               #    0.007 K/sec
    25,746,088,637    cycles                    #    0.960 GHz
    19,453,581,691    instructions              #    0.76  insns per cycle
     1,870,878,855    branches                  #   69.753 M/sec
        39,992,280    branch-misses             #    2.14% of all branches

      26.841126339 seconds time elapsed
```

Example: flac encoding

- Overview: cache misses

```
$ perf stat -e cache-references,cache-misses ./flac ...
[...]
 Performance counter stats for './flac -f -o /dev/null test.wav':

     6,344,978,310    cache-references
        86,353,380    cache-misses              #    1.361 % of all cache refs

      26.737117149 seconds time elapsed
```

Example: flac encoding

- Record samples

```
$ taskset 2 perf record ./flac -f -o /dev/null test.wav
[...]
[ perf record: Woken up 18 times to write data ]
[ perf record: Captured and wrote 4.283 MB perf.data (~187137 samples) ]
```

perf report

```
# Samples: 112K of event 'cycles'
# Event count (approx.): 26851110962
#
# Overhead      Shared Object                                      Symbol
# ........  ..................  ..........................................
#
    40.20%  flac                FLAC__lpc_compute_autocorrelation
     9.59%  flac                FLAC__fixed_compute_best_predictor
     8.71%  flac                FLAC__lpc_compute_residual_from_qlp_coefficients
     6.55%  flac                FLAC__lpc_window_data
     5.83%  flac                FLAC__bitwriter_write_rice_signed_block
     5.70%  flac                precompute_partition_info_sums_
     4.08%  flac                FLAC__fixed_compute_residual
     3.76%  flac                FLAC__MD5Transform
     2.37%  flac                FLAC__crc16
     2.12%  flac                FLAC__MD5Accumulate
     1.75%  flac                format_input
     1.53%  flac                FLAC__stream_encoder_process
     1.30%  libc-2.20.so        memcpy
```

Source code

```
void FLAC__lpc_compute_autocorrelation(float data[], unsigned data_len,
                                       unsigned lag, float autoc[])
{
    float d;
    unsigned sample, coeff;
    const unsigned limit = data_len - lag;

    for (coeff = 0; coeff < lag; coeff++)
        autoc[coeff] = 0.0;

    for (sample = 0; sample <= limit; sample++) {
        d = data[sample];
        for (coeff = 0; coeff < lag; coeff++)
            autoc[coeff] += d * data[sample+coeff];
    }

    for (; sample < data_len; sample++) {
        d = data[sample];
        for (coeff = 0; coeff < data_len - sample; coeff++)
            autoc[coeff] += d * data[sample+coeff];
    }
}
```

perf annotate

```
 0.00 : 4fc4c:  cmp      r2, #0
 0.00 : 4fc50:  push     {..., lr}
 0.00 : 4fc54:  mov      r8, r2
 0.00 : 4fc58:  mov      r7, r1
 0.00 : 4fc5c:  mov      r9, r0
 0.00 : 4fc60:  mov      r6, r3
 0.00 : 4fc64:  rsb      r10, r2, r1
 0.00 : 4fc68:  moveq    r4, r2
 0.00 : 4fc6c:  beq      4fc84
 0.00 : 4fc70:  lsl      r4, r2, #2
 0.00 : 4fc74:  mov      r0, r3
 0.00 : 4fc78:  mov      r2, r4
 0.00 : 4fc7c:  mov      r1, #0
 0.00 : 4fc80:  bl       11414 <memset@plt>
 0.00 : 4fc84:  add      r4, r6, r4
 0.00 : 4fc88:  mov      r3, r9
 0.00 : 4fc8c:  mov      r5, #0
 1.89 : 4fc90:  vldr     s13, [r3]
 0.95 : 4fc94:  cmp      r8, #0
 0.00 : 4fc98:  movne    r12, r6
 1.02 : 4fc9c:  movne    lr, r3
 0.00 : 4fca0:  beq      4fcbc
11.14 : 4fca4:  vldmia   lr!, {s14}
 9.44 : 4fca8:  vldr     s15, [r12]
 9.45 : 4fcac:  vfma.f32 s15, s14, s13
54.76 : 4fcb0:  vstmia   r12!, {s15}
 9.19 : 4fcb4:  cmp      r12, r4
 0.00 : 4fcb8:  bne      4fca4
```

```
 1.00 : 4fcbc:  add      r5, r5, #1
 0.00 : 4fcc0:  add      r3, r3, #4
 1.03 : 4fcc4:  cmp      r10, r5
 0.00 : 4fcc8:  bcs      4fc90
 0.00 : 4fccc:  cmp      r7, r5
 0.00 : 4fcd0:  popls    {..., pc}
 0.00 : 4fcd4:  rsb      r7, r5, r7
 0.00 : 4fcd8:  add      lr, r9, r5, lsl #2
 0.00 : 4fcdc:  add      r12, r6, r7, lsl #2
 0.01 : 4fce0:  vldr     s13, [lr]
 0.00 : 4fce4:  cmp      r7, #0
 0.00 : 4fce8:  movne    r2, r6
 0.00 : 4fcec:  movne    r1, lr
 0.00 : 4fcf0:  beq      4fd0c
 0.02 : 4fcf4:  vldmia   r1!, {s14}
 0.00 : 4fcf8:  vldr     s15, [r2]
 0.00 : 4fcfc:  vfma.f32 s15, s14, s13
 0.04 : 4fd00:  vstmia   r2!, {s15}
 0.01 : 4fd04:  cmp      r12, r2
 0.00 : 4fd08:  bne      4fcf4
 0.02 : 4fd0c:  sub      r12, r12, #4
 0.00 : 4fd10:  add      lr, lr, #4
 0.01 : 4fd14:  cmp      r12, r6
 0.00 : 4fd18:  sub      r7, r7, #1
 0.00 : 4fd1c:  bne      4fce0
 0.00 : 4fd20:  pop      {..., pc}
```

The main loop

```
 1.89 : 4fc90:   vldr      s13, [r3]
 0.95 : 4fc94:   cmp       r8, #0
 0.00 : 4fc98:   movne     r12, r6
 1.02 : 4fc9c:   movne     lr, r3
 0.00 : 4fca0:   beq       4fcbc
11.14 : 4fca4:   vldmia    lr!, {s14}
 9.44 : 4fca8:   vldr      s15, [r12]
 9.45 : 4fcac:   vfma.f32  s15, s14, s13
54.76 : 4fcb0:   vstmia    r12!, {s15}
 9.19 : 4fcb4:   cmp       r12, r4
 0.00 : 4fcb8:   bne       4fca4
 1.00 : 4fcbc:   add       r5, r5, #1
 0.00 : 4fcc0:   add       r3, r3, #4
 1.03 : 4fcc4:   cmp       r10, r5
 0.00 : 4fcc8:   bcs       4fc90
```

Optimise

- Invert the loop
  - Before

```
for (sample = 0; sample <= limit; sample++) {
    d = data[sample];
    for (coeff = 0; coeff < lag; coeff++)
        autoc[coeff] += d * data[sample+coeff];
}
```

  - After

```
for (coeff = 0; coeff < lag; coeff++) {
    d = 0.0;
    for (sample = 0; sample <= limit; sample++)
        d += data[sample] * data[sample+coeff];
    autoc[coeff] = d;
}
```

Verify speedup

- perf stat

```
$ perf stat ./flac -f -o /dev/null test.wav
[...]
 Performance counter stats for './flac -f -o /dev/null test.wav':

      20802.384992    task-clock (msec)         #    0.999 CPUs utilized
               704    context-switches          #    0.034 K/sec
                 0    cpu-migrations            #    0.000 K/sec
               177    page-faults               #    0.009 K/sec
    19,968,061,210    cycles                    #    0.960 GHz
    18,500,915,857    instructions              #    0.93  insns per cycle
     1,870,531,435    branches                  #   89.919 M/sec
        39,831,521    branch-misses             #    2.13% of all branches

      20.820567932 seconds time elapsed
```

perf report

```
# Samples: 88K of event 'cycles'
# Event count (approx.): 21093361393
#
# Overhead      Shared Object                                      Symbol
# ........  ..................  ..........................................
#
    23.97%  flac                FLAC__lpc_compute_autocorrelation
    12.05%  flac                FLAC__fixed_compute_best_predictor
    11.21%  flac                FLAC__lpc_compute_residual_from_qlp_coefficients
     8.17%  flac                FLAC__lpc_window_data
     7.39%  flac                FLAC__bitwriter_write_rice_signed_block
     7.20%  flac                precompute_partition_info_sums_
     5.33%  flac                FLAC__fixed_compute_residual
     4.85%  flac                FLAC__MD5Transform
     3.22%  flac                FLAC__crc16
     2.61%  flac                FLAC__MD5Accumulate
     2.24%  flac                format_input
     1.95%  flac                FLAC__stream_encoder_process
     1.63%  libc-2.20.so        memcpy
```

perf annotate

```
 0.00 : 4fc68:   vldr      s15, [pc, #136]
 0.05 : 4fc6c:   mov       lr, r5
 0.00 : 4fc70:   mov       r2, r0
 0.00 : 4fc74:   mov       r12, #0
19.54 : 4fc78:   vldmia    r2!, {s13}
20.09 : 4fc7c:   add       r12, r12, #1
19.89 : 4fc80:   vldmia    lr!, {s14}
20.03 : 4fc84:   cmp       r4, r12
20.07 : 4fc88:   vfma.f32  s15, s13, s14
 0.00 : 4fc8c:   bcs       4fc78
 0.04 : 4fc90:   vstmia    r6!, {s15}
 0.00 : 4fc94:   add       r5, r5, #4
 0.00 : 4fc98:   cmp       r6, r7
 0.00 : 4fc9c:   bne       4fc68
```

The twist

```
/*
 * this version tends to run faster because of better data
locality
 * ('data_len' is usually much larger than 'lag')
 */
```

- Probably true on some system
- Depends on result latency, cache size

Pipelining effects

- Result latency
- Multiple issue
- Out of order execution

Result latency

```
10.72 : 8498: vmul.f32  s1, s0, s0
44.92 : 849c: vmov.f32  s1, s1
11.42 : 84a0: vmov.f32  s2, s2
11.28 : 84a4: vmov.f32  s3, s3
10.75 : 84a8: vmov.f32  s4, s4
10.90 : 84ac: subs      r2, r2, #1
 0.00 : 84b0: bgt       8498
```

```
16.71 : 8510: vmul.f32  s4, s0, s0
17.01 : 8514: vmov.f32  s1, s1
16.71 : 8518: vmov.f32  s2, s2
16.87 : 851c: vmov.f32  s3, s3
16.09 : 8520: vmov.f32  s4, s4
16.62 : 8524: subs      r2, r2, #1
 0.00 : 8528: bgt       8510
```

Stalled cycles counted towards dependent instruction.

Multiple issue

```
33.15 : 85b0: ldr   r3, [sp]
 0.00 : 85b4: mov   r4, r5
34.26 : 85b8: add   r0, r0, r1
 0.00 : 85bc: orr   r6, r6, #1
32.59 : 85c0: subs  r2, r2, #1
 0.00 : 85c4: bgt   85b0
```

```
24.19 : 85d4: ldr   r3, [sp]
 0.00 : 85d8: mov   r4, r5
23.69 : 85dc: add   r0, r0, r1
27.27 : 85e0: orr   r6, r3, #1
 0.00 : 85e4: subs  r2, r2, #1
24.85 : 85e8: bgt   85d4
```

Samples fall on the first instruction in a multi-issue group.

Cache misses

```
10.43 : 85f4: ldr   r3, [r1], #64
68.09 : 85f8: bic   r1, r1, #131072
10.37 : 85fc: orr   r0, r0, r0
 0.00 : 8600: subs  r2, r2, #1
11.12 : 8604: bgt   85f4
```

```
25.13 : 862c: ldr   r3, [r1], #64
24.96 : 8630: bic   r1, r1, #64
24.65 : 8634: orr   r0, r0, r0
 0.00 : 8638: subs  r2, r2, #1
25.27 : 863c: bgt   862c
```

L1 miss in the load stalls the following instruction.

Other events

- Cache misses
- Mispredicted branches
- TLB misses
- CPU-specific events
- Software events
  - Page faults
  - Context switches

Resources

- Perf Wiki - Tutorial
  - https://perf.wiki.kernel.org/index.php/Tutorial
- ARM Information Center
  - http://infocenter.arm.com/

Questions?