Cycle Accurate Profiling With Perf

Paweł Moll <pawel.moll@arm.com>
The plan
The plan

- Hardware
- Linux perf
- Let’s hack!
Hardware
ARM CoreSight

- Source
- Bus
- Sink
Processor trace

- **Embedded Trace Macrocell**
  - Instructions
  - Data

- **Program Trace Macrocell**
  - Only branches

- **Bandwidth**
  - From 10Mbps to many Gbps per core

- **Non- (or low-) intrusive debug**
Sinks

- **Embedded Trace Buffer**
  - Dedicated, small SRAM
  - Flight recorder use case

- **Trace Port**
  - External analyser
  - Large buffer
  - External (high speed) pins

- **Embedded Trace Router**
  - Sinks data into main interconnect
  - Usually uses system DRAM
  - Consumes memory system bandwidth
ETM Protocol

- Highly compressed data
  - Generated against program memory
- Based on E/N atoms
  - eg. b1NEEEEO00 up to 16+1 instructions
- Branches
  - Only if not evident (eg. eg. B <imm>)
  - No address == previous address
  - Exceptions, instruction sets, processor state
- Synchronisation packets
- Data packets
- PTM protocol
  - One bit per conditional branch
Issues

- Requires memory contents for decompression
  - Multitasking OS
  - JIT engines
  - Self-modifying code (kernel runtime patching, kprobes, dynamic trace events)

- Parallel and out-of-order execution
Additional features

- Filtering
  - Address
  - CONTEXTID
  - VMID

- Trigerring
  - Address
  - DBG <imm>
  - Counter
  - Sequencer

- Timestamping
  - Correlation (synchronisation)
Linux perf
Linux perf framework

- PMU drivers
- Many use cases, eg. statistics
- Sampling profiler
  - Periodic PC (IP) sampling
  - Timer or PMU counter overflow interrupt
  - Typical sampling rate 1kHz (every 1ms)
Sampling profile

- Statistical approximation of a process
- Think analog/digital converter
- Shannon’s theorem: “If a function $x(t)$ contains no frequencies higher than $B$ cps, it is completely determined by giving its ordinates at a series of points spaced $1/(2B)$ seconds apart.”
CoreSight Linux framework

- Developed by Mathieu Poirier at Linaro
- Based on 2012 code from Code Aurora
- At v7 stage now (http://lwn.net/Articles/614232/)
- Control trace components via \texttt{sysfs}
  - Enable sink, enable source, dump buffer contents
- Separate decoders
- Full series at http://git.linaro.org/kernel/coresight.git
**Intel PT**

- “*an exciting new feature coming in future processors*” (2013)
- Integrating with perf
  - Auxiliary buffers
  - Decoder integrated with user space tool
- Kernel portions at v4 stage now (http://lwn.net/Articles/609010/)
- Full series at
  https://github.com/virtuoso/linux-perf/tree/intel_pt
Let’s hack!
Particularly pathologic example

- Rotate JPEG file

/ # time taskset 4 ./gm convert -rotate 90 in.jpg out.jpg
real 0m 0.01s
user 0m 0.01s
sys 0m 0.00s
Can it go faster?

// # time perf record -F 1000 -e cpu-clock \
// taskset 4 ./gm convert -rotate 90 in.jpg out.jpg
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.001 MB perf.data (~50 samples) ]
real 0m 0.27s
user 0m 0.14s
sys 0m 0.13s

■ It was:
real 0m 0.01s
user 0m 0.01s
sys 0m 0.00s
# Samples: 17 of event 'cpu-clock'
# Event count (approx.): 17000000
#
# Overhead  Command  Shared Object  Symbol
#        ..........  ............  ......................  .........................
# 17.65%    gm      [kernel.kallsyms]  [k] filemap_map_pages
5.88%      taskset  [kernel.kallsyms]  [k] filemap_map_pages
5.88%      gm      gm  LocaleCompare
5.88%      gm      gm  forward_DCT_float
5.88%      gm      gm  encode_mcu_huff
5.88%      gm      gm  ycc_rgb_convert
5.88%      gm      gm  decode_mcu_DC_first
5.88%      gm      gm  decode_mcu_AC_refine
<table>
<thead>
<tr>
<th>Percentage</th>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.88%</td>
<td>gm</td>
<td>jpeg_fdct_16x16</td>
</tr>
<tr>
<td>5.88%</td>
<td>gm</td>
<td>_IO_link_in</td>
</tr>
<tr>
<td>5.88%</td>
<td>gm</td>
<td>malloc</td>
</tr>
<tr>
<td>5.88%</td>
<td>gm</td>
<td>strncpy</td>
</tr>
<tr>
<td>5.88%</td>
<td>gm</td>
<td>[kernel.kallsyms] lock_acquire</td>
</tr>
<tr>
<td>5.88%</td>
<td>gm</td>
<td>[kernel.kallsyms] lock_release</td>
</tr>
<tr>
<td>5.88%</td>
<td>gm</td>
<td>[kernel.kallsyms] unmap_single_vma</td>
</tr>
</tbody>
</table>
Let’s have a closer look…

- Cortex-A7 ETM 3.5
- Instructions only
- Cycle accurate
- Captured with DStream & ARM DS-5
- 10MB of binary trace data
What can we see there?

- Decoded into text format
- 890MB file
- What to look for:

**ELF Header:**

Entry point address: 0x96dd

- 8360: 000096dd 0 FUNC GLOBAL DEFAULT 6 _start
- 8845: 0012ac44 0 FUNC GLOBAL DEFAULT 9 _fini
Here we go

2859224 S:0x8000E4CC E28DD00C 0 ADD sp,sp,#0xc  ret_to_user_from_irq
2859225 S:0x8000E4D0 E1B0F00E 1 MOVS pc,lr  ret_to_user_from_irq
   Return from exception
   Timestamp: 1878241989137
S:0x000096DC F04F0B00 29 MOV r11,#0  <Unknown>
   Exception: PREFETCH_ABORT (11)
2859227 S:0xFFFF000C EA000443 21 B  PRRR+16027512 ; 0xFFFF1120  <Unknown>
   Timestamp: 1878241989138
2859228 S:0xFFFF1120 E24EE004 18 SUB lr,lr,#4  <Unknown>
2859229 S:0xFFFF1124 E88D4001 3 STM sp,r0,lr  <Unknown>
Here we go again

2878794 S:0x8000E4D0 E1B0F00E 1 MOV pc,lr ret_to_user_from_irq
Return from exception
Timestamp: 1878241990817

2878795 S:0x000096DC F04F0B00 175 MOV r11,#0 <Unknown>
2878796 S:0x000096E0 F04F0E00 40 MOV lr,#0 <Unknown>
2878797 S:0x000096E4 BC02 4 POP r1 <Unknown>
2878798 S:0x000096E6 466A 1 MOV r2,sp <Unknown>

[...]
2878804 S:0x000096F6 4B04 1 LDR r3,[pc,#16] ; [0x9708] = 0xB114687C <Unknown>
2878805 S:0x000096F8 F0DAFDF4 0 BL __libc_start_main ; 0xE42E4 <Unknown>

S:0x000E42E4 E92D45F0 324 PUSH r4-r8,r10,lr __libc_start_main
Exception: PREFETCH_ABORT (11)

2878807 S:0xFFFF000C EA000443 21 B PRRR+16027512 ; 0xFFFF1120 <Unknown>
Going down

9747820 S:0x0012AC44 E91D4008 153 PUSH r3,lr <Unknown>
9747821 S:0x0012AC48 E8BD8008 2 POP r3,pc <Unknown>
9747822 S:0x000E9AA2 E7BF 9 B __run_exit_handlers+28 ; 0xE9A24
9747823 S:0x000E9A24 6873 3 LDR r3,[r6,#4] __run_exit_handlers
9747824 S:0x000E9A26 EB061403 3 ADD r4,r6,r3,LSL #4 __run_exit Handlers
9747825 S:0x000E9A2A B173 1 CBZ r3,__run_exit_handlers+66 ; 0xE9A4A
The end of the process

9747990 S:0x000FD536 4C12  2 LDR r4,[pc,#72] ; [0xFD580] = 0x64C55B39 _Exit
9747991 S:0x000FD538 E004  0 B _Exit+24 ; 0xFD544 _Exit
9747992 S:0x000FD544 4618  1 MOV r0,r3 _Exit
9747993 S:0x000FD546 F04F0CF8 1 MOV r12,#0xf8 _Exit
9747994 S:0x000FD54A F7E7F9C9 0 BL __libc_do_syscall ; 0xE48E0 _Exit
9747995 S:0x000E48E0 B580 1 PUSH r7,lr __libc_do_syscall
9747996 S:0x000E48E2 4667 2 MOV r7,r12 __libc_do_syscall
9747997 S:0x000E48E4 DF00 1 SVC #0x0 __libc_do_syscall

Exception: SUPERVISOR_CALL (10)

- #0xf8 is __NR_exit_group
Focus on the task

- It’s all interesting...
- ...but the goal is to see a cycle accurate profile of the process
- Limit data scope
  - Filter out kernel addresses (or cheat using entry/exit points)
  - Filter out other contexts (or cheat by protecting CPU)
  - Collate migrated data (or cheat by setting affinity)
  - Generate memory map with DSOs (or cheat by linking statically)
- Convert into perf data stream
perf.data

- Starts with header
- Description of all events
- Followed by data records ...
  - selection of samples
  - by default: PERF_SAMPLE_IP, PERF_SAMPLE_TID, PERF_SAMPLE_TIME, PERF_SAMPLE_PERIOD
  - generated on every timer/counter interrupt
- ...interleaved with system information
  - eg. PERF_RECORD_MMAP, PERF_RECORD_COMM, PERF_RECORD_EXIT, PERF_RECORD_FORK
$ perf report -D
[...]
0x1d0 [0x28]: event: 9
.
... raw event: size 40 bytes
  0000: 09 00 00 00 01 00 28 00 54 fd 05 80 00 00 00 00 .......(T....... 0010: 3b 08 00 00 3b 08 00 00 46 fb 39 91 dd 46 00 00 ;...;...F.9..F. 0020: 40 42 0f 00 00 00 00 00 @B......
.
77917438212934 0x1d0 [0x28]: PERF_RECORD_SAMPLE(IP, 1):
  2107/2107: 0x8005fd54 period: 1000000 addr: 0
... thread: gm:2107
....... dso: [kernel.kallsyms]
The Hack

- Replace data records with trace-based ones
  - reducing number of samples, from 40 to 24 bytes per record (attribute modification needed)
  - generating multiple samples, one per cycle used (175 samples if instruction took 175 cycles to execute)

- 192MB big perf.data
The Hack, cont.

0x1f0 [0x18]: event: 9
  ... raw event: size 24 bytes
  0000: 09 00 00 00 02 00 18 00 dc 96 00 00 00 00 00 00 ................
  0010: 3b 08 00 00 3b 08 00 00 ;...;...

0x1f0 [0x18]: PERF_RECORD_SAMPLE(IP, 2):
  2107/2107: 0x96dc period: 1 addr: 0
  ... thread: :2107:2107
  ....... dso: <not found>

0x208 [0x18]: event: 9
  ... raw event: size 24 bytes
  0000: 09 00 00 00 02 00 18 00 dc 96 00 00 00 00 00 00 ................
  0010: 3b 08 00 00 3b 08 00 00 ;...;...

0x208 [0x18]: PERF_RECORD_SAMPLE(IP, 2):
  2107/2107: 0x96dc period: 1 addr: 0
  ... thread: :2107:2107
Result

# Samples: 8M of event 'cycles'
# Event count (approx.): 8012563
#
# Overhead  Command  Shared Object  Symbol
# ........  ........  ................  ..................................
#
<table>
<thead>
<tr>
<th>Overhead</th>
<th>Command</th>
<th>Shared Object</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.98%</td>
<td>gm</td>
<td>gm</td>
<td>[.] jpeg_idct_islow</td>
</tr>
<tr>
<td>7.25%</td>
<td>gm</td>
<td>gm</td>
<td>[.] strncpy</td>
</tr>
<tr>
<td>6.60%</td>
<td>gm</td>
<td>gm</td>
<td>[.] jpeg_fdct_16x16</td>
</tr>
<tr>
<td>4.40%</td>
<td>gm</td>
<td>gm</td>
<td>[.] encode_mcu_huff</td>
</tr>
<tr>
<td>4.37%</td>
<td>gm</td>
<td>gm</td>
<td>[.] decode_mcu_AC_refine</td>
</tr>
<tr>
<td>4.25%</td>
<td>gm</td>
<td>gm</td>
<td>[.] LocaleCompare</td>
</tr>
<tr>
<td>3.73%</td>
<td>gm</td>
<td>gm</td>
<td>[.] rgb_ycc_convert</td>
</tr>
<tr>
<td>3.25%</td>
<td>gm</td>
<td>gm</td>
<td>[.] _int_free</td>
</tr>
<tr>
<td>Function Call</td>
<td>CPU Time</td>
<td>Usage Percentage</td>
<td>Source Code</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------</td>
<td>------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>memset</td>
<td>3.17%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
<tr>
<td>jpeg_gen_optimal_table</td>
<td>3.15%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
<tr>
<td>malloc</td>
<td>3.01%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
<tr>
<td>__int_malloc</td>
<td>2.63%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
<tr>
<td>forward_DCT_float</td>
<td>2.63%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
<tr>
<td>ycc_rgb_convert</td>
<td>2.59%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
<tr>
<td>jpeg_fdct_float</td>
<td>2.43%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
<tr>
<td>__pthread_mutex_unlock_usercnt</td>
<td>2.29%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
<tr>
<td>__memcpy_neon</td>
<td>2.25%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
<tr>
<td>pthread_mutex_lock</td>
<td>2.19%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
<tr>
<td>ReadJPEGImage</td>
<td>1.74%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
<tr>
<td>encode_mcu_gather</td>
<td>1.72%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
<tr>
<td>WriteJPEGImage</td>
<td>1.61%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
<tr>
<td>vfprintff</td>
<td>1.39%</td>
<td>gm gm</td>
<td>[.]</td>
</tr>
</tbody>
</table>
### Result, cont

<table>
<thead>
<tr>
<th>Function</th>
<th>GM</th>
<th>GM</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>consume_data</td>
<td>gm</td>
<td>gm</td>
<td>1.32%</td>
</tr>
<tr>
<td>forward_DCT</td>
<td>gm</td>
<td>gm</td>
<td>1.21%</td>
</tr>
<tr>
<td>RegisterMagickInfo</td>
<td>gm</td>
<td>gm</td>
<td>1.06%</td>
</tr>
<tr>
<td>UnregisterMagickInfo</td>
<td>gm</td>
<td>gm</td>
<td>0.88%</td>
</tr>
<tr>
<td>decode_mcu_AC_first</td>
<td>gm</td>
<td>gm</td>
<td>0.87%</td>
</tr>
<tr>
<td>strlen</td>
<td>gm</td>
<td>gm</td>
<td>0.81%</td>
</tr>
<tr>
<td>SyncCacheNexus</td>
<td>gm</td>
<td>gm</td>
<td>0.73%</td>
</tr>
<tr>
<td>strcpy</td>
<td>gm</td>
<td>gm</td>
<td>0.64%</td>
</tr>
<tr>
<td>jpeg_fill_bit_buffer</td>
<td>gm</td>
<td>gm</td>
<td>0.53%</td>
</tr>
<tr>
<td>decode_mcu_DC_first</td>
<td>gm</td>
<td>gm</td>
<td>0.49%</td>
</tr>
<tr>
<td>_IO_default_xsputn</td>
<td>gm</td>
<td>gm</td>
<td>0.40%</td>
</tr>
<tr>
<td>_init</td>
<td>gm</td>
<td>gm</td>
<td>0.35%</td>
</tr>
<tr>
<td>jpeg_make_d DerivedTbl</td>
<td>gm</td>
<td>gm</td>
<td>0.34%</td>
</tr>
<tr>
<td>DestroyMagickInfo</td>
<td>gm</td>
<td>gm</td>
<td>0.33%</td>
</tr>
<tr>
<td>Percentage</td>
<td>Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.33%</td>
<td>QueryColorDatabase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.32%</td>
<td>GetGeometry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.31%</td>
<td>SetNexus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.31%</td>
<td>free</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.29%</td>
<td>__strchrnul</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.27%</td>
<td>____strtod_l_internal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.27%</td>
<td>.divsi3_skip_div0_test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.24%</td>
<td>SetCacheNexus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.23%</td>
<td>access_virt_barray</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.23%</td>
<td>compress_output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.22%</td>
<td>____strtol_l_internal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.21%</td>
<td>UnlockSemaphoreInfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.21%</td>
<td>AcquireCacheNexus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.21%</td>
<td>format_message</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Result, cont

[...]
Result, cont

0.00% gm gm
0.00% gm gm
0.00% gm gm
0.00% gm gm
0.00% gm gm
0.00% gm gm
0.00% gm gm
0.00% gm gm
0.00% gm gm
0.00% gm gm
0.00% gm gm
0.00% gm gm
0.00% gm gm

[.] ___fini_from_thumb
[.] jpeg_destroy_compress
[.] __feupdateenv
[.] IdentityAffine
[.] __dcgettext
[.] _setjmp
[.] DestroyMagickResources
[.] jpeg_free_small
[.] fprintf
[.] DestroySemaphore
[.] jpeg_mem_term
[.] start_pass_downsample
[.] malloc_info
[.] __stpcpy
Result, cont

- 616 lines of report
- `perf annotate` works as well!
Summary
Summary

- Proof of concept
- Can help with pathological cases
- Scaling issues
- Powerful but need to be “civilised”
- Nearest future
  - Drivers in mainline
  - perf tool decoder integration
Thank You

The trademarks featured in this presentation are registered and/or unregistered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.