WHAT IF YOUR NIF GOES ADRIFT?

JULIAN SQUIRES

WHERE DO NIFS COME FROM?

WAYS NIFS GO WRONG

If you accept an incubus (otherwise known as a NIF) into your bed (otherwise known as the Erlang VM), you must not be surprised if you give birth to monsters.

— Richard A. O'Keefe

TRYING TO VALGRIND STOCK VM

```
% time ./rebar eunit
[...]
./rebar eunit  0.24s user 0.08s system 115% cpu 0.280 total

% time valgrind --trace-children=yes ./rebar eunit
[...]
[...] ./rebar eunit  111.02s user 0.57s system 100% cpu 1:51.28 total
```

1:51.28 total

OTHER BAD THINGS

```
% LD_PRELOAD=/usr/lib/x86_64-linux-gnu/libjemalloc.so.1 erl
Erlang/OTP 18 [erts-7.2.1] [source] [64-bit] [smp:4:4] [async-threads:10] [hip

Eshell V7.2.1  (abort with ^G)
1>
```

VS

```
% LD_PRELOAD=/usr/lib/libtcmalloc.so.4.2.2 erl
zsh: segmentation fault  LD_PRELOAD=/usr/lib/libtcmalloc.so.4.2.2 erl
```

DEBUG BUILDS OF ERTS

```
cd $ERL_TOP/erts/emulator
make TYPE=debug smp
$ERL_TOP/bin/cerl -rgdb
```

- debug
- valgrind
- lcnt
- --with-dynamic-trace={dtrace,systemtap}

LCNT

```
11> lcnt:apply(fun () -> erljson_bench:main(["lots-of-doubles-3.term", "100000
12> lcnt:conflicts().
      lock   id  #tries  #collisions  collisions [%]  time [us]  duration [%]
     -----  --- ------- ------------ --------------- ---------- -------------
 proc_main 1029   30148           43          0.1426         16        0.0000
 run_queue    4   79598            8          0.0101          4        0.0000
 proc_msgq 1029    6338            6          0.0947          1        0.0000
ok
```

BAD LCNT OUTPUT

```
(rtb-gateway@h091)21> lcnt:clear(), timer:sleep(1000), lcnt:collect(), lcnt:co
           lock    id  #tries  #collisions  collisions [%]  time [us]  duratio
          -----   --- ------- ------------ --------------- ---------- --------
  process_table     1   39419        17363         44.0473   12477844     1230
      run_queue    36  466820         9607          2.0580    2377975      234
   drv_ev_state    16  296173        11053          3.7319    1314002      129
        pollset     1   67534        23148         34.2761     765778       75
    proc_status 20894  356995          289          0.0810      51136        5
      proc_main 20894  391328         1937          0.4950      35799        3
      proc_link 20894  144304          365          0.2529      21964        2
       pix_lock  1024     415            3          0.7229      18001        1
      proc_msgq 20894  293336          424          0.1445      10521        1
   db_hash_slot  1344   77508           93          0.1200       3629        0
pollset_rm_list     1   13184           97          0.7357        102        0
       proc_btm 20894   68763           23          0.0334         80        0
         db_tab   120  428025            1          0.0002          1        0
```

EMULATOR TOOLBOX FOR PATHOLOGISTS

```
% $ERL_TOP/bin/cerl -rgdb -pa ebin -pa deps/*/ebin -pa test
Reading symbols from beam.smp...done.
%------------------------------------------------------------------------
% Use etp-help for a command overview and general help.
[...]
--------------- System Information ---------------
OTP release: 18
ERTS version: 7.2.1
Compile date: Wed Feb  3 08:37:09 2016
Arch: x86_64-unknown-linux-gnu
Endianness: Little
Word size: 64-bit
Halfword: no
HiPE support: no
SMP support: yes
Thread support: yes
Kernel poll: Supported
Debug compiled: no
Lock checking: no
Lock counting: no
System not initialized
--------------------------------------------------
(gdb)
```

INSPECTING NIF ARGUMENTS WITH ETP

```
4> erl_ip_index:lookup_ip([<<"bin">>, 42], {1,2,3,4}).

Breakpoint 1, lookup_ip_nif (env=env@entry=0x7ffff33bde20, argc=argc@entry=3,
187     {
(gdb) etp argv[0]
[#SubBinary<0x3,0,0x10000:0xd3a802a2>,42].
(gdb) etp argv[1]
16909060.
(gdb) etp-process-info env->proc
  Pid: <0.39.0>
  State: running | active | prq-prio-normal | usr-prio-normal | act-prio-norma
  Current function: erl_ip_index:lookup_subnet_nif/3
  CP: #Cp<erl_eval:do_apply/6+0x198>
  I: #Cp<code_server:call/2+0x90>
  Heap size: 610
  Old-heap size: 987
  Mbuf size: 0
  Msgq len: 0 (inner=0, outer=0)
  Parent: <0.27.0>
  Pointer: (Process *) 0x7ffff5d40378
```

PLACES YOU CAN'T USE A CUSTOM VM

- CI
- afl-fuzz
- production

IDEA: JUST ENOUGH NIF API

```
$ ./niffy ../jiffy/priv/jiffy.so -lv
../jiffy/priv/jiffy.so: jiffy 2.9
  nif_decode_init/2
  nif_decode_iter/5
  nif_encode_init/2
  nif_encode_iter/3
_ = erlang:load_nif(jiffy, []).
Term = jiffy:nif_decode_init(<<"[\"json\"]">>, []).
jiffy:nif_encode_init(Term, []).
```

CASE 1: VALIDATING CORRECTNESS

`erl_ip_index`

MEMORY LEAKS

```
==24893== HEAP SUMMARY:
==24893==     in use at exit: 42 bytes in 1 blocks
==24893==   total heap usage: 30 allocs, 29 frees, 7,673 bytes allocated
==24893==
==24893== 42 bytes in 1 blocks are definitely lost in loss record 1 of 1
==24893==    at 0x4C29C4F: malloc (vg_replace_malloc.c:299)
==24893==    by 0x405C73: enif_alloc (nif_stubs.c:849)
==24893==    by 0x57DE773: ???
==24893==    by 0x40350E: call (niffy.c:78)
==24893==    by 0x403B91: niffy_handle_statement (niffy.c:205)
==24893==    by 0x4121A1: yy_reduce (parse.y:56)
==24893==    by 0x4121A1: Parse (parse.c:1293)
==24893==    by 0x4031AB: main (main.c:100)
==24893==
==24893== LEAK SUMMARY:
==24893==    definitely lost: 42 bytes in 1 blocks
==24893==    indirectly lost: 0 bytes in 0 blocks
==24893==      possibly lost: 0 bytes in 0 blocks
==24893==    still reachable: 0 bytes in 0 blocks
==24893==         suppressed: 0 bytes in 0 blocks
```

GENERATE RANDOM INPUT, RUN UNDER VALGRIND IN CI

Generate random input, run under valgrind in CI

Sanitizers: asan, ubsan, etc

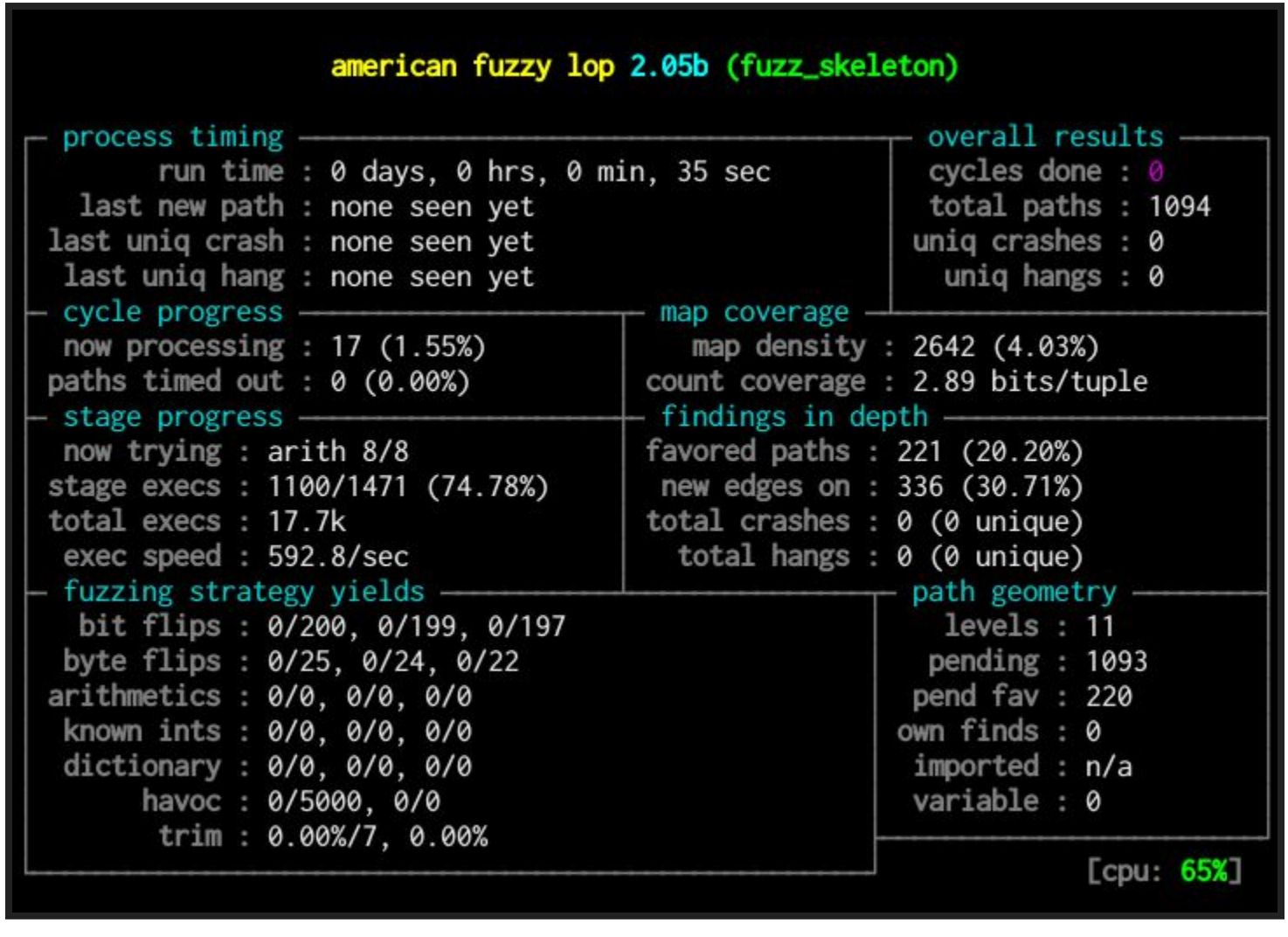
TRACING TESTS

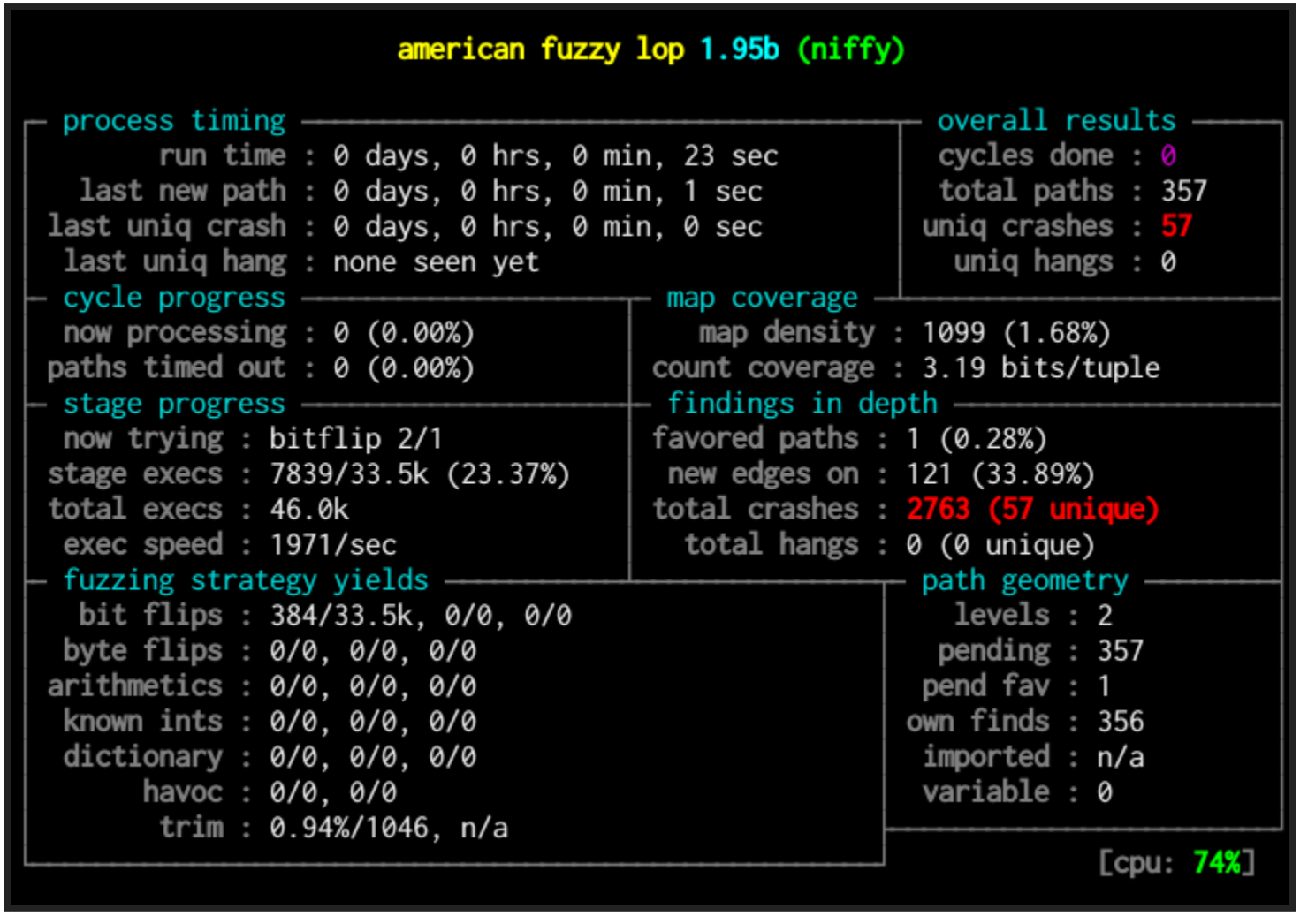
```
start() ->
    Port = open_port({spawn_executable, "/usr/bin/valgrind"},
                     [{args, ["--error-exitcode=42", "--",
                              "../niffy/niffy", "./priv/rtb_boolean_vm.so", "-
                      binary, stream, exit_status,
                      {line, 1024}]),
    Port ! {self(), {command, <<"_ = erlang:load_nif(rtb_boolean_vm, []).\n">>
    erlang:trace_pattern({rtb_boolean_vm, '_', '_'}, true, []),
    erlang:trace(all, true, [call]),
    loop(Port).

loop(Port) ->
    receive
        {trace, _, call, {Module, Function, Arguments}} ->
            Port ! {self(), {command, format_mfa(Module, Function, Arguments)}
            loop(Port);
        {_Port, {exit_status, Status}} ->
            io:format("niffy exited with code ~p~n", [Status])
    end.
```

AFL-FUZZ

```
CC=afl-gcc CXX=afl-g++ ./rebar co
```




american fuzzy lop 1.95b (niffy)
process timing
run time : 0 days, 0 hrs, 0 min, 23 sec
last new path : 0 days, 0 hrs, 0 min, 1 sec
last uniq crash : 0 days, 0 hrs, 0 min, 0 sec
last uniq hang : none seen yet
overall results
cycles done : 0
total paths : 357
uniq crashes : 57
uniq hangs : 0
cycle progress
now processing : 0 (0.00%)
paths timed out : 0 (0.00%)
map coverage
map density : 1099 (1.68%)
count coverage : 3.19 bits/tuple
stage progress
now trying : bitflip 2/1
stage execs : 7839/33.5k (23.37%)
total execs : 46.0k
exec speed : 1971/sec
findings in depth
favored paths : 1 (0.28%)
new edges on : 121 (33.89%)
total crashes : 2763 (57 unique)
total hangs : 0 (0 unique)
fuzzing strategy yields
bit flips : 384/33.5k, 0/0, 0/0
byte flips : 0/0, 0/0, 0/0
arithmetics : 0/0, 0/0, 0/0
known ints : 0/0, 0/0, 0/0
dictionary : 0/0, 0/0, 0/0
havoc : 0/0, 0/0
trim : 0.94%/1046, n/a
path geometry
levels : 2
pending : 357
pend fav : 1
own finds : 356
imported : n/a
variable : 0
[cpu: 74%]

CASE 2: FINDING AN ELUSIVE BUG

rtb-boolean

```
===> Performing EUnit tests...
..............Makefile:27: recipe for target 'eunit' failed
make: *** [eunit] Segmentation fault
```

REPRODUCTION

```
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x00000000500000004 in ?? ()
[Current thread is 1 (Thread 0x7f12cd53e700 (LWP 24586))]
(gdb) bt
#0  0x00000000500000004 in ?? ()
#1  0x00007f12d2097377 in _dl_close_worker (map=<optimized out>) at dl-close.c
#2  0x00007f12d2097e2c in _dl_close (_map=0x7f12c40029c0) at dl-close.c:775
#3  0x00007f12d2092114 in _dl_catch_error (objname=0x7f12bc002920, errstring=0
    args=0x7f12c40029c0) at dl-error.c:187
#4  0x00007f12d1c7d4d9 in _dlerror_run (operate=operate@entry=0x7f12d1c7cfd0 <
#5  0x00007f12d1c7cfff in __dlclose (handle=<optimized out>) at dlclose.c:46
#6  0x00007f12c813ecf6 in boolean_index_type_destructor (env=0x7f12cd53d8f0, c
    at /home/julian/work/rtb-gateway/_build/default/lib/rtb_boolean/c_src/rtb_
#7  0x00000000054f277 in ?? ()
#8  0x00000000005255eb in ?? ()
#9  0x00000000005274f8 in erts_garbage_collect ()
#10 0x00000000004462ea in process_main ()
#11 0x00000000004bf07d in ?? ()
#12 0x00000000005d73f7 in ?? ()
#13 0x00007f12d131c284 in start_thread (arg=0x7f12cd53e700) at pthread_create.
#14 0x00007f12d0e5197d in clone () at ../sysdeps/unix/sysv/linux/x86_64/clone.
```

UNDER VALGRIND WITH NIFFY

```
==24431== Invalid read of size 4
==24431==    at 0x5DEA92E: resolve_partial_index (boolean-runtime.c:519)
==24431==    by 0x5DEA9EF: resolve_partial_index_int (boolean-runtime.c:555)
==24431==    by 0x5DEA9EF: resolve_partial_index (boolean-runtime.c:520)
==24431==    by 0x5DEA9EF: resolve_partial_index_int (boolean-runtime.c:555)
==24431==    by 0x5DEA9EF: resolve_partial_index (boolean-runtime.c:520)
==24431==    by 0x5DEA639: resolve_index (boolean-runtime.c:513)
==24431==    by 0x5DEA184: evaluate_in (booleans.c:495)
==24431==    by 0x5DEA184: evaluate (booleans.c:515)
==24431==    by 0x57DF0A4: evaluate (rtb_boolean_vm.c:127)
==24431==    by 0x402E6E: call (niffy.c:78)
==24431==    by 0x4034F1: niffy_handle_statement (niffy.c:205)
==24431==    by 0x40A4F8: yy_reduce (parse.y:56)
==24431==    by 0x40B026: Parse (parse.c:1293)
==24431==    by 0x402B0B: main (main.c:100)
==24431==  Address 0x15000000b0 is not stack'd, malloc'd or (recently) free'd
```

CASE 3: PROFILING

`jiffy`

VARIANCE

```
12> erljson_bench:main(["lots-of-doubles-3.term", "100000", "1"]).
document: lots-of-doubles-3.term; iterations: 100000; workers: 1.
        module:       99th usecs       max usecs        variance
modified_jiffy:               46             683              41
    ejson_test:               84             749              19
         jiffy:               87            2425             113
         jsonx:               97             748              25
          json:              127             808              50

13> erljson_bench:main(["lots-of-doubles-3.term", "100000", "100"]).
document: lots-of-doubles-3.term; iterations: 100000; workers: 100.
        module:       99th usecs       max usecs        variance
modified_jiffy:               53          188327         6507054
         jiffy:              124          416312        32076341
    ejson_test:              148         1247405      1081107002
          json:              210         1937470      1984462505
         jsonx:              234         2388897      2541644970
```

IDEA: CATCHING SLOW ENIF_*() CALLS

```
#define TRACE_INT(f, prelude, postlude, ...)                                  \
  ({ uint64_t t0 = __rdtsc();                                                 \
    prelude f(__VA_ARGS__);                                                   \
    uint64_t dt = __rdtsc() - t0;                                             \
    if (dt > SLOW_ENIF_THRESHOLD) {                                           \
      fprintf(stderr, "%s call to " STRINGIFY(f) "(" STRINGIFY(__VA_ARGS__) \
              ") too slow: %lu\n", __FUNCTION__, dt);                         \
    }                                                                         \
    postlude })

#define TRACE(f, ...) TRACE_INT(f, __auto_type r =, r;, __VA_ARGS__)

#define enif_priv_data(...) TRACE(enif_priv_data, __VA_ARGS__)
#define enif_alloc(...) TRACE(enif_alloc, __VA_ARGS__)
#define enif_free(...) TRACE(enif_free, __VA_ARGS__)
#define enif_is_atom(...) TRACE(enif_is_atom, __VA_ARGS__)
#define enif_is_binary(...) TRACE(enif_is_binary, __VA_ARGS__)
[...]
```

CACHEGRIND

```
% valgrind --tool=cachegrind ./niffy ../jiffy/priv/jiffy.so -l < jiffy-1.in >/
==27550==
==27550== I   refs:      6,290,708
==27550== I1  misses:        2,364
==27550== LLi misses:        1,761
==27550== I1  miss rate:      0.04%
==27550== LLi miss rate:      0.03%
==27550==
==27550== D   refs:      3,071,530  (2,328,421 rd   + 743,109 wr)
==27550== D1  misses:       26,510  (   19,176 rd   +   7,334 wr)
==27550== LLd misses:       13,664  (    7,961 rd   +   5,703 wr)
==27550== D1  miss rate:       0.9% (      0.8%     +     1.0%  )
==27550== LLd miss rate:       0.4% (      0.3%     +     0.8%  )
==27550==
==27550== LL refs:          28,874  (   21,540 rd   +   7,334 wr)
==27550== LL misses:        15,425  (    9,722 rd   +   5,703 wr)
==27550== LL miss rate:        0.2% (      0.1%     +     0.8%  )
```

TANGENTIALLY-RELATED QUOTE

[…] Each micro-optimization might improve the performance by as little as 0.05%. If we get one that improves performance by 0.25%, that is considered a huge win. Each of these optimizations is unmeasurable on a real-world system (we have to use cachegrind to get repeatable run-times) but if you do enough of them, they add up.

– Richard D. Hipp

NEXT STEPS

contribute: `github.com/tokenrove/niffy`

feedback: `julian@cipht.net`