THINK OUTSIDE THE VM: UNOBTRUSIVE MEASUREMENT

JULIAN SQUIRES

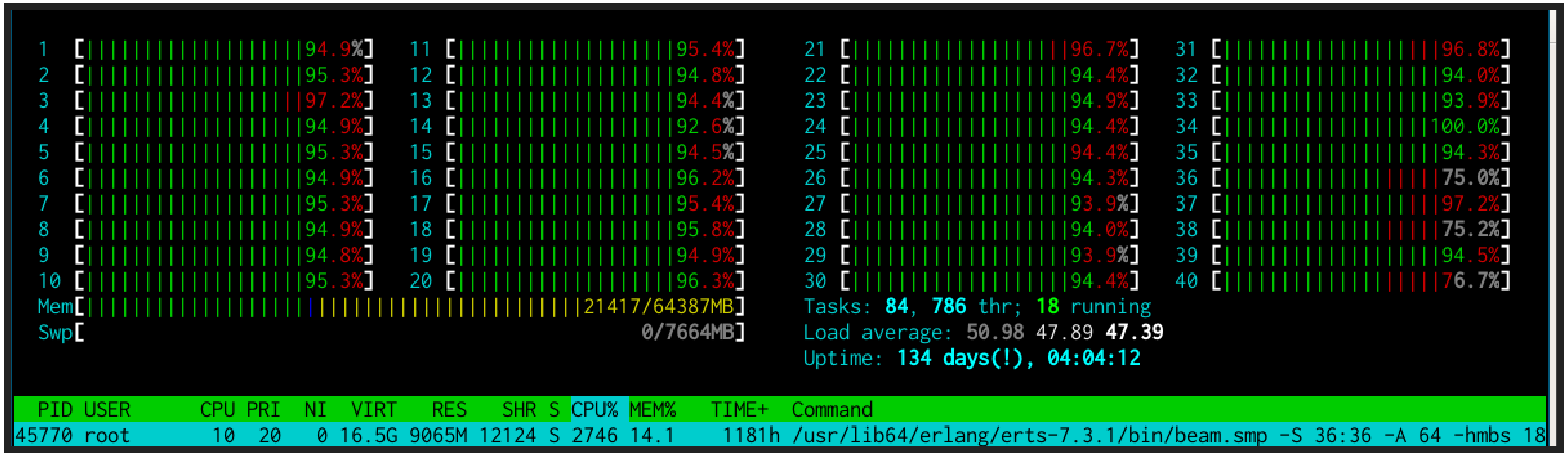
INSIDE THE VM

In-VM tools:

- fprof
- eep
- eflame

All built on tracing

PROFILING IN PROD IS HARD



```
  1  [|||||||||||||||||||94.9%]   11 [|||||||||||||||||||95.4%]    21 [|||||||||||||||||||96.7%]   31 [|||||||||||||||||||96.8%]
  2  [|||||||||||||||||||95.3%]   12 [|||||||||||||||||||94.8%]    22 [|||||||||||||||||||94.4%]   32 [|||||||||||||||||||94.0%]
  3  [|||||||||||||||||||97.2%]   13 [|||||||||||||||||||94.4%]    23 [|||||||||||||||||||94.9%]   33 [|||||||||||||||||||93.9%]
  4  [|||||||||||||||||||94.9%]   14 [|||||||||||||||||||92.6%]    24 [|||||||||||||||||||94.4%]   34 [||||||||||||||||||100.0%]
  5  [|||||||||||||||||||95.3%]   15 [|||||||||||||||||||94.5%]    25 [|||||||||||||||||||94.4%]   35 [|||||||||||||||||||94.3%]
  6  [|||||||||||||||||||94.9%]   16 [|||||||||||||||||||96.2%]    26 [|||||||||||||||||||94.3%]   36 [|||||||||||||||||||75.0%]
  7  [|||||||||||||||||||95.3%]   17 [|||||||||||||||||||95.4%]    27 [|||||||||||||||||||93.9%]   37 [|||||||||||||||||||97.2%]
  8  [|||||||||||||||||||94.9%]   18 [|||||||||||||||||||95.8%]    28 [|||||||||||||||||||94.0%]   38 [|||||||||||||||||||75.2%]
  9  [|||||||||||||||||||94.8%]   19 [|||||||||||||||||||94.9%]    29 [|||||||||||||||||||93.9%]   39 [|||||||||||||||||||94.5%]
  10 [|||||||||||||||||||95.3%]   20 [|||||||||||||||||||96.3%]    30 [|||||||||||||||||||94.4%]   40 [|||||||||||||||||||76.7%]
  Mem[|||||||||||||||||||||||||||||||||||||||||||||21417/64387MB]    Tasks: 84, 786 thr; 18 running
  Swp[                                                 0/7664MB]    Load average: 50.98 47.89 47.39
                                                                    Uptime: 134 days(!), 04:04:12

  PID USER      CPU PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
45770 root       10  20   0 16.5G 9065M 12124 S 2746 14.1   1181h /usr/lib64/erlang/erts-7.3.1/bin/beam.smp -S 36:36 -A 64 -hmbs 18
```

(although tracing is getting safer: see Lukas Larsson's talk)

HOW FAR OUTSIDE THE VM CAN WE GO?

Callan, Robert, et al. "Zero-overhead profiling via EM emanations." Proceedings of the 25th International Symposium on Software Testing and Analysis. ACM, 2016.

- or hardware-assisted profiling (not just performance counters)

USING OUT-OF-VM TOOLS ON THE VM

In order of obtrusiveness:

- ptrace
- ftrace
- systemtap
- `perf_events`

PTRACE, `/PROC/PID/MEM`

- used by gdb, strace
- has to stop processes to read from them
- interferes badly with systems with tight latency requirements

FTRACE

For system calls, can be less obtrusive than `strace`.

One application: tracing mmap syscalls to compare with allocator stats to detect fragmentation.

SYSTEMTAP

- scripting language → kernel module
- complicated
- has throttling, but didn't work well for me

Build VM with `--with-dynamic-trace=systemtap`

STAP: GC PROBES

```
/**
 * Fired when a major GC is starting.
 *
 * @param p the PID (string form) of the exiting process
 * @param need the number of words needed on the heap
 */
probe gc_major__start(char *p, int need);

/**
 * Fired when a minor GC is starting.
 *
 * @param p the PID (string form) of the exiting process
 * @param need the number of words needed on the heap
 */
probe gc_minor__start(char *p, int need);
```

STAP: PROCESS HEAP CHANGES

`lib/runtime_tools/examples/memory1.systemtap`

```
probe process("beam").mark("process-heap_grow")
{
    printf("proc heap grow pid %s %d -> %d bytes\n", user_string($arg1),
           $arg2, $arg3);
}

probe process("beam").mark("process-heap_shrink")
{
    printf("proc heap shrink pid %s %d -> %d bytes\n", user_string($arg1),
           $arg2, $arg3);
}
```

`PERF_EVENTS`

- originally for reading performance counters
- grew to sample registers and stack from kernel
- designed to be safe to use in production
- scales itself back if it takes too much time

PERF HAS OVERHEAD, TOO

V.M. Weaver. "Self-monitoring Overhead of the Linux `perf_event` Performance Counter Interface", IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2015), Philadelphia, Pennsylvania, March 2015.

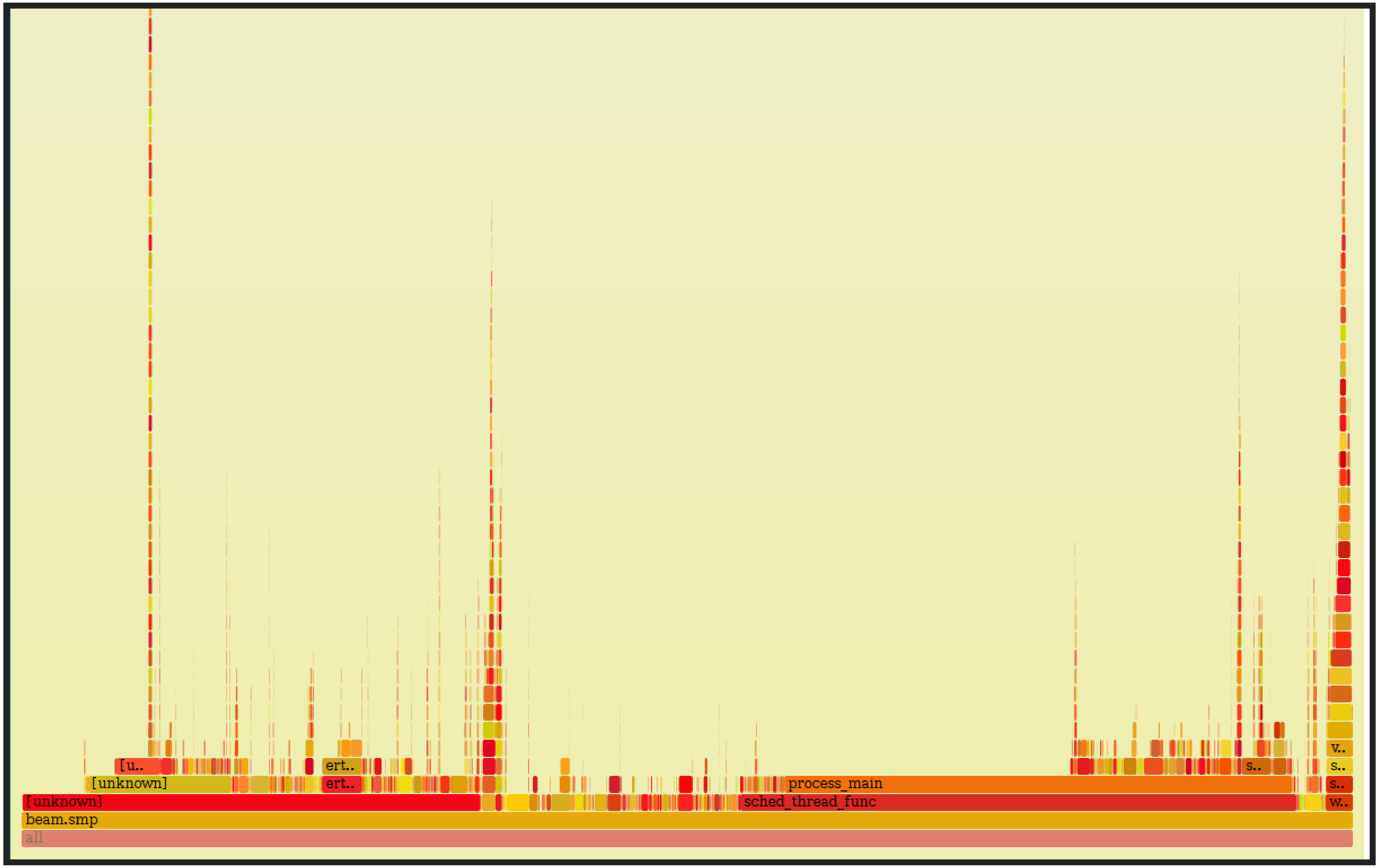
INFORMATION WE CAN GET JUST FROM EXISTING TOOLS

- native traces are still useful; tell us a lot about the workload
- `perf` is great; many modes
- Brendan Gregg's `perf-tools`
- Andi Kleen's `pmutools`

PERF TOP

```
16.61%  beam.smp                                        [.] process_main
 3.00%  beam.smp                                        [.] 0x0000000000d9a
 2.92%  beam.smp                                        [.] 0x0000000000d9a
 2.07%  beam.smp                                        [.] copy_shallow
 1.78%  beam.smp                                        [.] schedule
 1.63%  booleans_1473100378-465242-576460752303408164.so [.] evaluate
 1.48%  beam.smp                                        [.] copy_struct
 1.15%  jiffy.so                                        [.] encode_iter
 1.01%  libpthread-2.21.so                              [.] pthread_mutex_lo
 0.96%  beam.smp                                        [.] erts_garbage_col
 0.91%  beam.smp                                        [.] erts_alcu_check_
 0.89%  beam.smp                                        [.] eq
 0.84%  beam.smp                                        [.] size_object
 0.84%  beam.smp                                        [.] erts_alcu_alloc_
 0.81%  beam.smp                                        [.] erts_alcu_free_t
 0.78%  beam.smp                                        [.] db_get_hash
 0.74%  libpthread-2.21.so                              [.] pthread_getspeci
 0.72%  libc-2.21.so                                    [.] vfprintf
 0.70%  beam.smp                                        [.] 0x0000000000d9a
 0.57%  [kernel]                                        [k] system_call
 0.53%  beam.smp                                        [.] 0x0000000000d9a
 0.48%  libpthread-2.21.so                              [.] __pthread_mutex_
```

FLAME GRAPHS



SEE ALSO

- Brendan Gregg
  - http://www.brendangregg.com/linuxperf.html
- Gil Tene
  - http://stuff-gil-says.blogspot.ca/

DISCLAIMER: HACKS AHEAD

HOW CAN WE GET ERLANG STACK TRACES INTERMIXED WITH NATIVE ONES?

HOW CAN WE GET ERLANG STACK TRACES INTERMIXED WITH NATIVE ONES?

- sample registers and stack (`perf_events`);
- unwind till we find `process_main()` (`elfutils`);
- DWARF info (or perf sample) gives us registers that correspond to process, BEAM instruction;
- (maybe) walk process's stack exactly as `etp` does.

PROCESS_VM_READV

- reads from another process's memory without stopping it
- unsafe (racy), but unobtrusive

DWARF

- allows us to peek into ERTS at the C level
- libraries aren't great
- compilers are inconsistent in what they omit
- some systems strip by default (e.g. Gentoo)
- sometimes we have to look at the dissassembly by hand to pick out the registers we want

DWARF: LOCAL VARIABLES

```
 [  b94e]    subprogram
             name                     (strp) "process_main"
[...]
             low_pc                   (addr) 0x000000000043de00 <process_main>
             high_pc                  (data8) 47338 (0x00000000004496ea)
[...]
 [  b982]      variable
               name                     (string) "c_p"
               decl_file                (data1) 1
               decl_line                (data2) 1129
               type                     (ref4) [  5d64]
 [  b98e]      variable
               name                     (strp) "reds_used"
               decl_file                (data1) 1
               decl_line                (data2) 1130
               type                     (ref4) [  3d95]
               location                 (exprloc)
                [   0] reg12
 [  b99c]      variable
               name                     (string) "x0"
               decl_file                (data1) 1
               decl_line                (data2) 1138
               type                     (ref4) [  461b]
               location                 (exprloc)
                [   0] reg15
```

DWARF: UNWIND INFORMATION

```
[   7f8] FDE length=68 cie=[    30]
  CIE_pointer:             1996
  initial_location:        0x000000000043de00 <process_main> (offset: 0x3de0
  address_range:           0xb8ea (end offset: 0x496ea)

  Program:
    advance_loc 5 to 0x3de05
    def_cfa r10 (reg10) at offset 0
    advance_loc 9 to 0x3de0e
    expression r6 (reg6)
         [   0] breg6 0
    advance_loc 13 to 0x3de1b
    def_cfa_expression 3
         [   0] breg6 -40
         [   2] deref
    expression r15 (reg15)
         [   0] breg6 -8
    expression r14 (reg14)
         [   0] breg6 -16
    expression r13 (reg13)
         [   0] breg6 -24
    expression r12 (reg12)
         [   0] breg6 -32
    advance_loc 8 to 0x3de23
    expression r3 (reg3)
         [   0] breg6 -48
    advance_loc2 47258 to 0x496bd
```

```
advance_loc2 47258 to 0x496bd

remember_state
def_cfa r10 (reg10) at offset 0
advance_loc 13 to 0x496ca
def_cfa r7 (reg7) at offset 8
advance_loc 1 to 0x496cb
restore_state
```

STRUCT PROCESS

```
struct process {
    ErtsPTabElementCommon common; /* *Need* to be first in struct */
    [...]
    Eterm* htop;                /* Heap top */
    Eterm* stop;                /* Stack top */
    Eterm* heap;                /* Heap start */
    Eterm* hend;                /* Heap end */
    Uint heap_sz;               /* Size of heap in words */
    Uint min_heap_size;         /* Minimum size of heap (in words). */
    Uint min_vheap_size;        /* Minimum size of virtual heap (in words). */
    [...]
    Uint arity;                 /* Number of live argument registers (only val
                                 * when process is *not* running).
                                 */
    Eterm* arg_reg;             /* Pointer to argument registers. */
    unsigned max_arg_reg;       /* Maximum number of argument registers availa
    Eterm def_arg_reg[6];       /* Default array for argument registers. */

    BeamInstr* cp;              /* (untagged) Continuation pointer (for thread
    BeamInstr* i;               /* Program counter for threaded code. */
    Sint catches;               /* Number of catches on stack */
    Sint fcalls;                /*
                                 * Number of reductions left to execute.
                                 * Only valid for the current process.
                                 */
    Uint32 rcount;              /* suspend count */
    int  schedule_count;        /* Times left to reschedule a low prio process
    int  schedule_count;        /* Times left to reschedule a low prio process

    Uint reds;                  /* No of reductions for this process  */
    Eterm group_leader;         /* Pid in charge
                                   (can be boxed) */
    Uint flags;                 /* Trap exit, etc (no trace flags anymore) */
    Eterm fvalue;               /* Exit & Throw value (failure reason) */
    Uint freason;               /* Reason for detected failure */
    Eterm ftrace;               /* Latest exception stack trace dump */

    Process *next;              /* Pointer to next process in run queue */

    struct ErtsNodesMonitor_ *nodes_monitors;

    ErtsSuspendMonitor *suspend_monitors; /* Processes suspended by
                                             this process via
                                             erlang:suspend_process/1 */

    ErlMessageQueue msg;        /* Message queue */

    ErtsBifTimers *bif_timers;  /* Bif timers aiming at this process */
#ifdef ERTS_BTM_ACCESSOR_SUPPORT
    ErtsBifTimers *accessor_bif_timers; /* Accessor bif timers */
#endif

    ProcDict  *dictionary;       /* Process dictionary, may be NULL */

    Uint seq_trace_clock;
    Uint seq_trace_lastcnt;
    Eterm seq_trace_token;      /* Sequential trace token (tuple size 5 see be

#ifdef USE_VM_PROBES
```

```
#ifdef USE_VM_PROBES

    Eterm dt_utag;              /* Place to store the dynamc trace user tag */
    Uint dt_utag_flags;         /* flag field for the dt_utag */
#endif
    union {
        void *terminate;
        BeamInstr initial[3];   /* Initial module(0), function(1), arity(2), c
                                   of pointer to funcinfo instruction, hence t
    } u;
    BeamInstr* current;         /* Current Erlang function, part of the funcin
                                 * module(0), function(1), arity(2)
                                 * (module and functions are tagged atoms;
                                 * arity an untagged integer). BeamInstr * bec
                                 */


    /*
     * Information mainly for post-mortem use (erl crash dump).
     */
    Eterm parent;               /* Pid of process that created this process. *
    erts_approx_time_t approx_started; /* Time when started. */

    Uint32 static_flags;        /* Flags that do *not* change */

    /* This is the place, where all fields that differs between memory
     * architectures, have gone to.
     */

    Eterm *high_water;
    Eterm *old_hend;            /* Heap pointers for generational GC. */
    Eterm *old_htop;
    Eterm *old_heap;
    Eterm *old_heap;

    Uint16 gen_gcs;             /* Number of (minor) generational GCs. */
    Uint16 max_gen_gcs;         /* Max minor gen GCs before fullsweep. */
    ErlOffHeap off_heap;        /* Off-heap data updated by copy_struct(). */
    ErlHeapFragment* mbuf;      /* Pointer to message buffer list */
    Uint mbuf_sz;               /* Size of all message buffers */
    ErtsPSD *psd;               /* Rarely used process specific data */

    Uint64 bin_vheap_sz;        /* Virtual heap block size for binaries */
    Uint64 bin_vheap_mature;    /* Virtual heap block size for binaries */
    Uint64 bin_old_vheap_sz;    /* Virtual old heap block size for binaries */
    Uint64 bin_old_vheap;       /* Virtual old heap size for binaries */

    ErtsProcSysTaskQs *sys_task_qs;

    erts_smp_atomic32_t state;  /* Process state flags (see ERTS_PSFLG_*) */

#ifdef ERTS_SMP
    ErlMessageInQueue msg_inq;
    ErtsPendExit pending_exit;
    erts_proc_lock_t lock;
    ErtsSchedulerData *scheduler_data;
    Eterm suspendee;
    ErtsPendingSuspend *pending_suspenders;
    erts_smp_atomic_t run_queue;
#ifdef HIPE
    struct hipe_process_state_smp hipe_smp;
#endif
#endif

#ifdef CHECK_FOR_HOLES
```

```
#ifdef CHECK_FOR_HOLES

    Eterm* last_htop;           /* No need to scan the heap below this point.
    ErlHeapFragment* last_mbuf; /* No need to scan beyond this mbuf. */
#endif

#ifdef DEBUG
    Eterm* last_old_htop;       /*
                                 * No need to scan the old heap below this poi
                                 * when looking for invalid pointers into the
                                 * heap fragments.
                                 */
#endif

#ifdef FORCE_HEAP_FRAGS
    Uint space_verified;        /* Avoid HAlloc forcing heap fragments when */
    Eterm* space_verified_from; /* we rely on available heap space (TestHeap)
#endif
};
```

PROCESS_MAIN

```
(gdb) disassemble/m process_main
Dump of assembler code for function process_main:
1128            static int init_done = 0;
1129            Process* c_p = NULL;
1130            int reds_used;
[...]
1145            /*
1146             * Top of heap (next free location); grows upwards.
1147             */
1148            register Eterm* HTOP REG_htop = NULL;
1149
1150            /* Stack pointer.  Grows downwards; points
1151             * to last item pushed (normally a saved
1152             * continuation pointer).
1153             */
1154            register Eterm* E REG_stop = NULL;
1155
1156            /*
1157             * Pointer to next threaded instruction.
1158             */
1159            register BeamInstr *I REG_I = NULL;
1160
```

PROCESS_MAIN

```
1161        /* Number of reductions left.  This function
1162         * returns to the scheduler when FCALLS reaches zero.
1163         */
1164        register Sint FCALLS REG_fcalls = 0;
[...]
1311            SWAPIN;
   0x000000000043e015 <+533>:   mov    0x48(%r13),%r11
   0x000000000043e01c <+540>:   mov    0x50(%r13),%r10
   0x000000000043e020 <+544>:   jmpq   *(%rbx)
   0x000000000043e022 <+546>:   lea    0x2c8(%r13),%rdx
```

FINDING I

`objdump -d -S beam.smp`:

```
       I = handle_error(c_p, I, reg, NULL);
43e16b:       48 89 de                mov    %rbx,%rsi
43e16e:       4c 89 f2                mov    %r14,%rdx
43e171:       4c 89 ef                mov    %r13,%rdi
43e174:       e8 27 f2 ff ff          callq  43d3a0 <handle_error.constprop.3>
43e179:       48 89 c3                mov    %rax,%rbx
```

GENERATING A PERF.MAP

```
7fe15be4fc48 a8 cowboy:start_http/4
7fe15be4fcf0 a8 cowboy:start_https/4
7fe15be4fd98 e8 cowboy:start_spdy/4
7fe15be4fe80 38 cowboy:stop_listener/1
7fe15be4feb8 250 cowboy:set_env/3
7fe15be50108 68 cowboy:module_info/0
7fe15be50170 78 cowboy:module_info/1
7fe15be50ee0 38 cowboy_app:start/2
7fe15be50f18 38 cowboy_app:stop/1
7fe15be50f50 68 cowboy_app:module_info/0
7fe15be50fb8 78 cowboy_app:module_info/1
```

erts/emulator/beam/beam_ranges.c:

```
/*
 * The following variables keep a sorted list of address ranges for
 * each module.  It allows us to quickly find a function given an
 * instruction pointer.
 */
struct ranges {
    Range* modules;            /* Sorted lists of module addresses. */
    Sint n;                    /* Number of range entries. */
    Sint allocated;            /* Number of allocated entries. */
    erts_smp_atomic_t mid;     /* Cached search start point */
};
static struct ranges r[ERTS_NUM_CODE_IX];
```

erts/emulator/beam/code_ix.c:

```
erts_smp_atomic32_t the_active_code_index;
```

SAMPLE

```
processed 130287/134895 samples (96.584%)
19402            scheduler_wait                                         14.383
4218             sweep_one_area                                         3.1268
4103             erts_garbage_collect                                   3.0416
4035             copy_shallow                                           2.9912
3042             schedule                                               2.2550
2680             erts_cmp_compound                                      1.9867
2227             evaluate                                               1.6509
2086             erts_get_scheduler_data                                1.5463
2027             copy_struct                                            1.5026
1699             pthread_mutex_lock                                     1.2595
1499             enc_string                                             1.1112
1407             encode_iter                                            1.0430
1236             aoff_link_free_block                                   0.9162
[...]
543              timer:now_diff/2                                       0.4025
498              erts_cleanup_offheap                                   0.3691
491              db_put_hash                                            0.3639
491              statsderl:maybe_cast/4                                 0.3639
471              findTldNode                                            0.3491
470              __sched_yield                                          0.3484
465              make_hash2                                             0.3447
442              do_binary_match_compile                                0.3276
440              enif_get_list_cell                                     0.3261
```

PSTACK

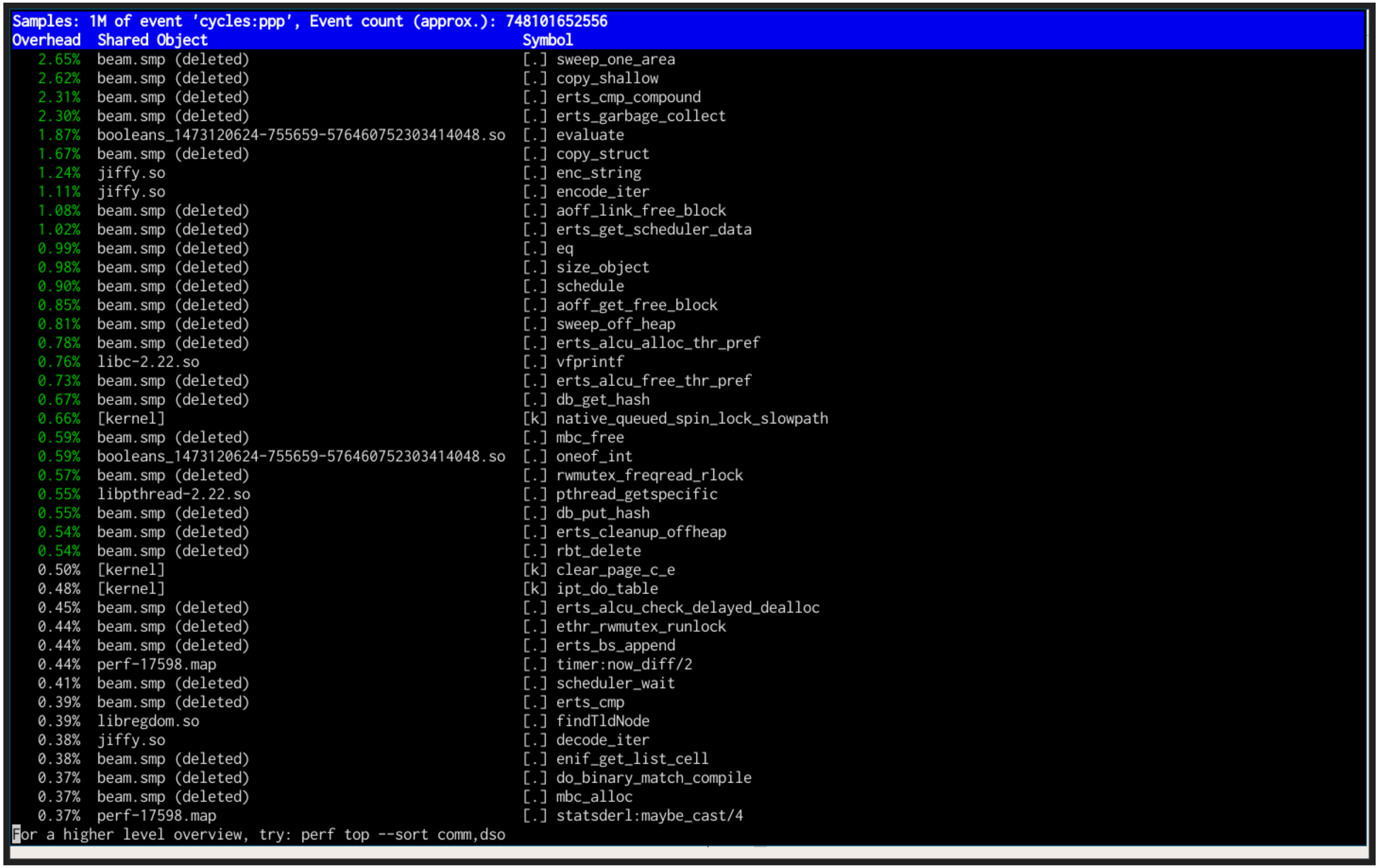
```
Stack for 5228:
       1  rtb_boolean:evaluate/2 () [7f1cce21cac8]
       2  flights_matcher:match_evaluate/4 () [7f1ccc3491a0]
       3  flights_matcher:-match_impression/4-lc$^0/1-0-/4 () [7f1ccc34c6a8]
       4  flights_matcher:match_impression/4 () [7f1ccc349098]
       5  flights_matcher:-match/5-lc$^0/1-0-/4 () [7f1ccc34cc00]
       6  flights_matcher:match/5 () [7f1ccc348d98]
       7  flights:match/4 () [7f1c26dd2f88]
       8  rtb_gateway_exchange:filter_flights_creatives/2 () [7f1c12c44078]
       9  rtb_gateway_exchange:generate_bid_request/2 () [7f1c12c47870]
      10  rtb_gateway_exchange:async_request/2 () [7f1c12c41fb8]
      12  sched_thread_func (/usr/lib64/erlang/erts-7.3.1/bin/beam.smp) [4d8e0
      13  thr_wrapper (/usr/lib64/erlang/erts-7.3.1/bin/beam.smp) [63ed73]
      14  start_thread (/lib64/libpthread-2.22.so) [7f1cd19ff494]
      15  __clone (/lib64/libc-2.22.so) [7f1cd153c5dd]

[...]

Stack for 5233:
       0  cowboy_protocol:parse_hd_name/8 () [7f1ccc307878]
       1  rtb_gateway_exchange:request/2 () [7f1c12c40ab8]
       2  rtb_gateway_request_handler:request/4 () [7f1ccd6c44a0]
       3  cowboy_handler:handler_handle/4 () [7f1ccd441050]
       4  cowboy_protocol:execute/4 () [7f1ccc30b9a8]
       6  sched_thread_func (/usr/lib64/erlang/erts-7.3.1/bin/beam.smp) [4d8e0
       7  thr_wrapper (/usr/lib64/erlang/erts-7.3.1/bin/beam.smp) [63ed73]
```

```
8  start_thread (/lib64/libpthread-2.22.so) [7f1cd19ff494]
9  __clone (/lib64/libc-2.22.so) [7f1cd153c5dd]
```

HOW BAD IS THE SKID?

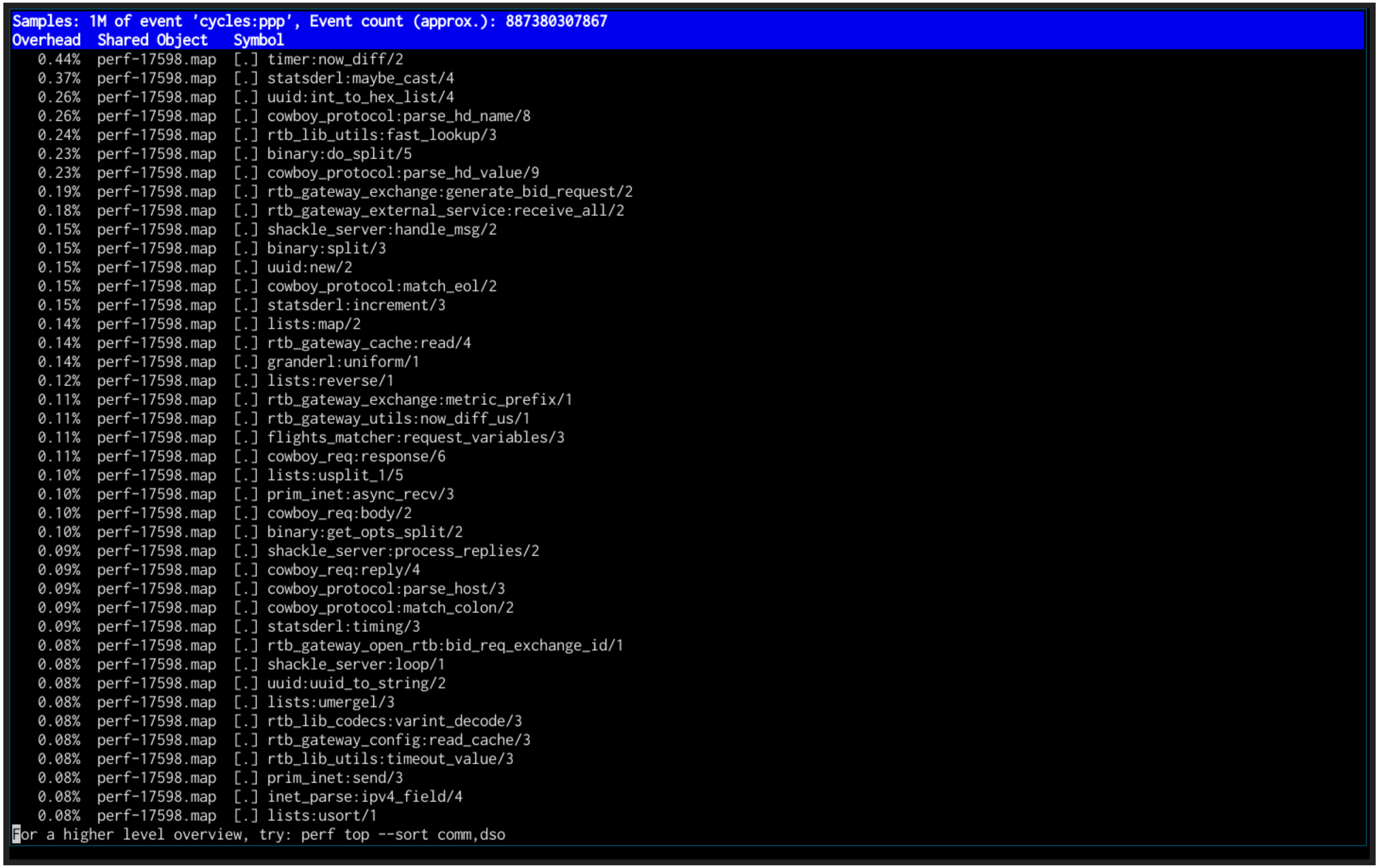
- can range from a few hundred microseconds to several seconds (!)
- still indicative if your workload is divided into many small processes
- probably wise to discard samples older than a millisecond.

PERF TOP WITH ERLANG SYMBOLS



```
Samples: 1M of event 'cycles:ppp', Event count (approx.): 748101652556
Overhead  Shared Object                                     Symbol
   2.65%  beam.smp (deleted)                                [.] sweep_one_area
   2.62%  beam.smp (deleted)                                [.] copy_shallow
   2.31%  beam.smp (deleted)                                [.] erts_cmp_compound
   2.30%  beam.smp (deleted)                                [.] erts_garbage_collect
   1.87%  booleans_1473120624-755659-576460752303414048.so  [.] evaluate
   1.67%  beam.smp (deleted)                                [.] copy_struct
   1.24%  jiffy.so                                          [.] enc_string
   1.11%  jiffy.so                                          [.] encode_iter
   1.08%  beam.smp (deleted)                                [.] aoff_link_free_block
   1.02%  beam.smp (deleted)                                [.] erts_get_scheduler_data
   0.99%  beam.smp (deleted)                                [.] eq
   0.98%  beam.smp (deleted)                                [.] size_object
   0.90%  beam.smp (deleted)                                [.] schedule
   0.85%  beam.smp (deleted)                                [.] aoff_get_free_block
   0.81%  beam.smp (deleted)                                [.] sweep_off_heap
   0.78%  beam.smp (deleted)                                [.] erts_alcu_alloc_thr_pref
   0.76%  libc-2.22.so                                      [.] vfprintf
   0.73%  beam.smp (deleted)                                [.] erts_alcu_free_thr_pref
   0.67%  beam.smp (deleted)                                [.] db_get_hash
   0.66%  [kernel]                                          [k] native_queued_spin_lock_slowpath
   0.59%  beam.smp (deleted)                                [.] mbc_free
   0.59%  booleans_1473120624-755659-576460752303414048.so  [.] oneof_int
   0.57%  beam.smp (deleted)                                [.] rwmutex_freqread_rlock
   0.55%  libpthread-2.22.so                                [.] pthread_getspecific
   0.55%  beam.smp (deleted)                                [.] db_put_hash
   0.54%  beam.smp (deleted)                                [.] erts_cleanup_offheap
   0.54%  beam.smp (deleted)                                [.] rbt_delete
   0.50%  [kernel]                                          [k] clear_page_c_e
   0.48%  [kernel]                                          [k] ipt_do_table
   0.45%  beam.smp (deleted)                                [.] erts_alcu_check_delayed_dealloc
   0.44%  beam.smp (deleted)                                [.] ethr_rwmutex_runlock
   0.44%  beam.smp (deleted)                                [.] erts_bs_append
   0.44%  perf-17598.map                                    [.] timer:now_diff/2
   0.41%  beam.smp (deleted)                                [.] scheduler_wait
   0.39%  beam.smp (deleted)                                [.] erts_cmp
   0.39%  libregdom.so                                      [.] findTldNode
   0.38%  jiffy.so                                          [.] decode_iter
   0.38%  beam.smp (deleted)                                [.] enif_get_list_cell
   0.37%  beam.smp (deleted)                                [.] do_binary_match_compile
   0.37%  beam.smp (deleted)                                [.] mbc_alloc
   0.37%  perf-17598.map                                    [.] statsderl:maybe_cast/4
For a higher level overview, try: perf top --sort comm,dso
```

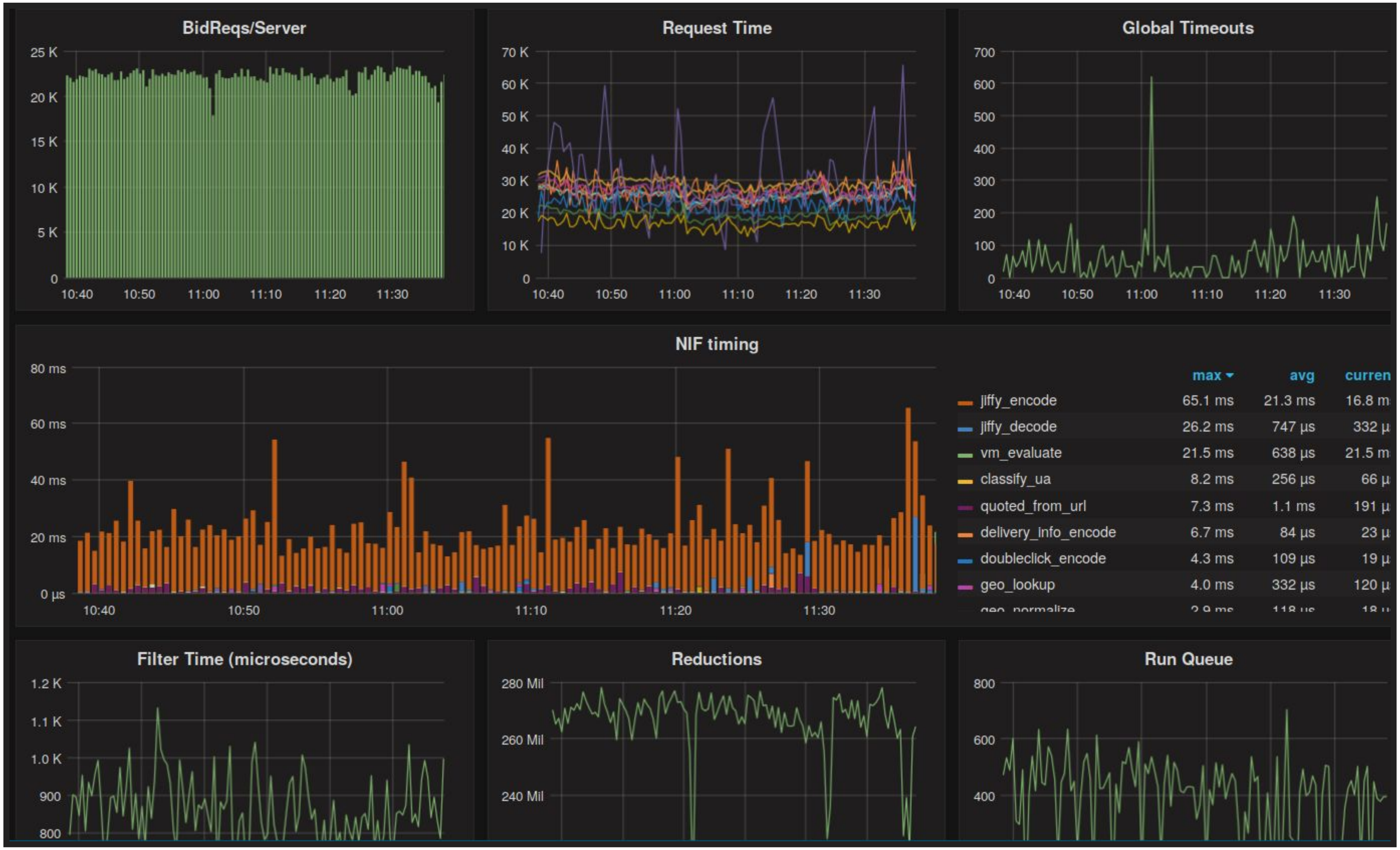
PERF TOP WITH ERLANG SYMBOLS



```
Samples: 1M of event 'cycles:ppp', Event count (approx.): 887380307867
Overhead  Shared Object   Symbol
   0.44%  perf-17598.map  [.] timer:now_diff/2
   0.37%  perf-17598.map  [.] statsderl:maybe_cast/4
   0.26%  perf-17598.map  [.] uuid:int_to_hex_list/4
   0.26%  perf-17598.map  [.] cowboy_protocol:parse_hd_name/8
   0.24%  perf-17598.map  [.] rtb_lib_utils:fast_lookup/3
   0.23%  perf-17598.map  [.] binary:do_split/5
   0.23%  perf-17598.map  [.] cowboy_protocol:parse_hd_value/9
   0.19%  perf-17598.map  [.] rtb_gateway_exchange:generate_bid_request/2
   0.18%  perf-17598.map  [.] rtb_gateway_external_service:receive_all/2
   0.15%  perf-17598.map  [.] shackle_server:handle_msg/2
   0.15%  perf-17598.map  [.] binary:split/3
   0.15%  perf-17598.map  [.] uuid:new/2
   0.15%  perf-17598.map  [.] cowboy_protocol:match_eol/2
   0.15%  perf-17598.map  [.] statsderl:increment/3
   0.14%  perf-17598.map  [.] lists:map/2
   0.14%  perf-17598.map  [.] rtb_gateway_cache:read/4
   0.14%  perf-17598.map  [.] granderl:uniform/1
   0.12%  perf-17598.map  [.] lists:reverse/1
   0.11%  perf-17598.map  [.] rtb_gateway_exchange:metric_prefix/1
   0.11%  perf-17598.map  [.] rtb_gateway_utils:now_diff_us/1
   0.11%  perf-17598.map  [.] flights_matcher:request_variables/3
   0.11%  perf-17598.map  [.] cowboy_req:response/6
   0.10%  perf-17598.map  [.] lists:usplit_1/5
   0.10%  perf-17598.map  [.] prim_inet:async_recv/3
   0.10%  perf-17598.map  [.] cowboy_req:body/2
   0.10%  perf-17598.map  [.] binary:get_opts_split/2
   0.09%  perf-17598.map  [.] shackle_server:process_replies/2
   0.09%  perf-17598.map  [.] cowboy_req:reply/4
   0.09%  perf-17598.map  [.] cowboy_protocol:parse_host/3
   0.09%  perf-17598.map  [.] cowboy_protocol:match_colon/2
   0.09%  perf-17598.map  [.] statsderl:timing/3
   0.08%  perf-17598.map  [.] rtb_gateway_open_rtb:bid_req_exchange_id/1
   0.08%  perf-17598.map  [.] shackle_server:loop/1
   0.08%  perf-17598.map  [.] uuid:uuid_to_string/2
   0.08%  perf-17598.map  [.] lists:umergel/3
   0.08%  perf-17598.map  [.] rtb_lib_codecs:varint_decode/3
   0.08%  perf-17598.map  [.] rtb_gateway_config:read_cache/3
   0.08%  perf-17598.map  [.] rtb_lib_utils:timeout_value/3
   0.08%  perf-17598.map  [.] prim_inet:send/3
   0.08%  perf-17598.map  [.] inet_parse:ipv4_field/4
   0.08%  perf-17598.map  [.] lists:usort/1
For a higher level overview, try: perf top --sort comm,dso
```

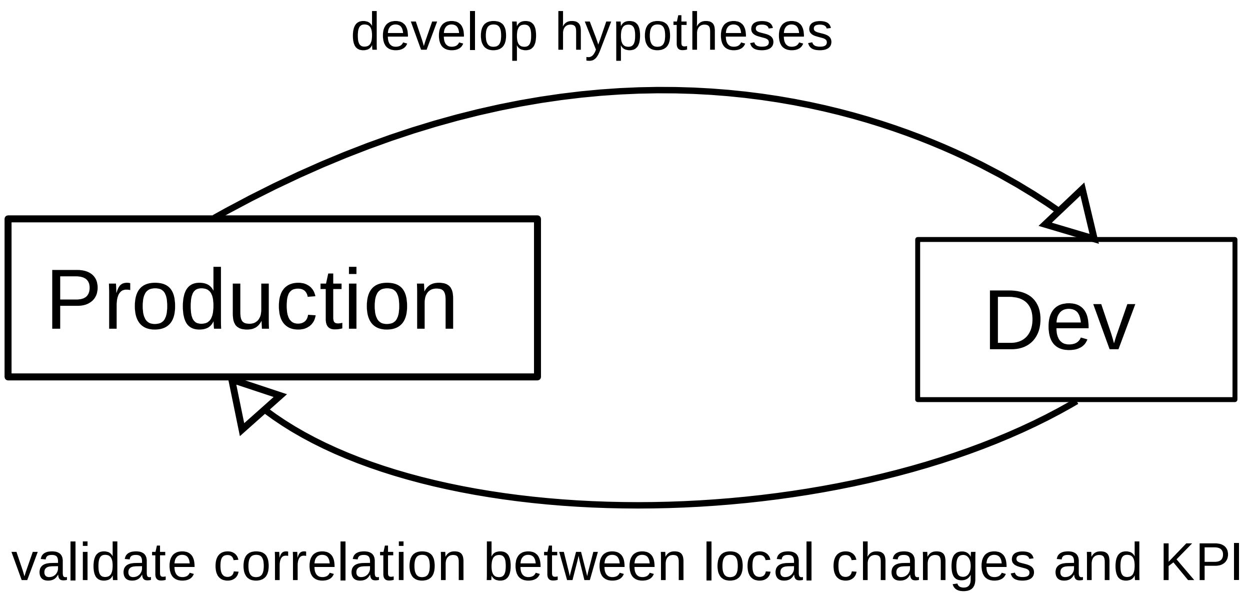
LINKING EXPERIMENTS

LINKING EXPERIMENTS: KPIS



	max	avg	curren
🟧 jiffy_encode	65.1 ms	21.3 ms	16.8 m
🟦 jiffy_decode	26.2 ms	747 µs	332 µ
🟩 vm_evaluate	21.5 ms	638 µs	21.5 m
🟨 classify_ua	8.2 ms	256 µs	66 µ
🟪 quoted_from_url	7.3 ms	1.1 ms	191 µ
🟧 delivery_info_encode	6.7 ms	84 µs	23 µ
🟦 doubleclick_encode	4.3 ms	109 µs	19 µ
🟪 geo_lookup	4.0 ms	332 µs	120 µ
geo_normalize	2.9 ms	118 µs	18 µ

LINKING EXPERIMENTS



IDEA: INTENTIONALLY SLOW SUSPECTED PATHS

See also Coz, the causal profiler

BEWARE GOODHART'S LAW

When a measure becomes a target, it ceases to be a good measure.

— Goodhart's Law

RIGOROUS / HONEST BENCHMARKING

Kalibera and Jones, "Rigorous Benchmarking in Reasonable Time", 2013.

WHO TRIGGERS GCS?

```
$ erlang-sample -d 60 --blame erts_garbage_collect 17598
1057    rtb_lib_indexer:get_entry/3
1011    bertconf:read/2
64      jiffy:nif_encode_init/2
63      rtb_gateway_exchange:request/2
44      jiffy:nif_encode_iter/3
36      cowboy_protocol:parse_hd_value/9
31      statsderl:maybe_cast/4
26      cowboy_protocol:parse_hd_name/8
25      lists:reverse/1
20      rtb_gateway_pacing:explode_pacings/2
[...]
```

- same as ordinary sampling, but only count functions seen under `erts_garbage_collect`
- also works with `copy_struct`, `erts_cmp_compound`, et cetera

ALLOCATOR STATS

- `recon` is nice, but can do a lot of work collecting allocator statistics
- ftrace `mmap(2)`, read `*_alloc_state`

```
struct Allctr_t_ {
    [...]
    int                     t;
    int                     ramv;
    Uint                    sbc_threshold;
    Uint                    sbc_move_threshold;
    Uint                    mbc_move_threshold;
    Uint                    main_carrier_size;
    Uint                    max_mseg_sbcs;
    Uint                    largest_mbc_size;
    Uint                    smallest_mbc_size;
    Uint                    mbc_growth_stages;
    [...]
    Uint                    mbc_header_size;
    Uint                    min_mbc_size;
    Uint                    min_mbc_first_free_size;
    Uint                    min_block_size;
    [...]
    CarriersStats_t         sbcs;
    CarriersStats_t         mbcs;
    carrierstats_t      mbes;

    [...]
}
```

A HACK TOO FAR

```
pid_t spy_pid;
uintptr_t spy_ptr;

static void *spy_fn(void)
{
    spy_pid = syscall(__NR_gettid);
    sched_setscheduler(spy_pid, SCHED_IDLE, &(struct sched_param){.sched_prior
    asm volatile("" ::: "memory");
    asm volatile ("forever:\n"
                  "movq %0, %%rsp\n"
                  "movl %1, %%eax\n"
                  "int $0x80\n"
                  "jmp forever\n"
                  : : "m" (spy_ptr), "r" (__NR_sched_yield) : "rsp");
    __builtin_unreachable();
}
```

THINGS TO IMPROVE

- make tools easier to use
- reduce skid, more correctness checks
- better debug info
- kcov for coverage in production
    - maybe with Processor Trace
- BPF + SystemTap = safe ustack helpers?

contribute: `github.com/tokenrove/extrospect-beam`

feedback: `julian@cipht.net`

adgear