Flame Graphs for MySQL DBAs

Valerii Kravchuk, Principal Support Engineer, MariaDB

vkravchuk@gmail.com

Profiling - challenges and solutions...

- Profiling is basically measuring frequency or duration of function calls, or any resource usage per function call
- **Problem:** for complex software like MySQL server **perf** (or any other profiler) produces too large data sets to study efficiently
- **Solutions:** filtering (with **grep**), summarizing (with **awk** etc, see how **pt-pmp** does this for **gdb** backtraces, some 120 lines of code) or … visualization as Heat Maps or Flame Graphs (or in some GUI)
- It's not a Linux-only profiling problem, Windows Performance Analyzer (WPA) also supports flame graphs

Raw profiling data are just timestamps and stacks

- Let's run typical profiling session with **perf** while MySQL is under load:

```
openxs@ao756:~$ sudo perf record -a -g -F99 -- sleep 30
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 1,144 MB perf.data (1684 samples) ]
```

- Raw **perf** data are hardly useful "as is":

```
openxs@ao756:~$ openxs@ao756:~$ sudo perf script | more
...
mysqld 143863 [001] 105802.967446:4744028   cycles:
         557a3c9d08bc insert_events_statements_history+0xac
(/usr/sbin/mysqld
)
         557a3c9c658b pfs_end_statement_v1+0x14ab (/usr/sbin/mysqld)
         557a3c6ec8b7 dispatch_command+0x557 (/usr/sbin/mysqld)
         557a3c6ee79f do_command+0x1ff (/usr/sbin/mysqld)
         557a3c7aecd8 handle_connection+0x2e8 (/usr/sbin/mysqld)
         557a3c9bf2c8 pfs_spawn_thread+0x168 (/usr/sbin/mysqld)
         7f2fea56d609 start_thread+0xd9
(/usr/lib/x86_64-linux-gnu/libpthread
-2.31.so) ...
```

- We still have to summarize them somehow for better overview!

Problem of MySQL profiling - overview of the data

- We can summarize them with **perf report**:

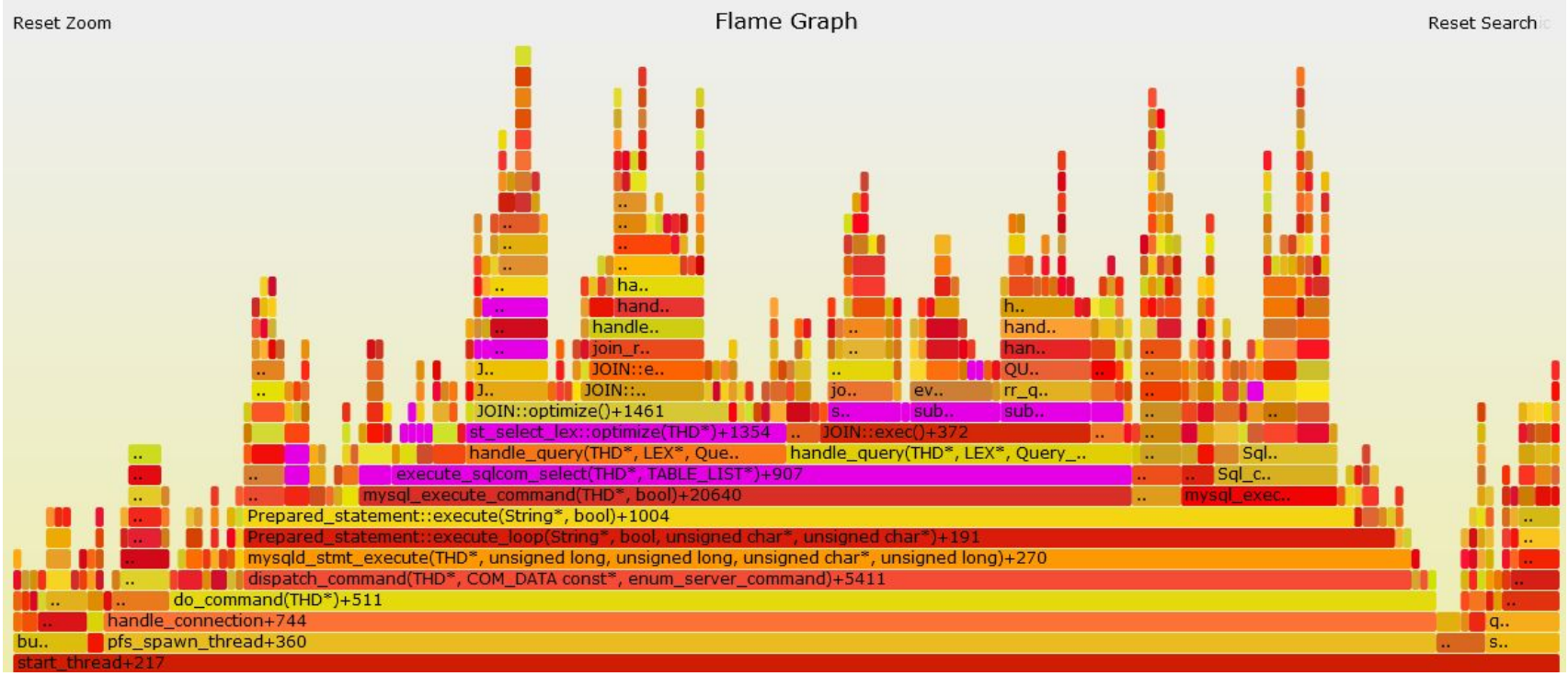
```
openxs@ao756:~$ sudo perf report > perf.out
```

- Here is a small part of the output in **perf.out** (small font is in purpose):

```
...
        36.54%   0.00%  mysqld            libpthread-2.31.so       [.] start_thread
        |
        ---start_thread
                |
                |--32.59%--pfs_spawn_thread
                |        |
                |        --32.55%--handle_connection
                |                  |
                |                  --32.45%--do_command
                |                            |
                |                            |--30.96%--dispatch_command
                |                            |        |
                |                            |        --29.39%--mysqld_stmt_execute
                |                            |                  |
                |                            |                  --29.14%--Prepared_statement::execute_loop
                |                            |                            |
                |                            |                            --29.01%--Prepared_statement::execute
                |                            |                                      |
                |                            |                                      --27.80%--mysql_execute_command
                |                            |                                                |
                |                            |                                                |--19.09%--execute_sqlcom_select
                |                            |                                                |        |
                |                            |                                                |        |--18.20%--handle_query
                |                            |                                                |        |        |
                |                            |                                                |        |        |--9.37%--JOIN::exec
                |                            |                                                |        |        |        |
                |                            |                                                |        |        |        |--7.65%--sub_select
...
openxs@ao756:~$ ls -l perf.out
-rw-rw-r-- 1 openxs openxs 1109381 кві 25 15:55 perf.out
```

Flame Graphs: what are they and how they help

- *Flame graph* is a way for visualizing any cumulative metrics in nested hierarchies (like call stacks and time spent in each function)
- Consider this example (PS 5.7.33, **sysbench** read-write, **bpftrace**):



Flame Graphs - use free tools by Brendan Gregg

- **http://www.brendangregg.com/flamegraphs.html**
- Flame graphs produced by these tools are a visualization (as **.svg** file to be checked in browser) of profiled software, allowing the most frequent code-paths to be identified quickly and accurately.
- The x-axis shows the stack profile population, sorted *alphabetically* (it is not the passage of time), and the y-axis shows stack depth. Each rectangle represents a stack frame. The wider a frame is, the more often it was present in the stacks.
- **CPU Flame Graphs** ← profiling by sampling at a fixed rate. Check this post.
- **Memory Flame Graphs** ← tracing **malloc()**, **free()**, **brk()**, **mmap()**, **page_fault**
- **Off-CPU Flame Graphs** ← tracing file I/O, block I/O or scheduler
- Other kinds of flame graphs (Hot-Cold, Differential, **pt-pmp**-based etc),
- **https://github.com/brendangregg/FlameGraph** + **perf** + ... or **bcc** tools like **offcputime.py**

flamegraph.pl - basic options

```
openxs@ao756:~/git/FlameGraph$ ./flamegraph.pl --help
USAGE: ./flamegraph.pl [options] infile > outfile.svg

    --title TEXT     # change title text
    --subtitle TEXT  # second level title (optional)
    --width NUM      # width of image (default 1200)
    --height NUM     # height of each frame (default 16)
    --minwidth NUM   # omit smaller functions (default 0.1 pixels)
    --fonttype FONT  # font type (default "Verdana")
    --fontsize NUM   # font size (default 12)
    --countname TEXT # count type label (default "samples")
    --nametype TEXT  # name type label (default "Function:")
    --colors PALETTE # set color palette. choices are: hot (default), mem, ...
    --bgcolors COLOR # set background colors. gradient choices are yellow
...
    --hash           # colors are keyed by function name hash
    --cp             # use consistent palette (palette.map)
    --reverse        # generate stack-reversed flame graph
    --inverted       # icicle graph
    --flamechart     # produce a flame chart (sort by time, do not merge ...)
    --negate         # switch differential hues (blue<->red)
    --notes TEXT     # add notes comment in SVG (for debugging)
...
```

flamegraph.pl - expected input format

- Flame graphs can be generated from any profile data that contains “stack traces”. This can be abused to show any (cumulative) metric over a hierarchical structure.
- Check comments in the source code for format details:

```
   ...
# The input is stack frames and sample counts formatted as single
# lines.  Each frame in the stack is semicolon separated, with a
# space and count at the end of the line.  These can be generated
# for Linux perf script output using stackcollapse-perf.pl, for
# DTrace using stackcollapse.pl, and for other tools
# using the other stackcollapse programs.  Example input:
#
#  swapper;start_kernel;rest_init;cpu_idle;default_idle;nati... 1
#
# An optional extra column of counts can be provided to generate a
# differential flame graph of the counts, colored red for more,
# and blue for less...
...
```

Flame Graphs - tools to process stack traces

- Different stack output formats are supported by the tools, including **gdb**, **perf** and **bpftrace**:

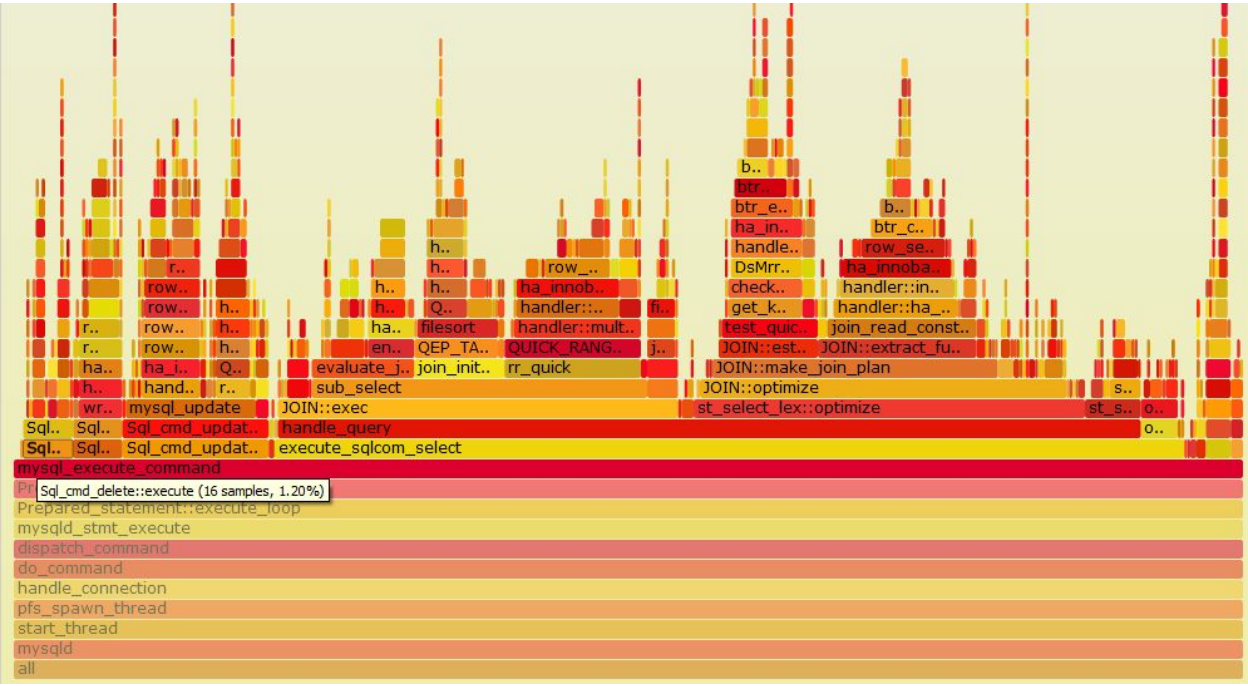
```
openxs@ao756:~/git/FlameGraph$ ls *.pl
aix-perf.pl                 stackcollapse-instruments.pl
difffolded.pl               stackcollapse-java-exceptions.pl
files.pl                    stackcollapse-jstack.pl
flamegraph.pl               stackcollapse-perf.pl
pkgsplit-perf.pl            stackcollapse.pl
range-perf.pl               stackcollapse-pmc.pl
stackcollapse-aix.pl        stackcollapse-recursive.pl
stackcollapse-bpftrace.pl   stackcollapse-stap.pl
stackcollapse-elfutils.pl   stackcollapse-vsprof.pl
stackcollapse-gdb.pl        stackcollapse-vtune.pl
stackcollapse-go.pl
```

- USAGE notes and sample command lines are presented in **.pl** files as comments

CPU Flame Graph - simple example

- Created based on these steps (while **sysbench oltp_read_write** was running):

```
openxs@ao756:~/git/FlameGraph$ sudo perf record -F 99 -a -g -- sleep 20
openxs@ao756:~/git/FlameGraph$ perf script | ./stackcollapse-perf.pl > /tmp/perf-folded.out
openxs@ao756:~/git/FlameGraph$ ./flamegraph.pl --width=1000 /tmp/perf-folded.out > /tmp/mysqld_sysbench_read_write.svg
```



Custom CPU Flame Graph - hot mutex waits

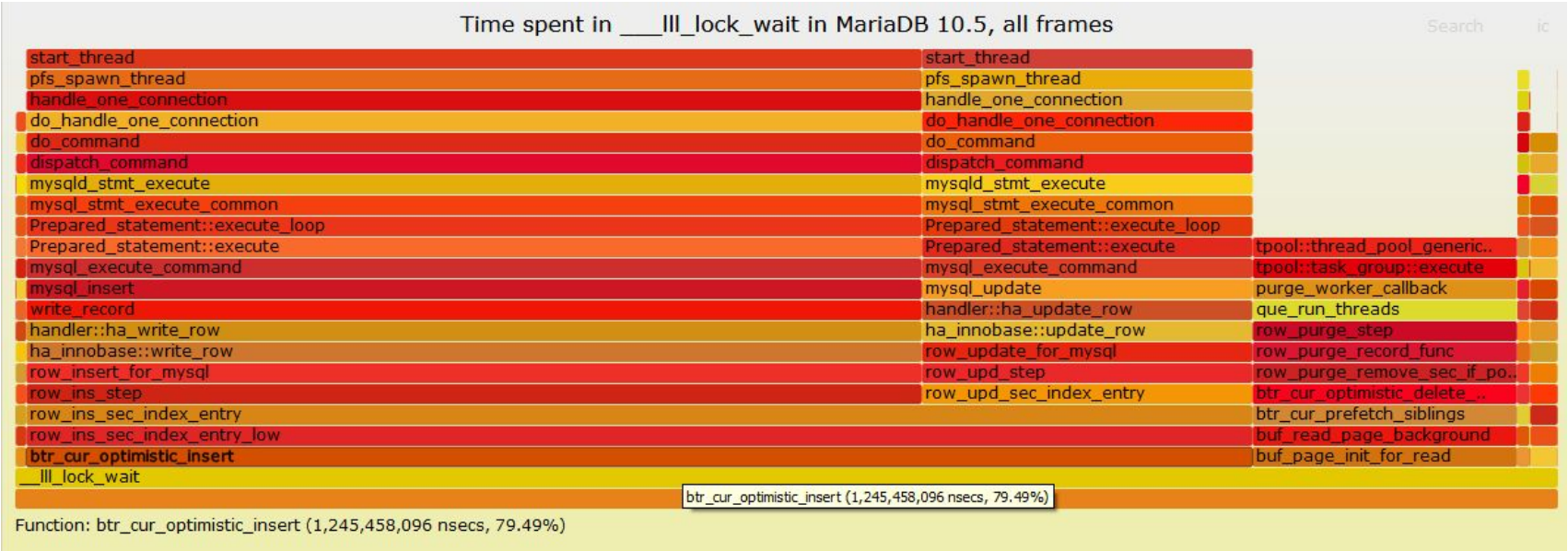
- In some cases you may want to collapse stacks yourself. Check this blog post for the details, but the idea was get “clean” frames from **bpftrace** (no address, arguments etc), for better summarizing, and remove “garbage” output:

```
[openxs@fc31 ~]$ time sudo ./lll_lock_wait2.bt 60 2>/dev/null | awk '
BEGIN { s = ""; }
/^@futexstack\[\]/ { s = ""; }
/^@futexstack/ { s = ""; }
/^\t/ { if (index($2, "(") > 0) {targ = substr($2, 1, index($2, "(") - 1)}
else {targ = substr($2, 1, index($2, "+") - 1)} ; if (s != "") { s = s ";"
targ } else { s = targ } }
/^]/ { print $2, s }
' > /tmp/collapsed_lll_lock_v2_raw.txt

[openxs@fc31 ~]$ cat /tmp/collapsed_lll_lock_v2_raw.txt | awk '{ if
(length($2) > 0) {print $2, $1} }' |
/mnt/home/openxs/git/FlameGraph/flamegraph.pl --title="Time spent in
___lll_lock_wait in MariaDB 10.5, all frames" --countname=nsecs >
~/Documents/lll_lock_v2_2.svg
```

Flame Graphs - what paths lead to mutex waits

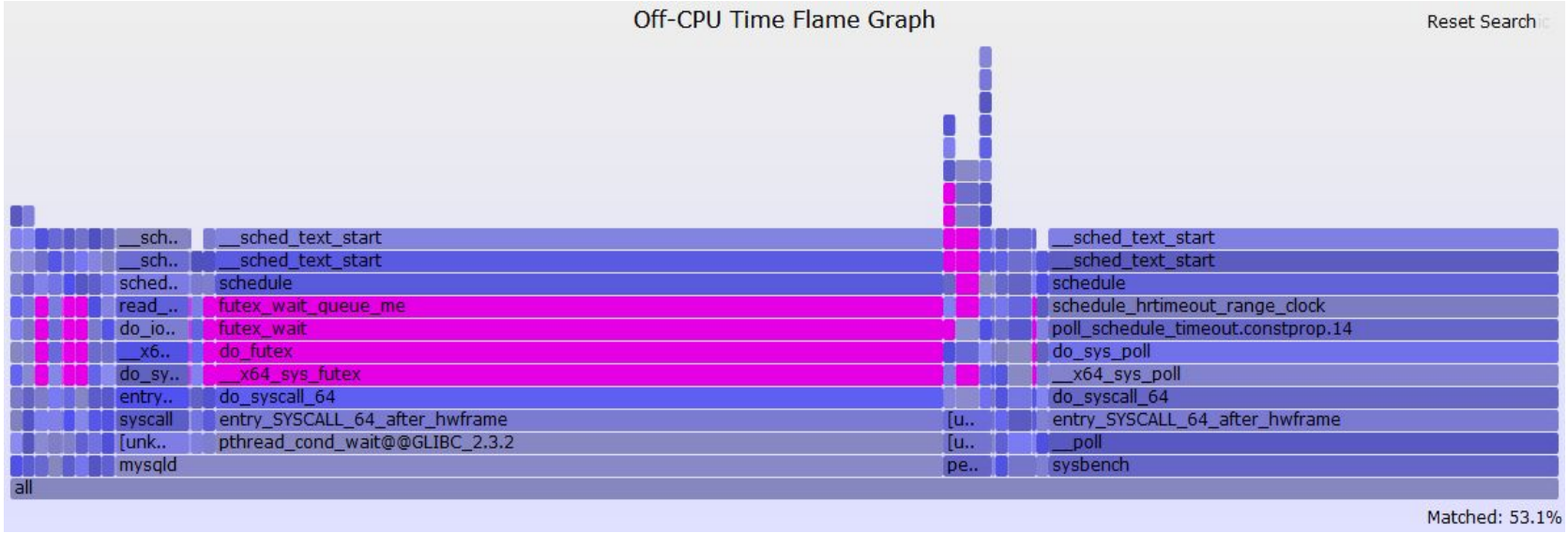
- We ended up with the following result for the **sysbench oltp_read_write** test running inserts into 5 tables from 32 threads on 4 cores:



Off-CPU Flame Graph - simple example

- Created based on these steps (while **oltp_update_index.lua** was running):

```
[openxs@fc29 FlameGraph]$ sudo /usr/share/bcc/tools/offcputime -df 60 > /tmp/out.stacks
WARNING: 459 stack traces lost and could not be displayed.
[openxs@fc29 FlameGraph]$ ./flamegraph.pl --color=io --title="Off-CPU Time Flame Graph" --countname=us < /tmp/out.stacks > ~/Documents/out.svg
```

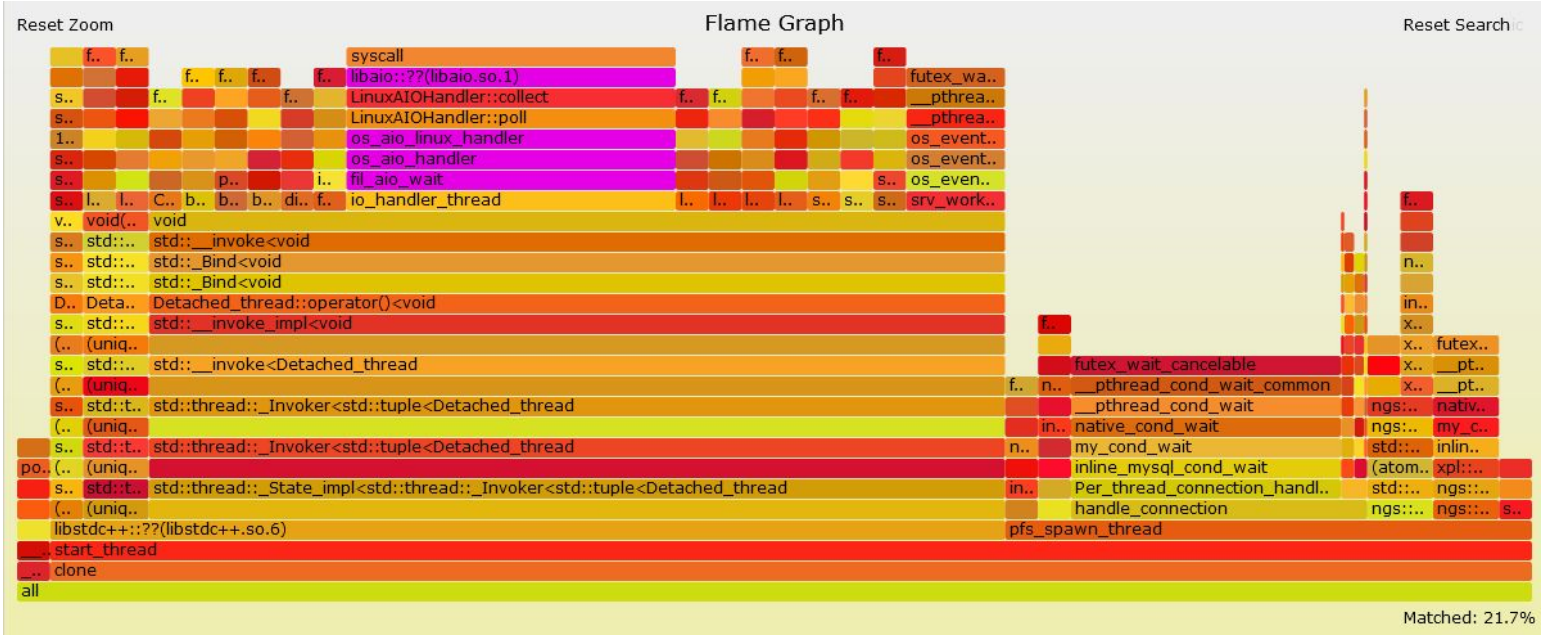


- I've searched for "futex" and related frames are highlighted

Flame Graph based on pt-pmp output

- Created based on this (while **oltp_read_only.lua** was running on 8.0.27):

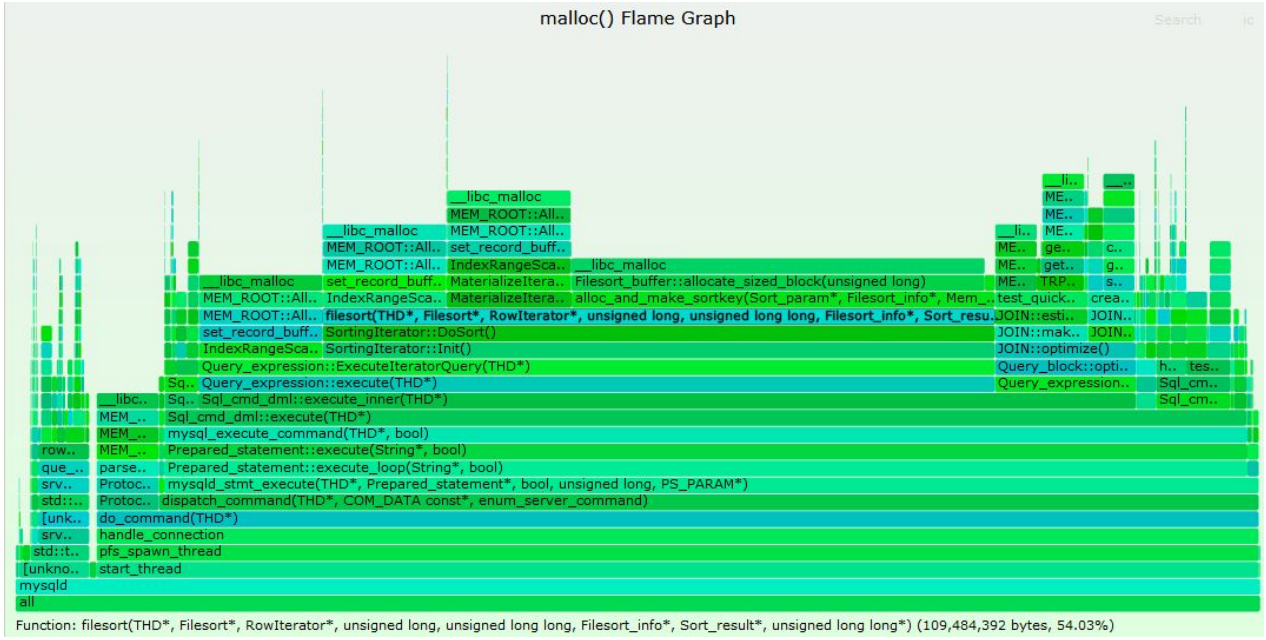
```
openxs@ao756:~$ sudo pt-pmp --iterations 10 --interval 1 > /tmp/pmp.out
openxs@ao756:~$ tail -n+2 /tmp/pmp.out | awk '{print $2, $1}' | sed -e
's/,/;/g'| ~/git/FlameGraph/flamegraph.pl --countname threads --reverse >
/tmp/pmp.svg
```



Memory Flame Graph

- Created based on output of hacked old **mallocstacks.py** from **BPF-Tools**. Better use this version from **bcc** tools today. See this blog post for details:

```
openxs@ao756:~$ sudo ~/git/BPF-tools/old/2017-12-23/mallocstacks.py -p $(pidof mysqld) -f >/tmp/alloc.out
openxs@ao756:~$ cat /tmp/alloc.out | ~/git/FlameGraph/flamegraph.pl --color=mem --title="malloc() Flame Graph" --countname="bytes" >/tmp/mysql8_malloc.svg
```



Flame Graphs based on Performance Schema

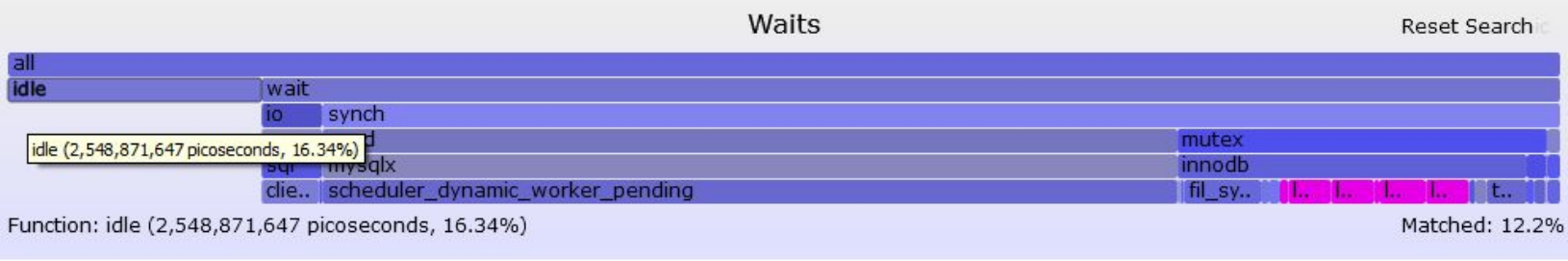
- Consider this output, where we see a clear hierarchy of instrumented waits:

```
mysql> select event_name, timer_wait from events_waits_history_long order by 1 desc limit 5;
…
| wait/synch/sxlock/innodb/trx_purge_latch      |    747273   |
…
```

- It takes just a few simple steps to convert this to a Flame Graph:

```
openxs@ao756:~/dbs/8.0$ cat /tmp/waits.txt | awk '{ printf("%s %d\n", $1, $2); }' | sed 's/\//;/g' | ~/git/FlameGraph/flamegraph.pl --inverted --colors io --title "Waits" --countname picoseconds --width 1000 > /tmp/wait.svg
```

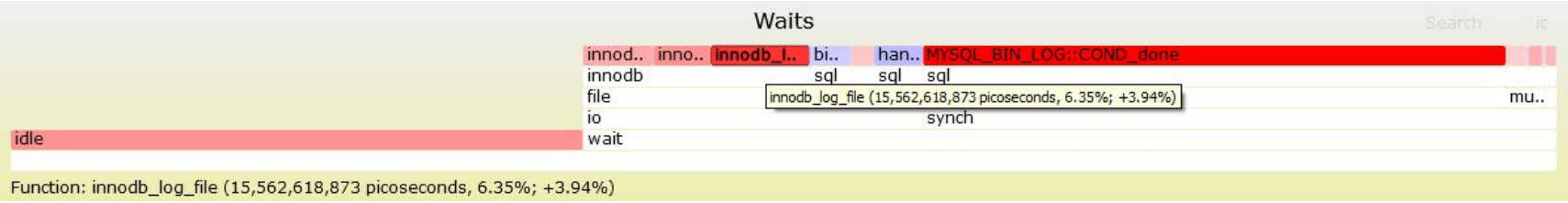
- You can add transactions, statements and stages on top:



Differential Flame Graphs

- The idea is to compare two flame graphs and highlight the difference (with red for increase and blue for decrease). See this blog post and links from it…
- Check this page for more details and types of differential flame graphs
- I've tried to compare **performance_schema** reported waits for write only **sysbench** test with **innodb_flush_log_at_trx_commit** values of 0 and 1:

```
openxs@ao756:~/dbs/8.0$ ~/git/FlameGraph/difffolded.pl /tmp/w64_0.out
/tmp/w64_1.out | ~/git/FlameGraph/flamegraph.pl --count picoseconds
--title Waits > /tmp/w64_01_diff.svg
```

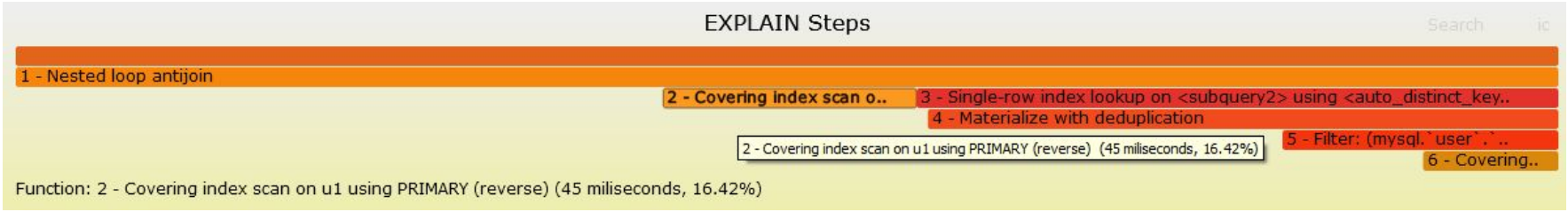


Visualizing MySQL Plan Execution Time

- **EXPLAIN ANALYZE** in MySQL 8.0.18+ presents a TREE view of query execution steps with several (estimated and real) metrics for each step:

```
mysql> explain analyze select user, host from mysql.user u1 where u1.user
not in (select distinct user from mysql.user) order by host desc\G
*************************** 1. row ***************************
EXPLAIN: -> Nested loop antijoin  (cost=3.75 rows=25) (actual
time=0.139..0.139 rows=0 loops=1)
    -> Covering index scan on u1 using PRIMARY (reverse)  (cost=0.75
rows=5) (actual time=0.058..0.064 rows=5 loops=1)
...
```

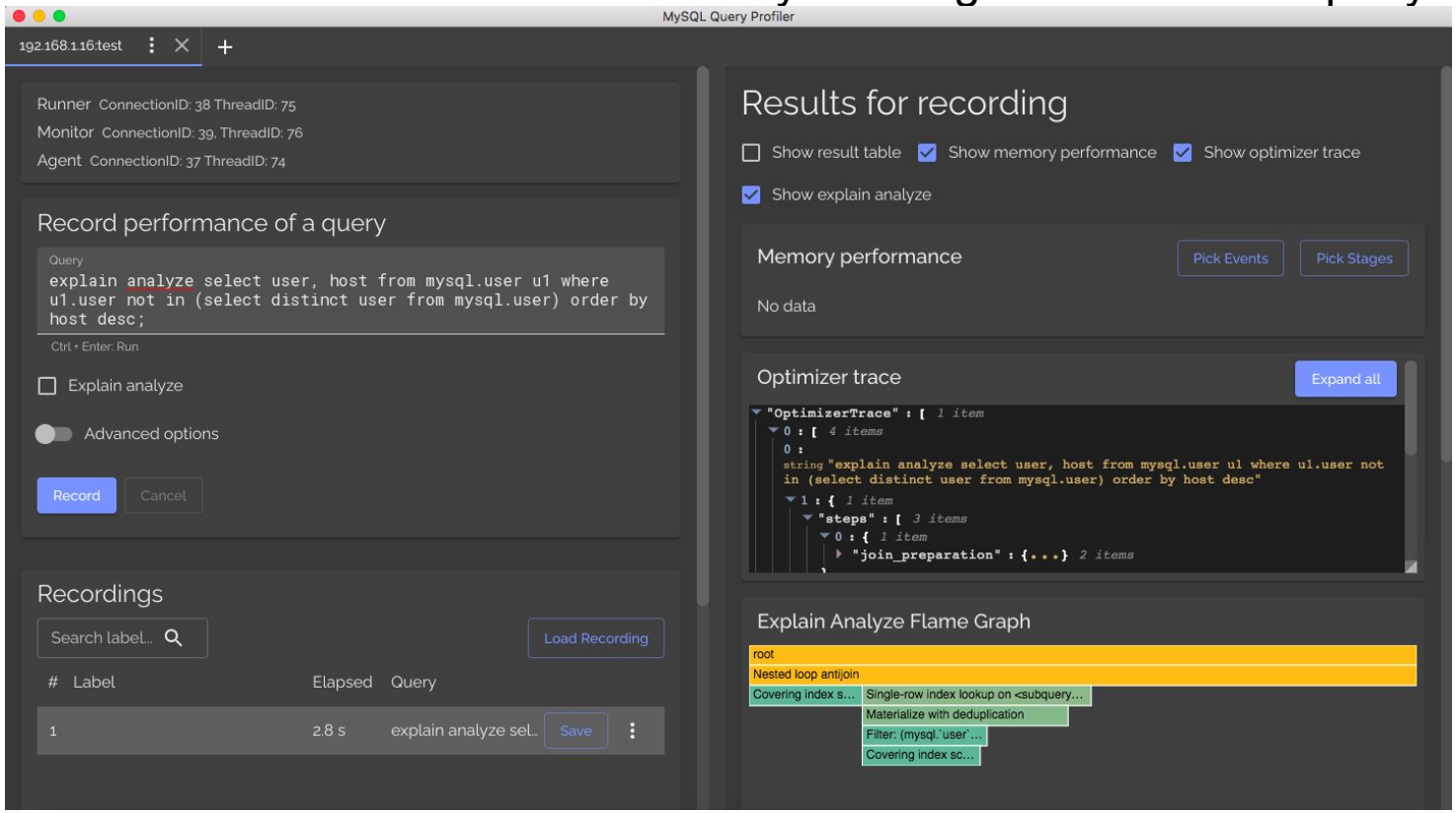
- With some efforts any tree of this kind can be visualized as a flame graph:



- I wish we had there in the table (like in Oracle) or in JSON format

Flame Graphs in MySQL Query Profiler

- You can get **EXPLAIN ANALYZE** output presented as a flame graph in the open source tool called MySQL Query Profiler
- I've built it on macOS and here is what you can get for the same query:



Flame Graphs - more examples, Q&A

- MySQL bug reports based on flame graphs (**Mark Callaghan**):
  - **Bug #102238** - "**log_writer uses too much CPU on small servers**". 8.0.22
  - **Bug #102037** - "**CPU overhead from inlists much larger in 8.0.22**".
- See also (from my collection):
  - https://www.percona.com/blog/2019/11/20/profiling-software-using-perf-and-flame-graphs/
  - https://www.percona.com/blog/2020/01/15/using-flame-graphs-to-process-outputs-from-pt-pmp
  - https://github.com/pingcap/tidb/pull/12986 - PR for TiDB (PingCap)
  - https://randomascii.wordpress.com/2013/03/26/summarizing-xperf-cpu-usage-with-flame-graphs/ - WPA/Windows
  - https://archive.fosdem.org/2020/schedule/event/mysql_cpu_flames/ - **"CPU performance analysis for MySQL using Hot/Cold Flame Graph"**
- **Questions and Answers?**