Linux perf
for Qt developers

Milian Wolff / KDAB
Agenda

- Setup
- Benchmarking
- Profiling
- Tracing
- Scripting
Setup

- Hardware
- Linux Kernel Prerequisites
- Building User-space perf
- Cross-compiling
- Permissions
Hardware

- Hardware performance counters
- Working PMU
Linux Kernel Prerequisites

$ uname -r  # should be at least 3.7
4.7.1-1-ARCH
$ zgrep PERF /proc/config.gz
CONFIG_HAVE_PERF_EVENTS=y
CONFIG_PERF_EVENTS=y
CONFIG_HAVE_PERF_USER_STACK_DUMP=y
CONFIG_HAVE_PERF_REGS=y
...
Building User-space perf

```bash
git clone -b milian/perf https://github.com/milianw/linux.git
# or, once all my patches got accepted upstream, use:
# git clone -b perf/core \
#  git://git.kernel.org/pub/scm/linux/kernel/git/acme/linux

cd linux/tools/perf
export CC=gcc # clang is not supported
make
```
Auto-detecting system features:

...  dwarf: [ on ]  # for symbol resolution
...  dwarf_getlocations: [ on ]  # for symbol resolution
...  glibc: [ on ]
...  gtk2: [ on ]
...  libaudit: [ on ]  # for syscall tracing
...  libbfd: [ on ]  # for symbol resolution
...  libelf: [ on ]  # for symbol resolution
...  libnuma: [ on ]
...  numa_num_possible_cpus: [ on ]
...  libperl: [ on ]  # for perl bindings
...  libpython: [ on ]  # for python bindings
...  libssl: [ on ]  # for TUI
...  libcrypto: [ on ]  # for JITed probe points
...  libunwind: [ on ]  # for unwinding
...  libdw-dwarf-unwind: [ on ]  # for unwinding
...  zlib: [ on ]
...  lzma: [ on ]
...  get_cpuid: [ on ]
...  bpf: [ on ]
Cross-compiling

```make
make prefix=somepath ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu-
```

Common pitfalls:

- **CC** must not contain any flags
- **CFLAGS** is ignored, use **EXTRA_CFLAGS**
- **prefix** path ignored for include and library paths
- Dependency issues:
  ```
  linux/tools/build/feature/test-$FEATURE.make.output
  ```
Permissions

#!/bin/bash
sudo mount -o remount,mode=755 /sys/kernel/debug
sudo mount -o remount,mode=755 /sys/kernel/debug/tracing
echo "0" | sudo tee /proc/sys/kernel/kptr_restrict
echo "-1" | sudo tee /proc/sys/kernel/perf_event_paranoïd
sudo chown root:tracing /sys/kernel/debug/tracing/uprobe_events
sudo chmod g+rw /sys/kernel/debug/tracing/uprobe_events
Benchmarking

- Be scientific!
- Take variance into account
- Compare before/after measurements
$ perf stat -r 5 -o baseline.txt -- ./ex_branches
$ cat baseline.txt
Performance counter stats for './ex_branches' (5 runs):

807.951072 task-clock:u (msec) # 0.999 CPUs utilized ( +- 1.97% )
  0 context-switches:u   # 0.000 K/sec
  0 cpu-migrations:u    # 0.000 K/sec
  520 page-faults:u     # 0.643 K/sec ( +- 0.15% )
2,487,366,239 cycles:u   # 3.079 GHz ( +- 1.97% )
1,484,737,283 instructions:u # 0.60 insn per cycle ( +- 0.00% )
  329,602,843 branches:u # 407.949 M/sec ( +- 0.00% )
  80,476,858 branch-misses:u # 24.42% of all branches ( +- 0.06% )

0.808952447 seconds time elapsed ( +- 1.97% )
Kernel vs. Userspace

Use event modifiers to separate domains:

```
$ perf stat -r 5 --event=cycles:{k,u} -- ./ex_qdatetime

Performance counter stats for './ex_qdatetime' (5 runs):

  13,337,722 cycles:k                      (  +  3.82% )
  9,745,474  cycles:u                      (  +  1.58% )

  0.008018321 seconds time elapsed        (  +  4.02% )
```

See `man perf list` for more.
$ perf list
List of pre-defined events (to be used in -e):

- branch-misses [Hardware event]
- cache-misses [Hardware event]
- cpu-cycles OR cycles [Hardware event]
- instructions [Hardware event]
- ref-cycles [Hardware event]
...

- alignment-faults [Software event]
- context-switches OR cs [Software event]
- page-faults OR faults [Software event]
...

- sched:sched_stat_sleep [Tracepoint event]
- sched:sched_stat_iowait [Tracepoint event]
- sched:sched_stat_runtime [Tracepoint event]
...

- syscalls:sys_enter_futex [Tracepoint event]
- syscalls:sys_exit_futex [Tracepoint event]
...
Qt TestLib

Count cycles in QBENCHMARKs:

```
$ ./my_qt_benchmark -perf -iterations 100
RESULT : BenchQDateTime::benchCurrentDateTime():
  25,346.54 CPU cycles per iteration (total: 2,534,654, iterations: 100)
RESULT : BenchQDateTime::benchCurrentDateTimeUtc():
  777.86 CPU cycles per iteration (total: 77,786, iterations: 100)
```
Qt TestLib

Count instructions:

$ ./my_qt_benchmark -perf -perfcounter instructions -iterations 100
RESULT : BenchQDateTime::benchCurrentDateTime()
  62,836.76 instructions per iteration (total: 6,283,676, iterations: 100)
RESULT : BenchQDateTime::benchCurrentDateTimeUtc()
  1,078.75 instructions per iteration (total: 107,876, iterations: 100)

Enable multiple perf event counters: codereview.qt-project.org/#/c/149580/
Qt TestLib

Available performance counters:

```bash
$ ./my_qt_benchmark -perfcounterlist
The following performance counters are available:
  alignment-faults          [software]
  branch-instructions       [hardware]
  branch-load-misses        [cache]
  branch-loads              [cache]
  branch-mispredicts        [cache]
  branch-misses             [hardware]
  branch-predicts           [cache]
  branch-read-misses        [cache]
  branch-reads              [cache]
  branches                  [hardware]
  bus-cycles                [hardware]
  cache-misses              [hardware]
  cache-references          [hardware]
  context-switches          [software]
  cpu-clock                 [software]
  ...
```
Profiling

- CPU profiling
- Sleep-time profiling
$ perf top
Samples: 12K of event 'cycles:ppp', Event count (approx.): 5456372201

<table>
<thead>
<tr>
<th>Overhead</th>
<th>Shared Object</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.11%</td>
<td>libQt5Core.so.5.7.0</td>
<td>[...] QHashData::nextNode</td>
</tr>
<tr>
<td>5.08%</td>
<td>libQt5Core.so.5.7.0</td>
<td>[...] operator==</td>
</tr>
<tr>
<td>2.90%</td>
<td>libQt5Core.so.5.7.0</td>
<td>[...] 0x0000000000012f0d1</td>
</tr>
<tr>
<td>2.33%</td>
<td>libQt5DBus.so.5.7.0</td>
<td>[...] 0x000000000002281f</td>
</tr>
<tr>
<td>1.62%</td>
<td>libQt5DBus.so.5.7.0</td>
<td>[...] 0x0000000000022810</td>
</tr>
</tbody>
</table>

...
Statistical Profiling

Sampling the **call stack** is crucial!
Unwinding and Call Stacks

- frame pointers (fp)
- debug information (dwarf)
- Last Branch Record (lbr)
Recommendation

- On embedded: enable frame pointers
- On the desktop: rely on DWARF
- On Intel: play with LBR
perf record

Profile new application and its children:

$ perf record --call-graph dwarf -- ./lab_mandelbrot -b 5
[ perf record: Woken up 256 times to write data ]
[ perf record: Captured and wrote 64.174 MB perf.data (7963 samples) ]
perf record

Attach to running process:

$ perf record --call-graph dwarf --pid $(pidof ...)
# wait for some time, then quit with CTRL + C
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 3.904 MB perf.data (70 samples) ]
perf record

Profile whole system for some time:

```sh
$ perf record -a -- sleep 5
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 1.498 MB perf.data (2731 samples) ]
```
perf report
perf report

Top-down inclusive cost report:

$ perf report

Samples: 8K of event 'cycles:ppp', Event count (approx.): 8164367769

Children             Self   Command               Shared Object      Symbol
-  93.67%  31.76%  lab_mandelbrot         lab_mandelbrot [.] main
  -  72.22%  main
    + 28.42%  hypot
      __hypot_finite
      19.87%  __muldc3
      3.45%  __muldc3@plt
      2.19%  cabs@plt
    + 1.85%  QColor::rgb
      1.61%  QImage::width@plt
      1.26%  QImage::height@plt
      0.97%  QColor::fromHsvF
    + 0.90%  QApplicationPrivate::init
      0.66%  QImage::setPixel
    + 21.44%  _start
      + 83.34%  0.00%  lab_mandelbrot  libc-2.24.so  [.] __libc_start_main
      + 83.33%  0.00%  lab_mandelbrot  lab_mandelbrot [.] _start

...
$ perf report --no-children

Samples: 8K of event 'cycles:ppp', Event count (approx.): 8164367769

Overhead Command Shared Object Symbol
-  31.76% lab_mandelbrot lab_mandelbrot [.] main
  - main
    - __libc_start_main
      _start
-  23.31% lab_mandelbrot libm-2.24.so [.] __hypotFinite
  - __hypotFinite
    -  22.56% hypot
      main
        __libc_start_main
          _start
-  23.04% lab_mandelbrot libgcc_s.so.1 [.] __muldc3
  - __muldc3
    + main
-   5.90% lab_mandelbrot libm-2.24.so [.] hypot
  + hypot
  ...

Bottom-up self cost report:
$ perf report --no-children -s dso,sym,srcline

Samples: 8K of event 'cycles:ppp', Event count (approx.): 8164367769
  Overhead  Shared Object  Symbol  Source:Line
-  7.82%  lab_mandelbrot  main  mandelbrot.h:41
  + main
-  7.79%  libgcc_s.so.1  __muldc3  libgcc2.c:1945
    __muldc3
    main
    __libc_start_main
    _start
-  7.46%  lab_mandelbrot  main  complex:1326
  - main
    + __libc_start_main
-  6.94%  libgcc_s.so.1  __muldc3  libgcc2.c:1944
    __muldc3
    main
    __libc_start_main
    _start
...
$ perf report --no-children -s dso,sym,srcline -g address

Samples: 8K of event 'cycles:ppp', Event count (approx.): 8164367769

<table>
<thead>
<tr>
<th>Overhead</th>
<th>Shared Object</th>
<th>Symbol</th>
<th>Source:Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.82%</td>
<td>lab_mandelbrot</td>
<td>[.] main</td>
<td>mandelbrot.h:41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>_libc_start_main +241</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>_start +4194346</td>
<td></td>
</tr>
<tr>
<td>2.84%</td>
<td>main mandelbrot.h:41</td>
<td>[.] main</td>
<td>mandelbrot.h:41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>_libc_start_main +241</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>_start +4194346</td>
<td></td>
</tr>
<tr>
<td>2.58%</td>
<td>main mandelbrot.h:41</td>
<td>[.] main</td>
<td>mandelbrot.h:41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>_libc_start_main +241</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>_start +4194346</td>
<td></td>
</tr>
<tr>
<td>2.01%</td>
<td>main mandelbrot.h:41</td>
<td>[.] main</td>
<td>mandelbrot.h:41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>_libc_start_main +241</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>_start +4194346</td>
<td></td>
</tr>
<tr>
<td>7.79%</td>
<td>libgcc_s.so.1</td>
<td>[.] __muldc3</td>
<td>libgcc2.c:1945</td>
</tr>
<tr>
<td>+ 3.93%</td>
<td>__muldc3</td>
<td>libgcc2.c:1945</td>
<td></td>
</tr>
<tr>
<td>+ 3.72%</td>
<td>__muldc3</td>
<td>libgcc2.c:1945</td>
<td></td>
</tr>
<tr>
<td>7.46%</td>
<td>lab_mandelbrot</td>
<td>[.] main</td>
<td>complex:1326</td>
</tr>
<tr>
<td></td>
<td></td>
<td>__libc_start_main +241</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>_start +4194346</td>
<td></td>
</tr>
<tr>
<td>4.65%</td>
<td>main complex:1326</td>
<td>[.] main</td>
<td>complex:1326</td>
</tr>
<tr>
<td></td>
<td></td>
<td>__libc_start_main +241</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>_start +4194346</td>
<td></td>
</tr>
<tr>
<td>2.81%</td>
<td>main complex:1326</td>
<td>[.] main</td>
<td>complex:1326</td>
</tr>
<tr>
<td></td>
<td></td>
<td>__libc_start_main +241</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>_start +4194346</td>
<td></td>
</tr>
</tbody>
</table>

...
Configure default output format:

[report]
children = false
sort_order = dso,sym,srcline

[call-graph]
record-mode = dwarf
print-type = graph
order = caller
sort-key = address
Flame Graphs

```
perf script report stackcollapse | flamegraph.pl > graph.svg
```
QML profiling
QML profiling

JIT breaks backtraces:

```
$ perf record --call-graph dwarf -- qml fibonacci.qml
$ perf report
Failed to open /tmp/perf-22849.map, continuing without symbols
Samples: 7K of event 'cycles:ppp', Event count (approx.): 7569496308
  Overhead  Shared Object  Symbol
-  12.42%  libQt5Qml.so.5.7.0  [.] 0x000000000001a06ca
  -  6.44%  0x7f6f6db7d213
    QV4::Runtime::callActivationProperty +117
    - QV4::ExecutionContext::getPropertyAndBase +1281
      -  3.54%  QV4::QmlContextWrapper::get +85
        QV4::Object::internalGet +173
          0x1a06ca
      -  2.89%  QV4::QmlContextWrapper::get +214
        QV4::Object::internalGet +173
          0x1a06ca
      -  5.98%  0x7f6f6db7d313
        QV4::Runtime::callActivationProperty +117
        - QV4::ExecutionContext::getPropertyAndBase +1281
          +  3.41%  QV4::QmlContextWrapper::get +85
          +  2.57%  QV4::QmlContextWrapper::get +214
...```
QML profiling

LBR unwinder can break the JIT barrier:

```
$ perf record --call-graph lbr -- qml fibonacci.qml
$ perf report
  Failed to open /tmp/perf-22849.map, continuing without symbols
  Samples: 7K of event 'cycles:ppp', Event count (approx.): 7410334159
   Overhead  Shared Object                Symbol
  -  11.87%  libQt5Qml.so.5.7.0            [.] 0x00000000001a06ca
  -  2.98%  0x7fa5f71d7310
  -  1.87%  QV4::Runtime::callActivationProperty +287
             0x1c1618
  -  1.64%  0x7fa5f71d7210
  -  0.89%  QV4::Runtime::callActivationProperty +112
             QV4::ExecutionContext::getPropertyAndBase +1278
  -  0.75%  QV4::Runtime::callActivationProperty +287
             0x1c1618
  -  0.58%  0x7fa5f71d7210
  -  0.53%  QV4::Runtime::callActivationProperty +112
             QV4::ExecutionContext::getPropertyAndBase +127
  + 1.10%  QV4::Runtime::callActivationProperty +112
  + 2.15%  QV4::Runtime::callActivationProperty +287
...```

QML profiling

Annotate the QML symbols:

```bash
$ QV4_PROFILE_WRITE_PERF_MAP=1 perf record --call-graph lbr -- qml myapp.qml
$ perf report

Samples: 7K of event 'cycles:ppp', Event count (approx.): 7490988495
  Overhead  Shared Object          Symbol
-  12.08%  libQt5Qml.so.5.7.0      [.] 0x000000000001a06ca
-  3.58%    fibonacci +656
  -  2.05%  QV4::Runtime::callActivationProperty +287
    - 0x1c1618
    -  1.71%    fibonacci +400
      -  1.04%  QV4::Runtime::callActivationProperty +112
        - QV4::ExecutionContext::getPropertyAndBase +1278
          -  0.62%  QV4::QmlContextWrapper::get +82
            QV4::Object::internalGet +168
            0x1a0670
          -  0.67%  QV4::Runtime::callActivationProperty +287
            - 0x1c1618
              0.60%    fibonacci +400
              + 1.53%  QV4::Runtime::callActivationProperty +112
```

Ov...
On-CPU vs. Off-CPU

PMU events are only measured while application is running.

How to measure sleep and IO time?
### Scheduler Statistics

**perf list sched_stat**

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

<table>
<thead>
<tr>
<th>Event</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>sched:sched_stat_blocked</td>
<td>Tracepoint event</td>
</tr>
<tr>
<td>sched:sched_stat_iowait</td>
<td>Tracepoint event</td>
</tr>
<tr>
<td>sched:sched_stat_runtime</td>
<td>Tracepoint event</td>
</tr>
<tr>
<td>sched:sched_stat_sleep</td>
<td>Tracepoint event</td>
</tr>
<tr>
<td>sched:sched_stat_wait</td>
<td>Tracepoint event</td>
</tr>
</tbody>
</table>
Sleep-time Profiling

```bash
#!/bin/bash

echo 1 | sudo tee /proc/sys/kernel/sched_schedstats

perf record \n    --event sched:sched_stat_sleep/call-graph=fp/ \n    --event sched:sched_process_exit/call-graph=fp/ \n    --event sched:sched_switch/call-graph=dwarf/ \n    --output perf.data.raw @

echo 0 | sudo tee /proc/sys/kernel/sched_schedstats

perf inject --sched-stat --input perf.data.raw --output perf.data
```
$ perf-sleep-record ./ex_sleep
$ perf report

Samples: 24 of event 'sched:sched_switch', Event count (approx.): 8883195296
Overhead Trace output
- 100.00% ex_sleep:24938 [120] S ==> swapper/7:0 [120]
  - 90.07% main main.cpp:10
    QThread::sleep +11
    0x1521ed
    __nanosleep .:0
    entry_SYSCALL_64_fastpath entry_64.o:0
    sys_nanosleep +18446744071576748154
    hrtimer_nanosleep +18446744071576748225
    do_nanosleep hrtimer.c:0
    schedule +18446744071576748092
    __schedule core.c:0
+ 9.02% main main.cpp:11
+ 0.91% main main.cpp:6
Cross-machine Reporting

When recording machine has symbols available:

# on first machine:
$ perf record ...
$ perf archive

Now please run:

$ tar xvf perf.data.tar.bz2 -C ~/.debug

wherever you need to run 'perf report' on.

# on second machine:
$ rsync machine1:path/to/perf.data{,.tar.bz2} .
$ tar xf perf.data.tar.bz2 -C ~/.debug
$ perf report
Cross-machine Reporting

When reporting machine has symbols available:

# on first machine:
$ perf record ...

# on second machine:
$ rsync machine1:path/to/perf.data .
$ perf report --symfs /path/to/sysroot
Tracing

- Syscalls
- Kernel Events
- User Defined Tracepoints
perf top (again)
Count syscalls per application

<table>
<thead>
<tr>
<th>Overhead</th>
<th>Samples</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>32.14%</td>
<td>353935</td>
<td>Xorg</td>
</tr>
<tr>
<td>27.25%</td>
<td>300076</td>
<td>skype</td>
</tr>
<tr>
<td>21.05%</td>
<td>231800</td>
<td>QXcbEventReader</td>
</tr>
<tr>
<td>6.46%</td>
<td>71178</td>
<td>plasmashell</td>
</tr>
<tr>
<td>2.65%</td>
<td>29135</td>
<td>owncloud</td>
</tr>
<tr>
<td>1.70%</td>
<td>18680</td>
<td>baloo_file_extr</td>
</tr>
<tr>
<td>1.48%</td>
<td>16260</td>
<td>kwin_x11</td>
</tr>
<tr>
<td>0.93%</td>
<td>10237</td>
<td>yakuake</td>
</tr>
<tr>
<td>0.75%</td>
<td>8259</td>
<td>kate</td>
</tr>
<tr>
<td>0.71%</td>
<td>7861</td>
<td>kmail</td>
</tr>
<tr>
<td>0.70%</td>
<td>7759</td>
<td>cc1plus</td>
</tr>
<tr>
<td>0.63%</td>
<td>6909</td>
<td>iceccd</td>
</tr>
<tr>
<td>0.63%</td>
<td>6885</td>
<td>g++</td>
</tr>
</tbody>
</table>

$ perf top -e raw_syscalls:sys_enter -m 50M -s comm --show-nr-samples
Samples: 2M of event 'raw_syscalls:sys_enter', Event count (ca.): 1101332
perf trace

Find slow syscalls:

$ perf trace --duration=100

767.271 (18446744073709.457 ms): mysqld/3321 poll(ufds: 0x7ff1f791bab0, nfds: 777.559 (674.075 ms): kded5/1870 poll(ufds: 0x241c940, nfds: 12, timeout_msec 778.634 (766.826 ms): plasmashell/1914 poll(ufds: 0x4368550, nfds: 10, timeout_msec 779.100 (266.541 ms): owncloud/1958 poll(ufds: 0x1146720, nfds: 5, timeout_msec 777.678 (263.587 ms): kate/5107 poll(ufds: 0x57737e0, nfds: 17, timeout_msec 791.292 (142.024 ms): mysqlld/2324 futex(uaddr: 0x7ff205bc411c, op: WAIT|PRI 813.918 (500.537 ms): chromium/2586 poll(ufds: 0x36fe3ccca1e0, nfds: 4, timeout_msec 823.785 (500.079 ms): mysqlld/2252 io_gete events(ctx_id: 140677667425336, m 892.293 (349.913 ms): chromium/29754 futex(uaddr: 0x7ffcba137794, op: WAIT B 931.227 (18446744073709.219 ms): mysqlld/3321 poll(ufds: 0x7ff1f791bab0, nfds 940.429 (500.080 ms): mysqlld/2244 io_gete events(ctx_id: 140677674315776, m 940.600 (139.214 ms): mysqlld/2324 futex(uaddr: 0x7ff205bc411c, op: WAIT|PRI 994.984 (500.126 ms): mysqlld/2253 io_gete events(ctx_id: 140677674233856, min_ ...
Aggregated syscall overhead:

```
$ perf trace -s -- ./lab_mandelbrot_concurrent -b
...

lab_mandelbrot_ (1594), 9174 events, 25.7%

<table>
<thead>
<tr>
<th>syscall</th>
<th>calls</th>
<th>total (msec)</th>
<th>min (msec)</th>
<th>avg (msec)</th>
<th>max (msec)</th>
<th>stddev (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>futex</td>
<td>1104</td>
<td>7529.412</td>
<td>0.001</td>
<td>6.820</td>
<td>1494.748</td>
<td>41.56%</td>
</tr>
<tr>
<td>clone</td>
<td>162</td>
<td>29.729</td>
<td>0.160</td>
<td>0.184</td>
<td>0.358</td>
<td>1.15%</td>
</tr>
<tr>
<td>mprotect</td>
<td>397</td>
<td>2.239</td>
<td>0.002</td>
<td>0.006</td>
<td>0.047</td>
<td>4.41%</td>
</tr>
<tr>
<td>mmap</td>
<td>475</td>
<td>1.895</td>
<td>0.002</td>
<td>0.004</td>
<td>0.016</td>
<td>1.59%</td>
</tr>
<tr>
<td>open</td>
<td>445</td>
<td>1.382</td>
<td>0.001</td>
<td>0.003</td>
<td>0.018</td>
<td>2.25%</td>
</tr>
<tr>
<td>poll</td>
<td>178</td>
<td>1.291</td>
<td>0.001</td>
<td>0.007</td>
<td>0.433</td>
<td>45.09%</td>
</tr>
<tr>
<td>writev</td>
<td>165</td>
<td>0.940</td>
<td>0.001</td>
<td>0.006</td>
<td>0.407</td>
<td>43.54%</td>
</tr>
<tr>
<td>stat</td>
<td>497</td>
<td>0.914</td>
<td>0.001</td>
<td>0.002</td>
<td>0.010</td>
<td>2.56%</td>
</tr>
<tr>
<td>munmap</td>
<td>43</td>
<td>0.612</td>
<td>0.003</td>
<td>0.014</td>
<td>0.167</td>
<td>29.02%</td>
</tr>
<tr>
<td>read</td>
<td>169</td>
<td>0.436</td>
<td>0.001</td>
<td>0.003</td>
<td>0.040</td>
<td>14.05%</td>
</tr>
</tbody>
</table>
```
$ perf trace record --call-graph dwarf -o perf.raw.data -- \\n   ./lab_mandelbrot_concurrent -b 5
$ perf inject --trace -i perf.raw.data -o perf.data
$ perf report

Samples: 16K of event 'raw_syscalls:sys_enter', Event count: 285504696458
  Overhead Trace output
  - 77.61% NR 202 (7ffd1fe2f0c0, 80, 3, 0, 0, 0)
    __clone .:0
    start_thread pthread_create.c:0
    0xacd78
    0xa901f
    QtConcurrent::RunFunctionTask::run qtconcurrentstoredfunctioncall.h:679
    (anonymous namespace)::firstPassMandelbrotRow qmutex.h:138
    QMutex::lock +71
    QBasicMutex::lockInternal +53
    syscall +25

perf inject: Make it possible to merge sys_enter/exit events
perf timechart

Visualize scheduling of processes:

$ perf timechart record -- sleep 5
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 2.093 MB perf.data (5852 samples) ]
$ perf timechart
Written 5.0 seconds of trace to output.svg.
perf timechart
Scripting

- Inspecting Data
- Generating Scripts
- Example Scripts
perf script

Output raw data stream:

$ perf record --call-graph dwarf -- ...
$ perf script
...
lab_mandelbrot  28224  13656.319122:  995647 cycles:ppp:
  403a8a  _start+0x2a (/ssd/milian/projects/kdab/training-material/addon)
  7f8a88e1c291  __libc_start_main+0xf1 (/usr/lib/libc-2.24.so)
  40307a  main+0x4a (/ssd/milian/projects/kdab/training-material/addon)
  7f8a8ae39ae9  QApplicationPrivate::init+0x9 (/usr/lib/libQt5Widgets.so.5.7)
  7f8a8a69e7af  QGuiApplicationPrivate::init+0x2f (/usr/lib/libQt5Gui.so.5.7)
  7f8a8a15b3bf  QCoreApplicationPrivate::init+0xaaf (/usr/lib/libQt5Core.so.5)
  7f8a8a69c2ed  QGuiApplicationPrivate::createEventDispatcher+0x2d (/usr/lib)
  7f8a8a69c252  QGuiApplicationPrivate::createPlatformIntegration +0x5c2 (/us
  7f8a8a68e01f  QPlatformIntegrationFactory::create+0xef (/usr/lib/libQt5Gui
  7f8a8a13f9c2  QFactoryLoader::instance+0x112 (/usr/lib/libQt5Core.so.5.7.0)
  ...
perf script

Python bindings:

```bash
generated Python script: perf-script.py

in trace_begin

syscalls__sys_enter_nanosleep 1 27967.765309546 28821 ex_sleep
[4009da] __start
[7f73e75a4291] __libc_start_main
[40095e] main
[7f73e87020cb] QThread::sleep
[7f73e87a81ed]
[7f73e81d5780] __nanosleep

...

in trace_end
```
def trace_begin():
    print "in trace_begin"

def trace_end():
    print "in trace_end"

def syscalls__sys_exit_nanosleep(event_name, context, common_cpu, ...):
    print_header(event_name, common_cpu, common_secs, ...)
    print "__syscall_nr=%d, ret=%d" % (__syscall_nr, ret)
    for node in common_callchain:
        if 'sym' in node:
            print "\t[%x] %s"
            % (node['ip'], node['sym']["name"])
        else:
            print "[%x]" % (node['ip'])
    print "\n"

def syscalls__sys_enter_nanosleep(event_name, context, common_cpu, ...):
    ...
**perf script**

Convert perf.data to callgrind format:

```bash
$ perf record --call-graph dwarf ...
$ perf script report callgrind > perf.callgrind
$ kcachegrind perf.callgrind
```

github.com/milianw/linux/.../callgrind.py
perf script

Convert perf.data to callgrind format:

<table>
<thead>
<tr>
<th>Incl.</th>
<th>Self</th>
<th>Called</th>
<th>Function</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 110 302 401</td>
<td>0</td>
<td>1</td>
<td>_libc_start_main</td>
<td>libc-2.24.so</td>
</tr>
<tr>
<td>1 110 302 401</td>
<td>1 110 302 401</td>
<td>1</td>
<td>_nanosleep</td>
<td>libpthread-2.24.so</td>
</tr>
<tr>
<td>1 110 302 401</td>
<td>0</td>
<td>(0)</td>
<td>_start</td>
<td>ex_sleep</td>
</tr>
<tr>
<td>1 110 302 401</td>
<td>0</td>
<td>3</td>
<td>(unknown)</td>
<td>libQt5Core.so.5.7.0</td>
</tr>
<tr>
<td>1 110 302 401</td>
<td>0</td>
<td>1</td>
<td>main</td>
<td>ex_sleep: main.cpp</td>
</tr>
<tr>
<td>1 000 005 152</td>
<td>0</td>
<td>1</td>
<td>QThread::sleep</td>
<td>libQt5Core.so.5.7.0</td>
</tr>
<tr>
<td></td>
<td>100 146 889</td>
<td>0</td>
<td>QThread::msleep</td>
<td>libQt5Core.so.5.7.0</td>
</tr>
<tr>
<td></td>
<td>10 150 360</td>
<td>0</td>
<td>QThread::usleep</td>
<td>libQt5Core.so.5.7.0</td>
</tr>
</tbody>
</table>
Questions?

milian.wolff@kdab.com
http://www.kdab.com

We offer **Debugging and Profiling** trainings and workshops

github.com/milianw/linux/tree/milian/perf

```
git clone -b milian/perf https://github.com/milianw/linux.git
```