Extending the Performance Analysis Toolset
Christoph Sterz, KDAB
Embedded Performance

This system is *different* than your desktop.
Time in this system is spent *differently*.

*be scientific and measure!*

2. Analyze the info to change your code with Hotspot.

3. Produce additional, different information with LTTNG Userspace Tracing in Qt.
Linux Perf

Don't forget perf in your Kernel Config!

$ zgrep -E "PERF|TRACE" /proc/config.gz

[...]

# can we do perf at all?
CONFIG_HAVE_PERF_EVENTS=y
CONFIG_PERF_EVENTS=y

# can we do dwarf unwinding?
CONFIG_HAVE_PERF_REGS=y
CONFIG_HAVE_PERF_USER_STACK_DUMP=y

# do we have tracepoints?
CONFIG_FTRACE=y
CONFIG_TRACEPOINTS=y
CONFIG_SCHED_TRACER=y
CONFIG_HAVE_SYSCALL_TRACEPOINTS=y
Debugsymbs:
For Profiling:
Always build Release with Debug info!

```c
int intSum(int to) {
    int sum = 0;
    for (int i = 0; i < to; ++i) {
        sum += i;
    }
    return sum;
}

#Compiled with and without -O2
```
Debugsymbs:
How does the (embedded) world look like?

We don't have space

libQt5Core.so

90M
5M
Debugsymbs:
How does the (embedded) world look like?

We don't have space

libQt5Core.so

90M

5M

Separate dwarf debug Info

.debug on host
(somewhere in SDK)
Debugsymbs:
How does the (embedded) world look like?

Architectures mismatch

```markdown
<table>
<thead>
<tr>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM / MIPS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td>x86</td>
</tr>
</tbody>
</table>
```
Debugsymbs:
How does the (embedded) world look like?

Architectures mismatch

target
ARM / MIPS

x86
host

unwind embedded architecture
SP info on the host.
Linux Perf

Does the CPU support low-overhead with PMU?

Look for "PMU" in dmesg

-> on some ARM CPUs only core0 is counted.

Lets Get our debug info.

target ~$ perf record --callgraph dwarf -- myapp
Now we have our profiling info, we need to process it.
On the device unwind result looks like this

<table>
<thead>
<tr>
<th>Children</th>
<th>Self</th>
<th>Command</th>
<th>Shared Object</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>settings</td>
<td>libc-2.26.so</td>
</tr>
<tr>
<td>+ 36.61%</td>
<td>11.42%</td>
<td>settings</td>
<td>libQt5Qml.so.5.10.0</td>
</tr>
<tr>
<td>+ 29.33%</td>
<td>5.84%</td>
<td>settings</td>
<td>libQt5Core.so.5.10.0</td>
</tr>
<tr>
<td>+ 28.41%</td>
<td>7.94%</td>
<td>settings</td>
<td>settings</td>
</tr>
<tr>
<td>+ 27.76%</td>
<td>1.05%</td>
<td>settings</td>
<td>libglib-2.0.so.0.5400.0</td>
</tr>
<tr>
<td>+ 20.34%</td>
<td>1.38%</td>
<td>settings</td>
<td>libQt5Quick.so.5.10.0</td>
</tr>
<tr>
<td>+ 20.14%</td>
<td>0.07%</td>
<td>settings</td>
<td>libQt5XcbQpa.so.5.10.0</td>
</tr>
<tr>
<td>+ 19.29%</td>
<td>1.51%</td>
<td>settings</td>
<td>libQt5Gui.so.5.10.0</td>
</tr>
<tr>
<td>+ 15.81%</td>
<td>2.76%</td>
<td>QSGRenderThread</td>
<td>libQt5Quick.so.5.10.0</td>
</tr>
<tr>
<td>+ 14.11%</td>
<td>11.88%</td>
<td>QSGRenderThread</td>
<td>i965_dri.so</td>
</tr>
<tr>
<td>+ 14.11%</td>
<td>2.82%</td>
<td>QSGRenderThread</td>
<td>libc-2.26.so</td>
</tr>
<tr>
<td>+ 13.85%</td>
<td>0.92%</td>
<td>QSGRenderThread</td>
<td>libQt5Core.so.5.10.0</td>
</tr>
<tr>
<td>+ 13.65%</td>
<td>0.33%</td>
<td>QSGRenderThread</td>
<td>libpthread-2.26.so</td>
</tr>
<tr>
<td>+ 12.86%</td>
<td>0.85%</td>
<td>QQuickPixmapRea</td>
<td>libc-2.26.so</td>
</tr>
<tr>
<td>+ 12.20%</td>
<td>0.20%</td>
<td>QQuickPixmapRea</td>
<td>libQt5Core.so.5.10.0</td>
</tr>
<tr>
<td>+ 12.07%</td>
<td>0.07%</td>
<td>QQuickPixmapRea</td>
<td>libpthread-2.26.so</td>
</tr>
<tr>
<td>+ 12.07%</td>
<td>0.00%</td>
<td>QQuickPixmapRea</td>
<td>libQt5Quick.so.5.10.0</td>
</tr>
<tr>
<td>+ 12.01%</td>
<td>0.26%</td>
<td>QQuickPixmapRea</td>
<td>libQt5Gui.so.5.10.0</td>
</tr>
<tr>
<td>+ 12.01%</td>
<td>0.00%</td>
<td>QQuickPixmapRea</td>
<td>settings</td>
</tr>
<tr>
<td>+ 11.35%</td>
<td>3.94%</td>
<td>QQuickPixmapRea</td>
<td>libpng16.so.16.34.0</td>
</tr>
<tr>
<td>+ 11.09%</td>
<td>4.86%</td>
<td>settings</td>
<td>libpng16.so.16.34.0</td>
</tr>
<tr>
<td>+ 8.27%</td>
<td>0.00%</td>
<td>settings</td>
<td>[unknown]</td>
</tr>
</tbody>
</table>
On the device unwind result looks like this:

```
Samples: 1K of event 'cycles:uppp', Event count (approx.): 1524000000

<table>
<thead>
<tr>
<th>Children</th>
<th>Self</th>
<th>Command</th>
<th>Shared Object</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>22.83% __libc_start_main</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20.54% main</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20.08% QCoreApplication::exec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19.88% 0xc5151</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>QEventDispatcherGlib::processEvents</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+ g_main_context_iteration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.30% __libc_csu_init</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>22.77% __libc_start_main</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20.47% main</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20.01% QCoreApplication::exec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19.82% 0xc5151</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>QEventDispatcherGlib::processEvents</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- g_main_context_iteration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- 19.49% 0x6cf68</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- g_main_context_dispatch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- 16.93% 0x2f82d1</td>
</tr>
</tbody>
</table>
```

[.] __libc_start_main
[.] _start
[.] __libc_start_main
[.] _start
[.] __libc_start_main
[.] _start
[.] __libc_start_main
[.] _start
[.] __libc_start_main
[.] _start
[.] __libc_start_main
[.] _start
[.] __libc_start_main
[.] _start
Hotspot

Hotspot is a FOSS Linux UI for visualizing perf results written in Qt and C++

... it uses perfparsert from QtCreator

... which uses libunwind / libdw for unwinding even off-architecture!

<Demo>

99,5% curtesy goes to @milianw
Here is what I like most

# settings to help you unwind off-target

--debugPaths <paths>
--extraLibPaths <paths>
--appPath <paths>
--sysroot <path>
--kallsyms <path>
LTTNG: We want it inside Qt

Great Source of Information
Existing Tools, but
Traces are mostly Kernel :(
LTNG in Userspace Qt

Qt has now a tool called tracegen.
Which creates the tracepoints for qtbase.

QGuiApplicationPrivate_init_entry()
QGuiApplicationPrivate_init_exit()

QGuiApplicationPrivate_processWindowSystemEvent_entry(int type)
QGuiApplicationPrivate_processWindowSystemEvent_exit(int type)

QFontDatabase_addApplicationFont(const QString &filename)
QFontDatabase_load(const QString &family, int pointSize)
QFontDatabase_loadEngine(const QString &family, int pointSize)
QFontDatabasePrivate_addAppFont(const QString &fileName)

QImageReader_read_before_reading(QImageReader *reader, const QString &filename)
QImageReader_read_after_reading(QImageReader *reader, bool result)
LTTNG in Userspace Qt

They look like this:

```c
#ifndef TP_QTGUI_QIMAGEREADER_READ_AFTER_READING
#define TP_QTGUI_QIMAGEREADER_READ_AFTER_READING
namespace QtPrivate {
inline void trace_QImageReader_read_after_reading(QImageReader * reader, bool result) {
    tracepoint(qtgui, QImageReader_read_after_reading, reader, result);
}
inline void do_trace_QImageReader_read_after_reading(QImageReader * reader, bool result) {
    do_tracepoint(qtgui, QImageReader_read_after_reading, reader, result);
}
inline bool trace_QImageReader_read_after_reading_enabled() {
    return tracepoint_enabled(qtgui, QImageReader_read_after_reading);
}
} // namespace QtPrivate
#endif // TP_QTGUI_QIMAGEREADER_READ_AFTER_READING
```
LTTNG in Userspace Qt

And are used like this:

```cpp

QTRACE(QImageReader_read_before_reading, this, fileName);
const bool result = d->handler->read(image);
QTRACE(QImageReader_read_after_reading, this, result);

```
How is LTNG different than perf?

Events

Events
Thanks!
How can I help?

christoph.sterz@kdab.com

Pictures: Wikimedia