Erlang debugging

mats cronqvist

Ruminations on tools and strategies.

With boring anecdotes!

rumination

n 1: a calm lengthy intent consideration [syn: contemplation, reflection, reflexion, musing, thoughtfulness]
2: (of ruminants) chewing (the cud); "ruminants have remarkable powers of rumination"
3: regurgitation of small amounts of food; seen in some infants after feeding

Masse goes to Denmark...

- Embedded system running OTP R5.
- Live in Denmark.
- There was no way to log into the CPU.
- There was no way to load new code.
- There was no usable application debugging tool.
- You could physically connect a terminal.

The node got overloaded after 90 days.
A tech traveled there and rebooted every 90 days.

...virtually...

- Wrote a one-liner...
- ...that ran a one-minute trace and wrote to a file.
- Sent it to the Danish tech by mail...
- ...who ran it by pasting it into a shell...
- ...before and after the reboot...
- ...and emailed the files to me (base-64 encoded)

...tracing saves the day!

Wrote a comparison profiler.
Compare the average execution time for each function, before and after the reboot.

`ets:lookup/2`
was 100 times slower before the reboot.

42

The hash function was broken for bignums.

The Dark Side

Embedded system.
Multi-million lines of C++.
The disks were too small for core files.
100s of units deployed.

...but...
The network worked.

The Point...

Debuggability is a property of a system

In a distributed system, fail-stop bugs are easy

The 3 kinds of bugs

- It crashes “randomly”
- It uses too much of a resource
- It gives the wrong answer

Strategy

	Monitoring	Narrowing
crashes	logging	context
performance	real-time logging	process → function
wrong result	contracts (test cases)	context

polling

The Erlang VM has many info methods

- `erlang:memory`
- `erlang:system_info`
- `erlang:statistics`
- `erlang:process_info`
- `inet:i().`

Narrowing, procs

UNIX top

```
top - 13:54:15 up 24 days,  2:59,  9 users,  load average: 0.15, 0.42, 0.49
Tasks: 192 total,   5 running, 185 sleeping,   0 stopped,   2 zombie
Cpu(s):  7.0%us,  3.3%sy,  0.0%ni, 89.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   3106248k total,  2978996k used,   127252k free,    67844k buffers
Swap:        0k total,        0k used,        0k free,  2066100k cached

  PID %MEM  VIRT SWAP  RES CODE DATA  SHR nFLT nDRT S  PR  NI %CPU COMMAND
 8748  4.5  181m  44m 136m 1544 174m 2156   15     0 S  20   0     1 beam.smp
 2842  3.6  307m 196m 110m   44 236m  27m    0     0 R  20   0     7 firefox-bin
29380  3.0  395m 304m  91m   48 225m  30m 1095     0 S  20   0     0 amarok
24748  2.6  171m  93m  77m  31m 113m  12m    7     0 S  20   0     0 chrome
```

Narrowing, procs

dtop

```
dtop:start()
--------------------------------------------------------------------------------
kr@sterlett              size: 4(45)M, cpu%: 0(27), procs: 39, runq: 0, 13:57:19
memory[kB]:  proc     591, atom     300, bin      40, code    1905, ets     128

pid         name                                current                msgq   mem cpu
<0.49.0>    prfTarg                             prfPrc:pidinfo/2          0    31   0
<0.40.0>    group:server/3                      group:get_line1/3         0    11   0
<0.46.0>    dtop                                prfHost:loop/1            0    21   0
<0.28.0>    user_drv:server/2                   user_drv:server_loo       0    17   0
<0.50.0>    erlang:apply/2                      file_io_server:serv       0     8   0
<0.51.0>    erlang:apply/2                      file_io_server:serv       0     8   0
```

Interrupts

The Erlang VM has 2 interrupt mechanisms

- `erlang:trace/3` (redbug)
- `erlang:system_monitor/2` (watchdog)

Narrowing, functions

UNIX strace

```
STRACE(1)                                                    STRACE(1)

NAME
       strace - trace system calls and signals

SYNOPSIS
       strace  [  -dffhiqrtttTvxx  ]  [  -acolumn ] [ -eexpr ] ...  [ -ofile ] [
       -ppid ] ...  [ -sstrsize ] [ -uusername ] [ -Evar=val ] ...   [  -Evar  ]
       ...  [ command [ arg ...  ] ]

       strace -c [ -eexpr ] ...  [ -Ooverhead ] [ -Ssortby ] [ command [ arg ...
       ] ]

DESCRIPTION
       Strace intercepts and records the system calls which are called by a process
       and the signals which are received by a process.  The name of each system
       call, its arguments and its return value are printed...
```

Narrowing, functions

redbug

```
redbug:start(Trc,Opts).

 redbug is a tool to interact with the Erlang trace facility.
 It will instruct the Erlang VM to generate so called
 'trace messages' when certain events (such as a particular
 function being called) occur.
 The trace messages are either printed (i.e. human readable)
 to a file or to the screen; or written to a trc file.
```

redbug trace patterns

```
Trc: list('send'|'receive'|{M}|{M,F}|{M,RMSs}|{M,F,RMSs})
Proc: 'all'|pid()|atom(Regname)|{'pid',I2,I3}
Targ: node()
RMSs: (restricted match specs): list(RMS)
RMS: 'stack'|'return'|tuple(ArgDescriptor)
ArgDescriptor: '_'|literal()
```

redbug opts

```
Opts: list({Opt,Val})
  general opts:
time          (15000)        stop trace after this many ms
msgs          (10)           stop trace after this many msgs
proc          (all)          (list of) Erlang process(es)
targ          (node())       node to trace on
  print-related opts
max_queue     (5000)         fail if internal queue gets this long
max_msg_size (50000)         fail if seeing a msg this big
buffered      (no)           buffer messages till end of trace
print_form    ("~s~n")       print msgs using this format
print_file    (standard_io) print to this file
print_re      ("")           print only strings that match this
  trc file related opts
file          (none)         use a trc file based on this name
file_size     (1)            size of each trc file
file_count    (8)            number of trc files
```

redbug examples

```
> redbug:start({erlang},[{msgs,1}]).
14:51:05 <kdb_sync> {erlang,'++',
                          ["/home/masse/svn/kred/devel-xmlrepc/logs",
                           "/TICK"]}

> redbug:start({erlang,'++'},[{msgs,1}]).
14:51:48 <kdb_sync> {erlang,'++',
                          ["/home/masse/svn/kred/devel-xmlrepc/logs",
                           "/TICK"]}

> redbug:start({erlang,'++',[{"",'_'}]},[{msgs,2}]).
14:52:54 <ssl_broker_sup> {erlang,'++',[[],[<0.550.0>,acceptor,[]]]}
```

more examples

```
> redbug:start({erlang,'++',[{"",'_'}]},[{msgs,20},{print_file,"/tmp/foo"}]).

> redbug:start({erlang,'++'},[{msgs,2},{proc,kdb_sync}]).
14:57:00 <kdb_sync> {erlang,'++',
                     ["/home/masse/svn/kred/devel-xmlrepc/logs",
                      "/TICK"]}
```

redbug for wizards

```
> redbug:start({erlang,'++',[stack,return]},[{msgs,2},{proc,kdb_sync}]).
14:58:36 <kdb_sync> {erlang,'++',
                             ["/home/masse/svn/kred/devel-xmlrepc/logs",
                              "/TICK"]}

  {proc_lib,init_p_do_apply,3}
  {gen_server,handle_msg,5}
  {kdb_sync,handle_info,2}

14:58:36 <kdb_sync> {erlang,'++',2} -> "/home/masse/svn/kred/devel-
xmlrepc/logs/TICK"
```