Swimming with the New KernelShark

Yordan Karadzhov

VMware Inc. - OSTC
What is KernelShark?

* Front end reader of Linux kernel tracing data (Ftrace)
What is KernelShark?

* Front end reader of Linux kernel tracing data (Ftrace)

* The original version - started in 2009.

* Written in Gtk+-2.0

* Main goal: analyse and fully understood the performance of the Real-time scheduler.
What is KernelShark?

* Front end reader of Linux kernel tracing data (Ftrace)

* New KernelShark: use all lessons learned from the old version.

* Completely rewritten to use Qt.

* But not only this ...
The New KernelShark is

a. Optimized for processing significantly larger amounts of data.

b. New scalable data model - log(n) time complexity.

c. OpenGL-based visualization.

d. Preconfigurable - Json config I/O.

e. User modifiable - plugins.
Why do we need KernelShark?

a. Ftrace - the official tracing infrastructure of the Linux kernel.

b. Extremely powerful instrument.

c. Allows to see what is happening in the kernel.

d. But you must know what you are looking for
<table>
<thead>
<tr>
<th>Process</th>
<th>Time</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compositor-1224</td>
<td>236364.639085</td>
<td>_cond_resched &lt;- futex_wait_queue_me</td>
</tr>
<tr>
<td>Compositor-1224</td>
<td>236364.639086</td>
<td>rcu_all_qs &lt;- _cond_resched</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639086</td>
<td>do_syscall_64 &lt;- entry_SYSCALL_64 after_hwframe</td>
</tr>
<tr>
<td>Compositor-1224</td>
<td>236364.639086</td>
<td>drop_futex_key_refs.isra.14 &lt;- futex_wait</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639087</td>
<td>syscall_trace_enter &lt;- do_syscall_64</td>
</tr>
<tr>
<td>Compositor-1224</td>
<td>236364.639087</td>
<td>hrtimer_cancel &lt;- futex_wait</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639087</td>
<td>__secure_computing &lt;- syscall_trace_enter</td>
</tr>
<tr>
<td>Compositor-1224</td>
<td>236364.639087</td>
<td>hrtimer_try_to_cancel &lt;- hrtimer_cancel</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639087</td>
<td>__seccomp_filter &lt;- __secure_computing</td>
</tr>
<tr>
<td>Compositor-1224</td>
<td>236364.639088</td>
<td>hrtimer_active &lt;- hrtimer_try_to_cancel</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639088</td>
<td>seccomp_run_filters &lt;- seccomp_filter</td>
</tr>
<tr>
<td>Compositor-1224</td>
<td>236364.639088</td>
<td>lock_hrtimer_base.isra.20 &lt;- hrtimer_try_to_cancel</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639088</td>
<td>raw_spin_lock_irqsave &lt;- lock_hrtimer_base.isra.20</td>
</tr>
<tr>
<td>Compositor-1224</td>
<td>236364.639089</td>
<td>do_syscall_64 &lt;- do_syscall_64</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639089</td>
<td>ktime_get &lt;- SysS_futex</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639089</td>
<td>ktime_add_safe &lt;- SysS_futex</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639090</td>
<td>do_futex &lt;- SysS_futex</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639091</td>
<td>futex_wait &lt;- do_futex</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639091</td>
<td>hrtimer_init &lt;- futex_wait</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639092</td>
<td>hrtimer_init &lt;- hrtimer_init</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639092</td>
<td>hrtimer_init_sleep &lt;- futex_wait</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639092</td>
<td>ktime_add_safe &lt;- futex_wait</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639093</td>
<td>futex_wait_setup &lt;- futex_wait</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639093</td>
<td>get_futex_key &lt;- futex_wait_setup</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639093</td>
<td>get_futex_key_refs.isra.13 &lt;- get_futex_key</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639094</td>
<td>hash_futex &lt;- futex_wait_setup</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639095</td>
<td>_raw_spin_lock &lt;- futex_wait_setup</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639095</td>
<td>get_futex_value_locked &lt;- futex_wait_setup</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639095</td>
<td>futex_wait_queue_me &lt;- futex_wait</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639096</td>
<td>hrtimer_start_range_ns &lt;- futex_wait_queue_me</td>
</tr>
<tr>
<td>Compositor-1224</td>
<td>236364.639097</td>
<td>do_syscall_64 &lt;- entry_SYSCALL_64 after_hwframe</td>
</tr>
<tr>
<td>chromium-browse-1117</td>
<td>236364.639097</td>
<td>lock_hrtimer_base.isra.20 &lt;- hrtimer_start_range_ns</td>
</tr>
</tbody>
</table>
**Ftrace** is extremely powerful but ... 

... it can be hard to find what you need. You must be **True (kernel) detective**.
**Ftrace** is extremely powerful but ...  

... it can be hard to find what you need. You must be **True (kernel) detective**.

Interactive visualization can be very useful.
Kernel Shark

Hmm, why do you think that this complete mess of colors can be of any help?
If we **Zoom in** a bit, we will start seeing the time structure of the trace.
Kernel Shark

This is a deep zoom. Let’s try to understand what is visualized here.
a. Double click on the graph selects a trace record.

b. The same can be done by clicking on the table row which shows the record as text.
Kernel Shark

Search panel

a. Provides interactive searching.

b. Very useful in combination with the Dual marker.
Kernel Shark

Selecting which CPU to visualize.
Kernel Shark

Selecting which Task to visualize.
Kernel Shark

Filter menu

a. Filter out tasks
b. Filter in tasks
c. Filter events
d. Advanced (content-based) filtering
Kernel Shark

Event filters
Kernel Shark - Sessions

- Import/Export session
- Restor last session.
Kernel Shark - Recording

a. Trace data can be recorded directly from Kernel Shark.
b. Root password is required.
Ftrace is extremely powerful but ...
Ftrace is extremely powerful but ...

What do we do if:

a. We have a very large data-set of trace records.

b. We are limited by the number of screen pixels available.

And we have to process all this in a reasonable amount of time.
Fitting something large inside something small ...

... and we have to do this quickly!
Visualization model - How does it work?

### Visualization Model

- **Entire data set**
  - **Subset of interest**
  - **Overflow Bin**
  - **Bin0**
  - **Bin1**
  - **Bin2**
  - **Bin_{N-1}**

#### Steps:

a. Break the data-set into **time-bins** $\leftrightarrow$ like a histogram.

b. Check only the records at the beginning and at the end of each bin. $\leftrightarrow$ constant time.
Visualization model - How does it work?

c. Have the trace records, sorted in time.

d. Knowing the index of the first record in each Bin determines the state of the model.

e. But the first element can be found with a binary search $\Rightarrow \log(n)$ complexity.
Visualization model - How does it work?

Data Binning provides $O(\log_2(n))$ average time complexity of all operations of the model.
Visualization model & tracing data formats

- The KernelShark Visualization model is not coupled to a particular data format.
- Uses KernelShark-specific data structure.
- Contains only the absolute minimum of information need by the model.
- The rest of the information - available on demand (can be slow)
Visualization model & tracing data

<table>
<thead>
<tr>
<th>Entire data set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overflow Bin</td>
</tr>
<tr>
<td>Bin0</td>
</tr>
<tr>
<td>Bin1</td>
</tr>
<tr>
<td>Bin2</td>
</tr>
<tr>
<td>Bin3</td>
</tr>
<tr>
<td>BinN-1</td>
</tr>
<tr>
<td>Overflow Bin</td>
</tr>
</tbody>
</table>

- Only one model (data structure) for all graphs.
- Worst-case complexity becomes linear.
- Solution - Data collections.
Visualization model & Data collections.

- Only one model (data structure) for all graphs.
- Worst-case complexity becomes linear.
- Solution - Data collections.
DEMO
KernelShark: current version 0.9

https://git.kernel.org/pub/scm/utils/trace-cmd/trace-cmd.git/

To build the code follow the instructions in

/trace-cmd/kernel-shark-qt/README

and

/trace-cmd/README
KernelShark is not a GUI. KernelShark is a toolkit.

What’s next after KernelShark 1.0?

a. KernelShark engine (libkshark.so)
b. Available under GNU LGPL v2.1
c. Highly customizable (via plugins)
d. Will read multiple data formats
KernelShark is not a GUI. KernelShark is a toolkit.

- KernelShark
- Apache
- Python Scripts

libkshark engine

- trace.dat
- perf.data
- CTF

What’s next after KernelShark 1.0?

a. Any tool will be able to use the library
b. Available for Python applications (libkshark.py)
c. The KernelShark application is just a “shell”.

©2018 VMware, Inc.