PROFILING AND DEBUGGING

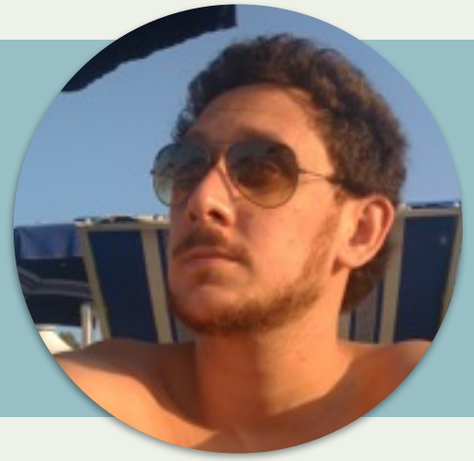
Erlang Systems

Roberto Aloi - Martin Kjellin

Erlang SOLUTIONS

Klarna

Erlang SOLUTIONS



M.Sc in Computer Engineering
Working with Erlang technologies since 2007
Senior Consultant and Trainer at Erlang Solutions
Sicilian of origin, based in Stockholm

M.Sc in Computer Science, almost
Working with Erlang Technologies since 2005
Developer at Klarna

Klarna

Contents

Profiling and Debugging?

Not enough to determine where a system spends time - why it takes time needs to be answered too

Shared toolset for both profiling and debugging

You find - you fix!

Profiling mindset

Measure, don't guess
Trust your measurements
Measure before optimising
Know what you're measuring

Measure!

What are we looking for?

OS Level

- Memory Usage
- Disk IO & IO Wait
- CPU usage
- Network utilisation

Erlang VM

- Message queue
- Reductions
- (garbage collections, stack traces, excessive bif usage ...)

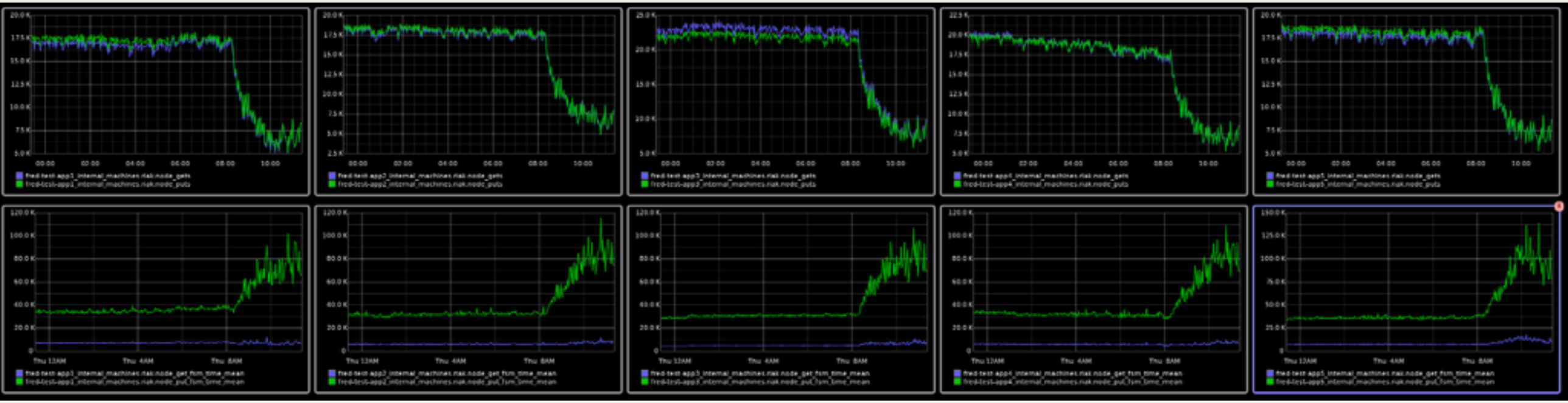
Rarely that simple... or?

How do we find it?

OTP Tools	fprof, cprof, eprof, observer, dbg, trace bifs
3rd party Erlang Tools	redbug, recon, dtop, erlgrind
OS Tools	htop, iostat, glances, iftop, sar, strace
	and many, many more

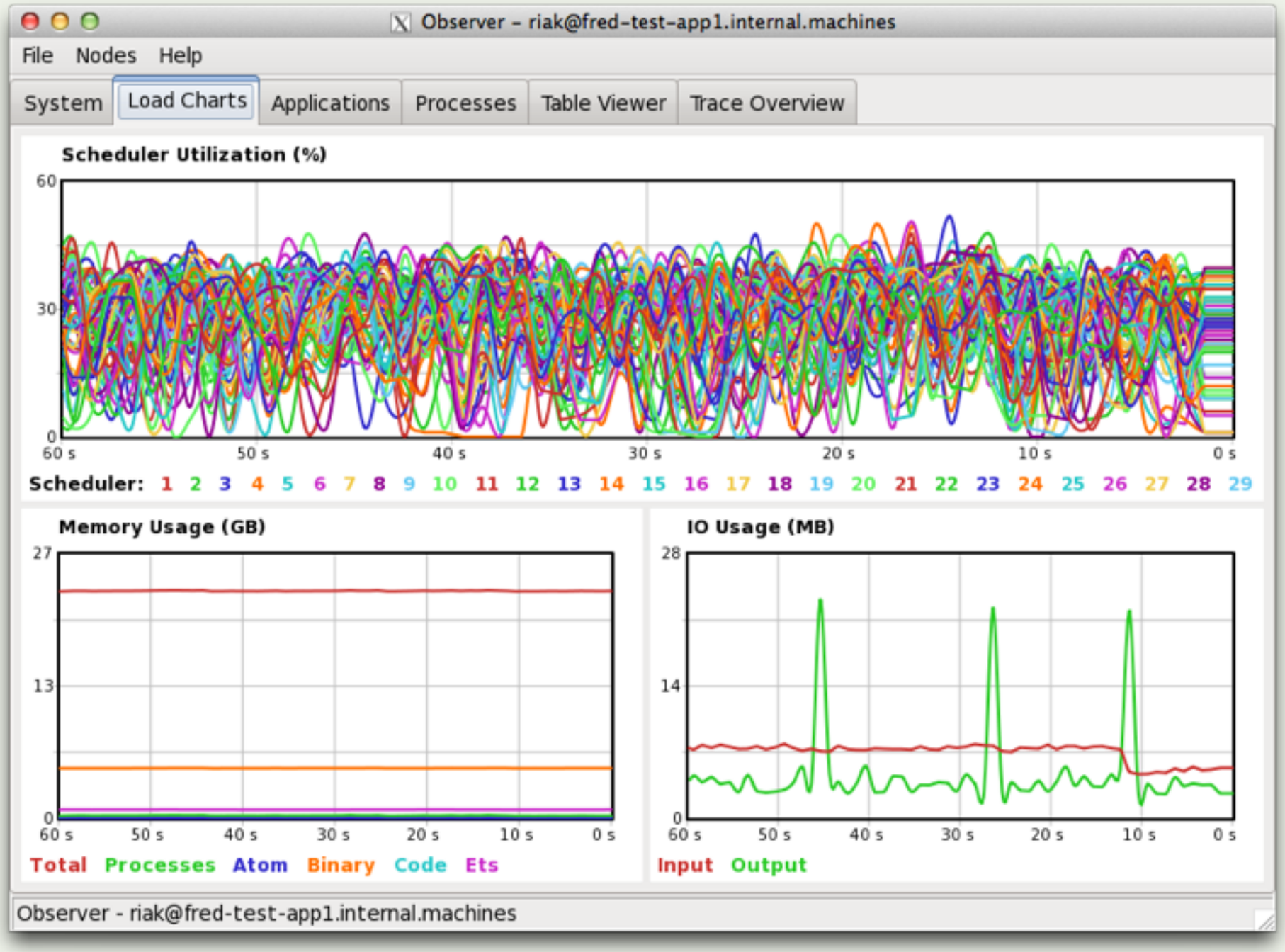
http://www.erlang.org/doc/efficiency_guide/profiling.html
https://code.google.com/p/eper/wiki/redbug

A misbehaving system



Database write times increases
Number of handled database writes decreases

Health check with Observer - load charts



Health check with Observer - Process overview

Observer - riak@fred-test-app1.internal.machines

File View Trace Nodes Help

System | Load Charts | Applications | Processes | Table Viewer | Trace Overview

Pid	Name or Initial Func	Reds	Memory	MsgQ	Current Function
<9648.3606.0>	proc_lib:init_p/5	2907832	89176	0	gen_fsm:loop/7
<9648.4019.0>	proc_lib:init_p/5	2904657	89176	0	gen_fsm:loop/7
<9648.5284.0>	proc_lib:init_p/5	2903657	89176	0	gen_fsm:loop/7
<9648.2384.0>	proc_lib:init_p/5	2903242	89176	0	gen_fsm:loop/7
<9648.2984.0>	proc_lib:init_p/5	2902583	89240	0	gen_fsm:loop/7
<9648.3053.0>	proc_lib:init_p/5	2901069	55648	0	gen_fsm:loop/7
<9648.4733.0>	proc_lib:init_p/5	2895723	89096	0	gen_fsm:loop/7
<9648.3955.0>	proc_lib:init_p/5	2889978	89176	0	gen_fsm:loop/7
<9648.5209.0>	proc_lib:init_p/5	2883386	89176	0	gen_fsm:loop/7
<9648.4595.0>	proc_lib:init_p/5	2880107	89176	0	gen_fsm:loop/7
<9648.4883.0>	proc_lib:init_p/5	2875386	55728	0	gen_fsm:loop/7
<9648.2096.0>	proc_lib:init_p/5	2871258	89176	0	gen_fsm:loop/7
<9648.1135.0>	proc_lib:init_p/5	2870136	89176	0	gen_fsm:loop/7
<9648.4475.0>	proc_lib:init_p/5	2860145	89320	0	gen_fsm:loop/7
<9648.4084.0>	proc_lib:init_p/5	2858853	89176	0	gen_fsm:loop/7
<9648.2710.0>	proc_lib:init_p/5	2856568	89176	0	gen_fsm:loop/7
<9648.5075.0>	proc_lib:init_p/5	2856019	89176	0	gen_fsm:loop/7
<9648.3257.0>	proc_lib:init_p/5	2851645	89176	0	gen_fsm:loop/7
<9648.2164.0>	proc_lib:init_p/5	2851390	89096	0	gen_fsm:loop/7
<9648.2232.0>	proc_lib:init_p/5	2851363	89176	0	gen_fsm:loop/7
<9648.5143.0>	proc_lib:init_p/5	2850250	89320	0	gen_fsm:loop/7
<9648.2301.0>	proc_lib:init_p/5	2847788	89176	0	gen_fsm:loop/7

Observer - riak@fred-test-app1.internal.machines

Inspect info from Observer closer with redbug

(user@db-node)1> redbug:start(“gen_fsm:loop").

```
23:54:04 <{db_core_vnode,init,1}> {gen_fsm,loop,
                                   [<0.192.0>,<0.5143.0>,active,
                                    {state,
                                     1370157784997721485815954530671515330927436759040,
                                     db_kv_vnode,
                                     {state,
                                      1370157784997721485815954530671515330927436759040,
                                      db_kv_multi_backend,
                                      {state,
                                       [{<<"memory_multi">>,
                                         db_kv_memory_backend,
                                         {state,103678989,103547916,
                                          undefined,undefined,0,undefined}},
```

Code inspection showed this to be expected

System health check - htop

```
1  [||||||||          26.6%]   9  [||||||          20.5%]   17 [|||||||         23.4%]   25 [|||||           17.6%]
2  [||||||||          30.4%]   10 [||||||||        28.0%]   18 [||||||||||      36.9%]   26 [||||||||        25.5%]
3  [||||||||          27.1%]   11 [|||||||         24.2%]   19 [||||||         20.4%]   27 [||||||          21.7%]
4  [||||||||          28.0%]   12 [||||||||        29.3%]   20 [||||||         23.1%]   28 [||||||||        24.5%]
5  [||||||            22.2%]   13 [||||||          17.8%]   21 [||||||||||||   48.1%]   29 [|||||           15.8%]
6  [||||||||          28.7%]   14 [|||||||||       28.5%]   22 [|||||||         25.3%]   30 [||||||||        28.5%]
7  [||||||            21.0%]   15 [||||||          22.3%]   23 [|||||           16.6%]   31 [|||||           17.6%]
8  [|||||||           26.6%]   16 [|||||||         24.8%]   24 [|||||||         26.6%]   32 [||||||||||||    48.4%]
Mem[||||||||||||||||||||||||||||||||||||||||||||||25254/129013MB]   Tasks: 59, 359 thr; 19 running
Swp[||||                                          107/2015MB]      Load average: 17.01 15.52 14.68
                                                                   Uptime: 50 days, 20:08:00

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
10505 riak       20   0 35.2G 20.2G  227M S 933. 16.0     857h /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10710 riak       20   0 35.2G 20.2G  227M S 46.4 16.0 14h45:52 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10717 riak       20   0 35.2G 20.2G  227M S 37.5 16.0 14h05:09 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10719 riak       20   0 35.2G 20.2G  227M R 22.9 16.0 14h02:33 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10711 riak       20   0 35.2G 20.2G  227M R 22.2 16.0 14h47:08 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10049 fred       20   0 2764M  228M  2624 S 21.0  0.2 10h33:01 /opt/fred/erts-5.9.3.3/bin/beam.smp -K true -A 5 -- -root /opt/fred -progna
10716 riak       20   0 35.2G 20.2G  227M R 19.7 16.0 14h02:25 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10708 riak       20   0 35.2G 20.2G  227M S 19.7 16.0 14h46:33 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10704 riak       20   0 35.2G 20.2G  227M S 19.7 16.0 14h51:14 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10730 riak       20   0 35.2G 20.2G  227M R 19.7 16.0 13h52:19 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10718 riak       20   0 35.2G 20.2G  227M R 19.1 16.0 14h01:24 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10722 riak       20   0 35.2G 20.2G  227M S 19.1 16.0 14h00:14 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10733 riak       20   0 35.2G 20.2G  227M R 19.1 16.0 13h46:55 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10720 riak       20   0 35.2G 20.2G  227M S 18.4 16.0 13h54:12 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10706 riak       20   0 35.2G 20.2G  227M R 18.4 16.0 14h45:33 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10721 riak       20   0 35.2G 20.2G  227M R 17.8 16.0 13h54:05 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10712 riak       20   0 35.2G 20.2G  227M R 17.8 16.0 14h40:01 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10707 riak       20   0 35.2G 20.2G  227M S 17.2 16.0 14h42:39 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10727 riak       20   0 35.2G 20.2G  227M R 17.2 16.0 13h49:27 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10726 riak       20   0 35.2G 20.2G  227M R 17.2 16.0 13h58:13 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10705 riak       20   0 35.2G 20.2G  227M S 17.2 16.0 14h46:44 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10709 riak       20   0 35.2G 20.2G  227M R 16.5 16.0 14h47:45 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10715 riak       20   0 35.2G 20.2G  227M S 16.5 16.0 14h09:32 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10732 riak       20   0 35.2G 20.2G  227M R 16.5 16.0 13h48:02 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10713 riak       20   0 35.2G 20.2G  227M S 15.9 16.0 14h40:18 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10728 riak       20   0 35.2G 20.2G  227M S 15.9 16.0 13h56:23 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10723 riak       20   0 35.2G 20.2G  227M R 15.2 16.0 13h52:17 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
```

OS Tools - iostat

martin.kjellin — martin.kjellin@fred-test-app1:~ — ssh — 111×18

```
Every 5.0s: iostat -x                                                   Thu Jun  5 13:25:45 2014

Linux 2.6.32-431.3.1.el6.x86_64 (fred-test-app1)       06/05/2014      _x86_64_        (32 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          10.20    0.02   47.70    0.01    0.00   42.07

Device:         rrqm/s   wrqm/s     r/s     w/s   rsec/s   wsec/s avgrq-sz avgqu-sz   await  svctm  %util
sda               0.12   160.55    0.26   99.39    17.02  2079.49    21.04     0.01    0.06   0.06   0.57
sdb               0.02   807.69   10.16   49.09  2028.57  6854.24   149.92     0.04    0.64   0.15   0.90
dm-0              0.00     0.00    0.29    0.89     9.45     7.14    14.02     0.00    0.96   0.32   0.04
dm-1              0.00     0.00    0.00    0.00     0.01     0.00     8.00     0.00    0.64   0.53   0.00
dm-2              0.00     0.00   10.17  856.78  2028.56  6854.24    10.25     0.66    0.75   0.01   0.90
dm-3              0.00     0.00    0.00    0.44     0.02     3.51     8.00     0.00    0.69   0.52   0.02
dm-4              0.00     0.00    0.07  258.44     7.34  2067.48     8.03     0.01    0.06   0.02   0.50
dm-5              0.00     0.00    0.01    0.17     0.10     1.37     8.00     0.00    0.81   0.03   0.00
```

Average values since last invocation (or system start)

On linux, use -x and keep an eye on await (i/o op latency in ms)

Use watch: watch -n 5 iostat -x

Back to htop

```
1  [||||||||          26.6%]   9  [||||||          20.5%]   17 [|||||||         23.4%]   25 [|||||           17.6%]
2  [||||||||          30.4%]   10 [||||||||        28.0%]   18 [||||||||||      36.9%]   26 [||||||||        25.5%]
3  [||||||||          27.1%]   11 [|||||||         24.2%]   19 [||||||         20.4%]   27 [||||||          21.7%]
4  [||||||||          28.0%]   12 [||||||||        29.3%]   20 [||||||         23.1%]   28 [||||||||        24.5%]
5  [||||||            22.2%]   13 [||||||          17.8%]   21 [||||||||||||   48.1%]   29 [|||||           15.8%]
6  [||||||||          28.7%]   14 [|||||||||       28.5%]   22 [|||||||        25.3%]   30 [||||||||        28.5%]
7  [||||||            21.0%]   15 [||||||          22.3%]   23 [|||||          16.6%]   31 [|||||           17.6%]
8  [|||||||           26.6%]   16 [|||||||         24.8%]   24 [|||||||        26.6%]   32 [||||||||||||    48.4%]
Mem[||||||||||||||||||||||||||||||||||||||||||||25254/129013MB]   Tasks: 59, 359 thr; 19 running
Swp[||||                                        107/2015MB]      Load average: 17.01 15.52 14.68
                                                                 Uptime: 50 days, 20:08:00

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
10505 riak       20   0 35.2G 20.2G  227M S 933. 16.0     857h /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10710 riak       20   0 35.2G 20.2G  227M S 46.4 16.0 14h45:52 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10717 riak       20   0 35.2G 20.2G  227M S 37.5 16.0 14h05:09 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10719 riak       20   0 35.2G 20.2G  227M R 22.9 16.0 14h02:33 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10711 riak       20   0 35.2G 20.2G  227M R 22.2 16.0 14h47:08 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10049 fred       20   0 2764M  228M  2624 S 21.0  0.2 10h33:01 /opt/fred/erts-5.9.3.3/bin/beam.smp -K true -A 5 -- -root /opt/fred -progna
10716 riak       20   0 35.2G 20.2G  227M R 19.7 16.0 14h02:25 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10708 riak       20   0 35.2G 20.2G  227M S 19.7 16.0 14h46:33 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10704 riak       20   0 35.2G 20.2G  227M S 19.7 16.0 14h51:14 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10730 riak       20   0 35.2G 20.2G  227M R 19.7 16.0 13h52:19 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10718 riak       20   0 35.2G 20.2G  227M R 19.1 16.0 14h01:24 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10722 riak       20   0 35.2G 20.2G  227M S 19.1 16.0 14h00:14 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10733 riak       20   0 35.2G 20.2G  227M R 19.1 16.0 13h46:55 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10720 riak       20   0 35.2G 20.2G  227M S 18.4 16.0 13h54:12 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10706 riak       20   0 35.2G 20.2G  227M R 18.4 16.0 14h45:33 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10721 riak       20   0 35.2G 20.2G  227M R 17.8 16.0 13h54:05 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10712 riak       20   0 35.2G 20.2G  227M R 17.8 16.0 14h40:01 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10707 riak       20   0 35.2G 20.2G  227M S 17.2 16.0 14h42:39 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10727 riak       20   0 35.2G 20.2G  227M R 17.2 16.0 13h49:27 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10726 riak       20   0 35.2G 20.2G  227M R 17.2 16.0 13h58:13 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10705 riak       20   0 35.2G 20.2G  227M S 17.2 16.0 14h46:44 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10709 riak       20   0 35.2G 20.2G  227M R 16.5 16.0 14h47:45 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10715 riak       20   0 35.2G 20.2G  227M S 16.5 16.0 14h09:32 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10732 riak       20   0 35.2G 20.2G  227M R 16.5 16.0 13h48:02 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10713 riak       20   0 35.2G 20.2G  227M S 15.9 16.0 14h40:18 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10728 riak       20   0 35.2G 20.2G  227M S 15.9 16.0 13h56:23 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
10723 riak       20   0 35.2G 20.2G  227M R 15.2 16.0 13h52:17 /usr/lib64/riak/erts-5.9.1/bin/beam.smp -A 64 -K true -P 256000 -S 64:64 -W
```

Swap in use but large amount of free memory?

On redhat-like systems, /proc/buddyinfo holds memory fragmentation info:

```
Node 0, zone      DMA      2      1      1      1      1      0      1      0      1      1      3
Node 0, zone    DMA32    397    281    227    485    417    263    144     52     83     11      6
Node 0, zone   Normal   1043    830    485    194     69    375    265     72     18     35     21
Node 1, zone   Normal   2981    435    411    251      0     44      0     20      0      6      1
```

Use watch to monitor:

```
watch -n5 cat /proc/buddyinfo
```

Dropping page cache restored write performance to initial numbers.

Not all bottlenecks directly visible!

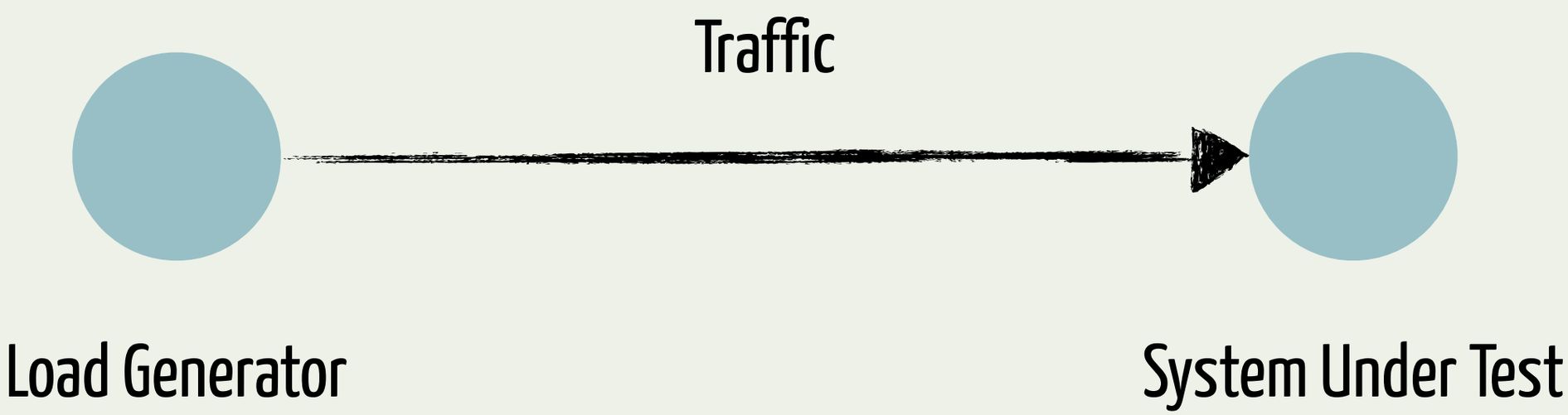
A few last words

Trace BIFs	Powerful but complex, use redbug when debugging under load.
What to profile?	Profile the entire system including disk, network & third party software
Which tools?	It depends - but use observer
Measure?	Yes! and visualise your measurements

Debug Read other people's code Cry Drink Coffee Laugh Profile Learn from mistakes
Observer Redbug Is that a timer:sleep? eprof prof cprof flags dtop percept Scream
Priority Ask that guy with the beard Bugs everywhere! Nobody Knows Why Works
on my machine Pain Optimize I should have studied law instead Strace Reductions

WAR STORY

Cry Throughput My eyes hurt Everyone else is at the party 95th Percentile load
Tracing htop percept2 Graphite UI sucks Stress Test microseconds kcachegrind
Queues Expectations Metrics Latency WTF Measure It's dark outside erlgrind
kcachegrind suspend Garbage Collection scheduler make_ref I cannot believe it



WHAT BREAKS FIRST?

lager_stress@127.0.0.1

System | Load Charts | Applications | Processes | Table Viewer | Trace Overview

Pid	Name or Initial Func	Reds	Memory	MsgQ	Current Function
<0.148.0>	lager_event	69116427	3722407424	46139	gen_event:server_notify/4
<0.3.0>	erl_prim_loader	0	263600	0	erl_prim_loader:loop/3
<0.5.0>	error_logger	0	24840	0	gen_event:fetch_msg/5
<0.6.0>	application_controller	0	17032	0	gen_server:loop/6
<0.8.0>	proc_lib:init_p/5	0	7032	0	application_master:main_loop/2
<0.9.0>	application_master:start_i...	0	2760	0	application_master:loop_it/4
<0.10.0>	kernel_sup	0	1802000	0	gen_server:loop/6
<0.11.0>	rex	0	2824	0	gen_server:loop/6
<0.12.0>	global_name_server	0	2904	0	gen_server:loop/6
<0.13.0>	erlang:apply/2	0	2720	0	global:loop_the_locker/1
<0.14.0>	erlang:apply/2	0	2720	0	global:loop_the_registrar/0
<0.15.0>	inet_db	0	16752	0	gen_server:loop/6
<0.17.0>	net_sup	0	13856	0	gen_server:loop/6
<0.18.0>	erl_epmd	0	2864	0	gen_server:loop/6
<0.19.0>	auth	0	2824	0	gen_server:loop/6
<0.20.0>	net_kernel	7	9016	0	gen_server:loop/6
<0.21.0>	inet_tcp_dist:accept_loop/2	0	2792	0	prim_inet:accept0/2
<0.22.0>	net_kernel:ticker/2	2	2720	0	net_kernel:ticker_loop/2
<0.23.0>	global_group	0	2824	0	gen_server:loop/6
<0.24.0>	file_server_2	3525	26632	0	gen_server:loop/6
<0.25.0>	code_server	0	5686136	0	code_server:loop/1
<0.26.0>	standard_error_sup	0	2864	0	gen_server:loop/6
<0.27.0>	standard_error	0	2864	0	standard_error:server_loop/1

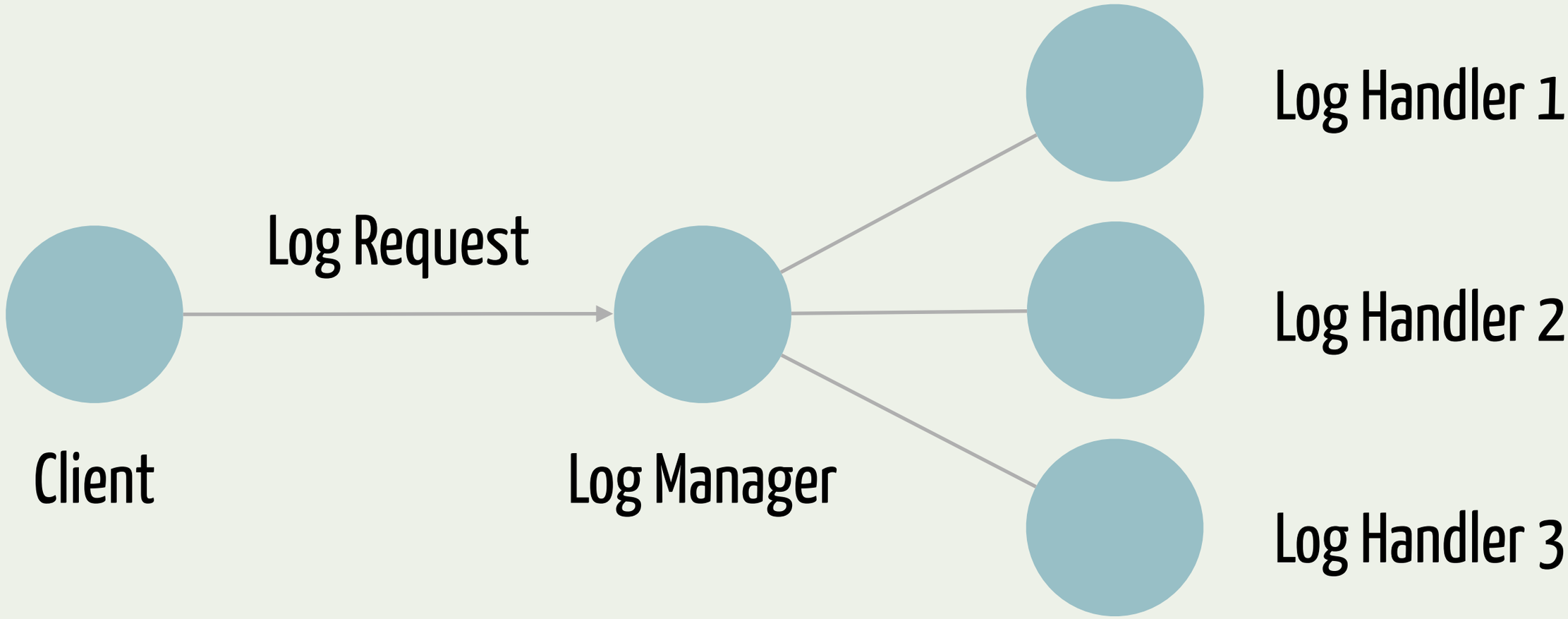
lager_stress@127.0.0.1

https://github.com/basho/lager

"Lager (as in the beer) is a logging framework for Erlang. Its purpose is to provide a more traditional way to perform logging in an Erlang application that plays nicely with traditional UNIX logging tools like logrotate and syslog."

- DISCLAIMER -

The problems we found were not in lager itself, but in the way lager was used



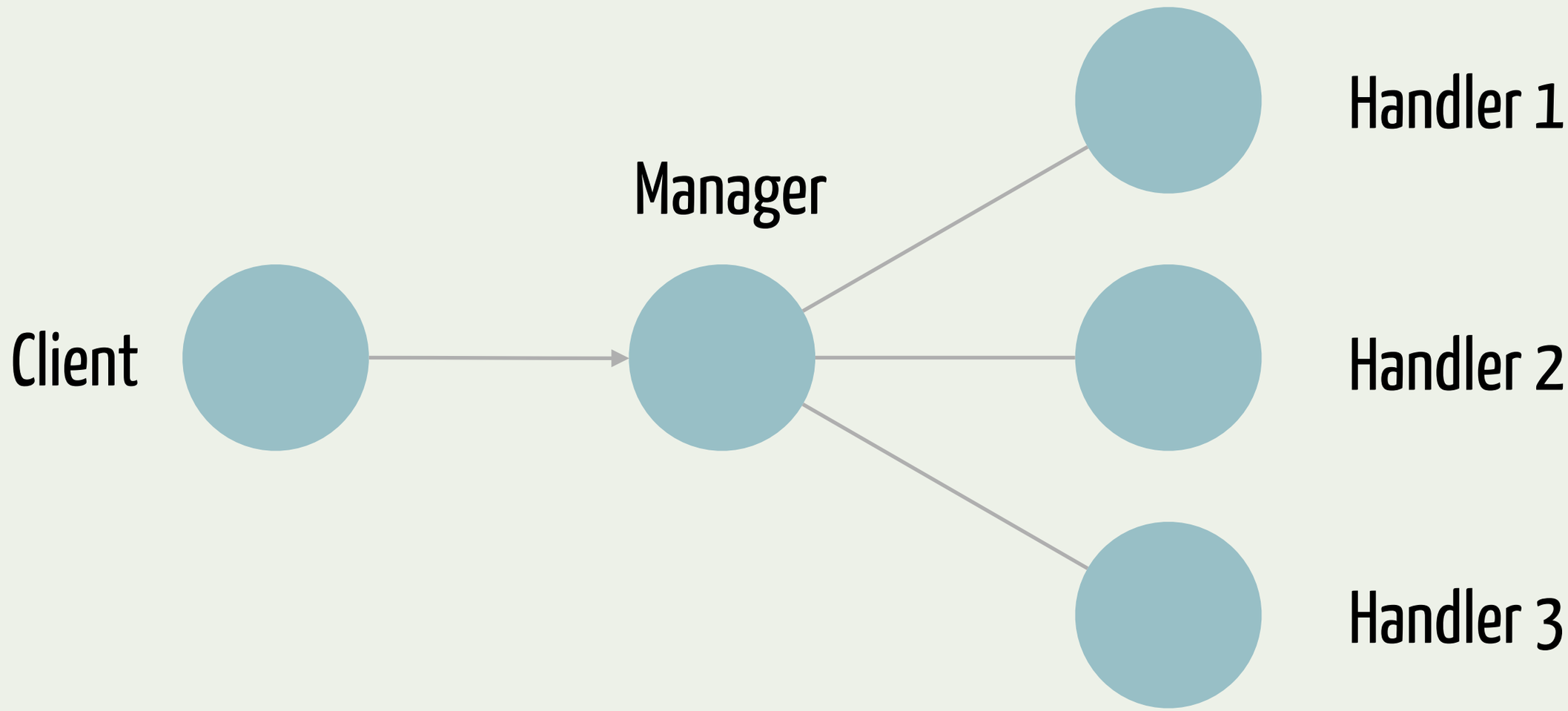

```
{lager, [
  {handlers, [
    {lager_console_backend, info},
    {lager_file_backend, [{file, “error.log"},   {level, error}]},
    {lager_file_backend, [{file, "console.log"}, {level, info}]}
  ]}
]}.
```

LAGER IS BASED ON GEN_EVENT

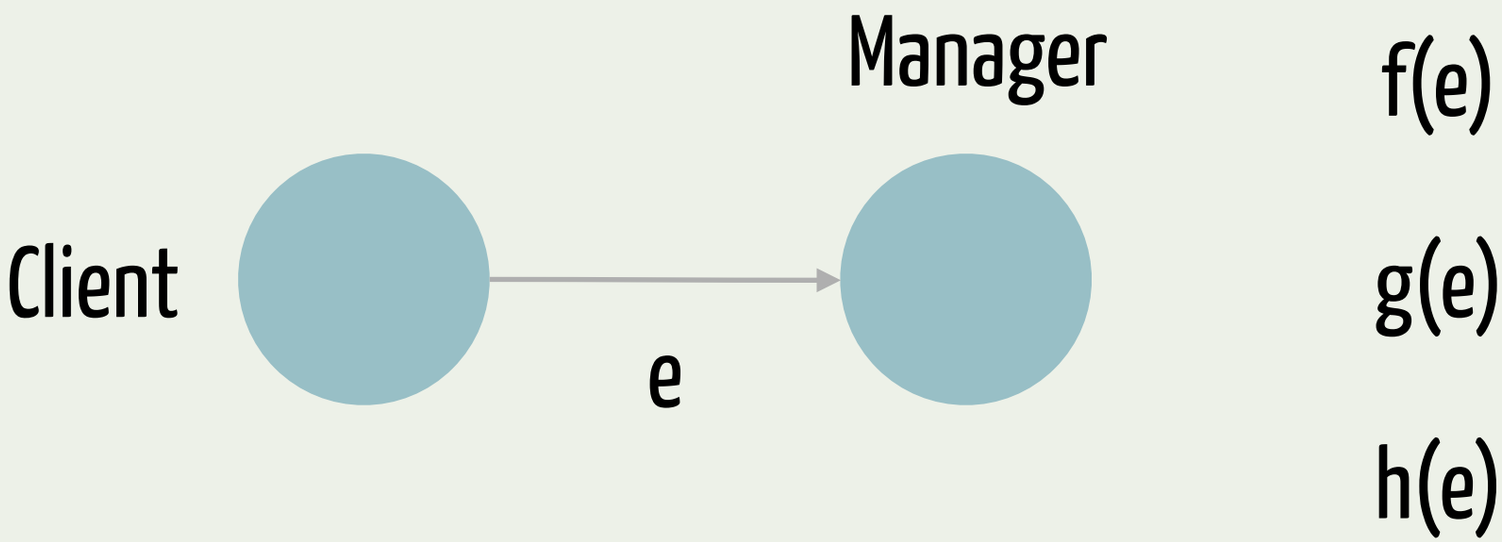
Did you know?

In the Erlang gen_event behaviour event manager and event handlers run within the same context

gen_event not



gen_event yes

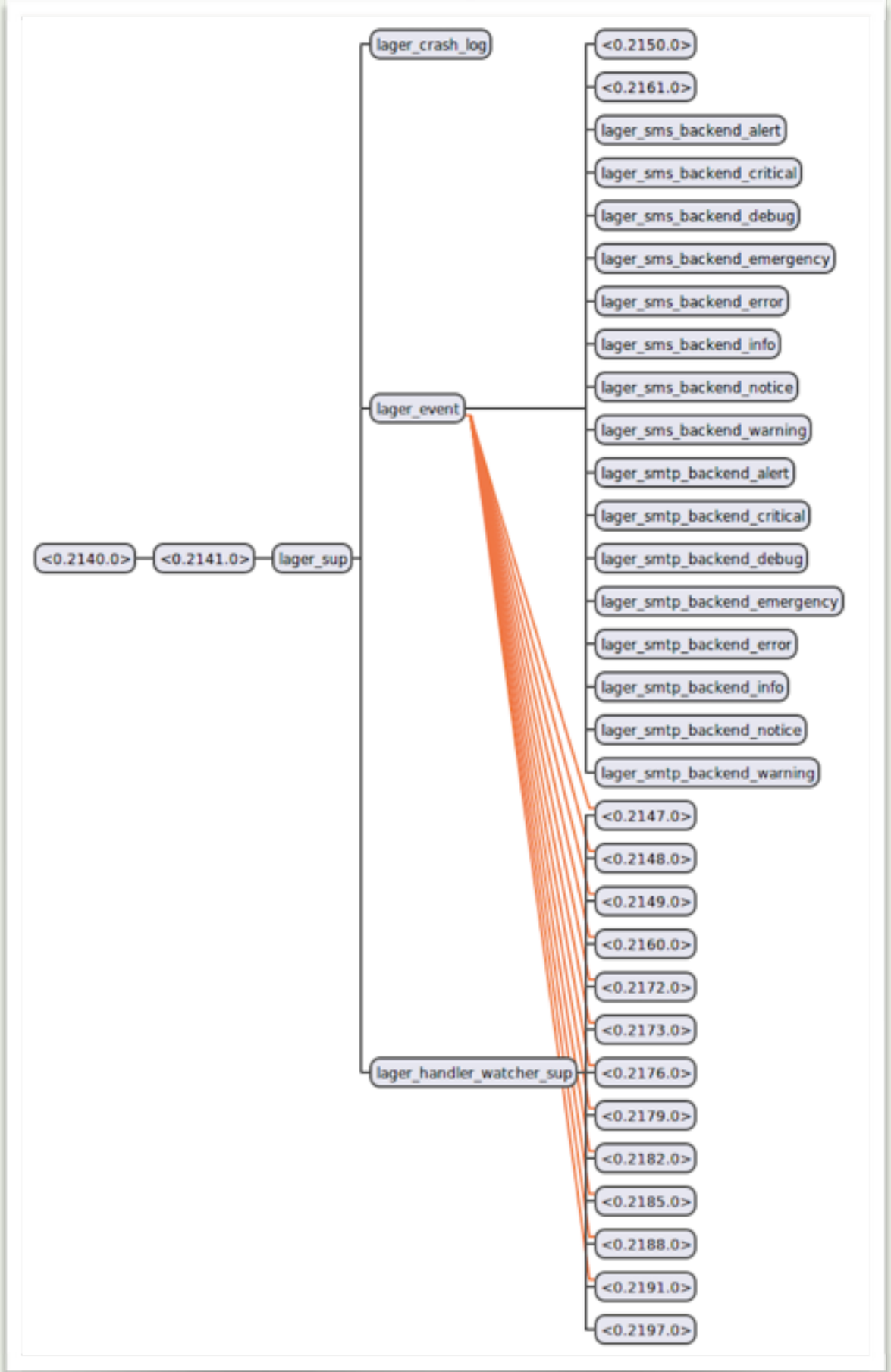


Lesson #1

USE gen_event cautiously

- Avoid synchronous calls in the handlers
- Use the manager as a “dispatcher”
- Spawn new processes whenever meaningful
- Avoid too many handlers

OR YOU MIGHT END UP WITH...



OK, THERE IS A QUEUE
IN THE EVENT MANAGER

WHY IS A BIG MESSAGE QUEUE BAD?

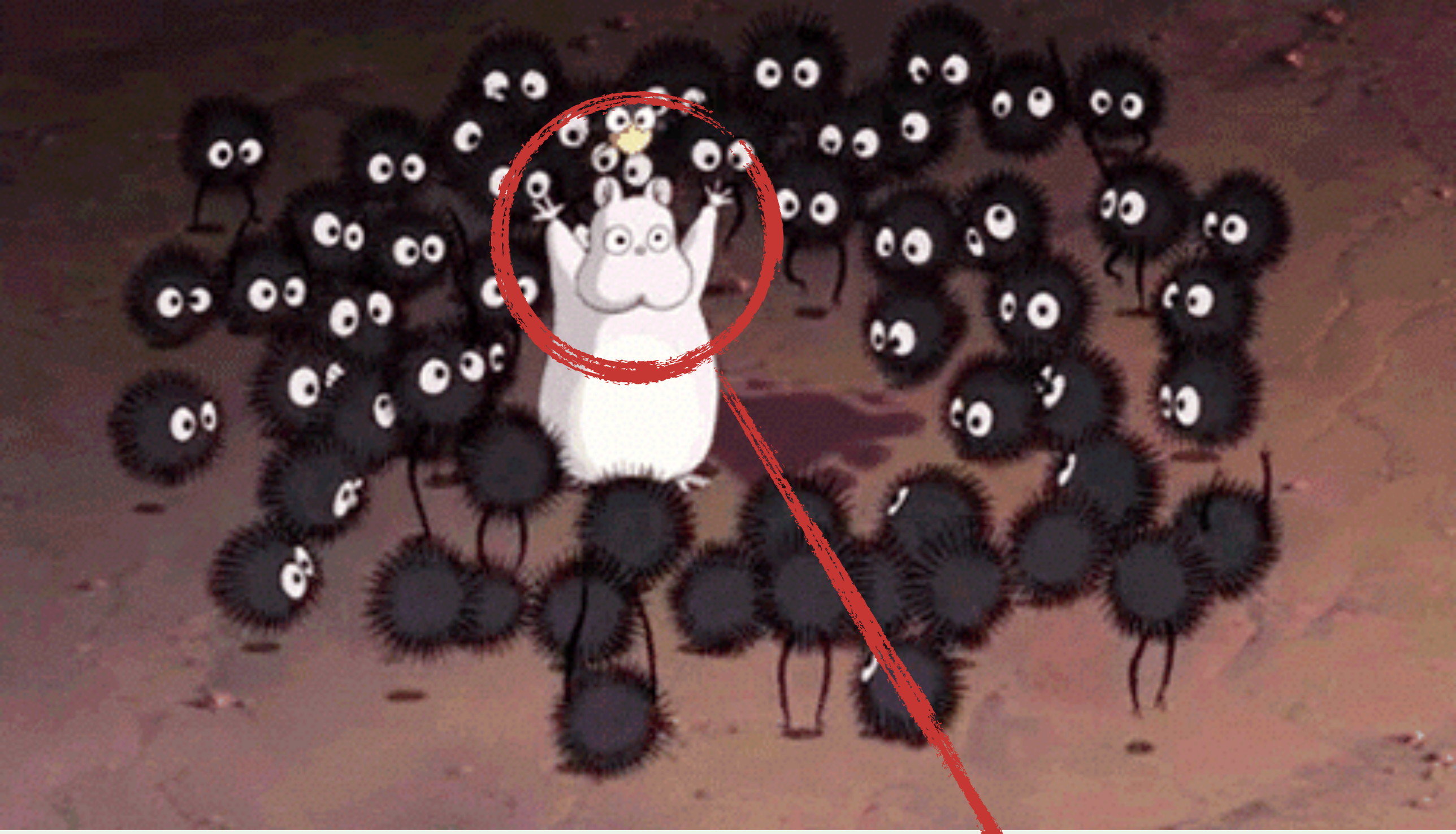
(Aside from the obvious reasons, such as memory consumption and having an overloaded process)

Did you know?

In Erlang
if you send a message
to a process which has a big mailbox
you get punished

```
if (erts_use_sender_punish)
  res *= 4;
else
  res = 0;
```

https://github.com/erlang/otp/blob/master/erts/emulator/beam/bif.c



lager_event

Lager Overload Protection

Lager 1.x

```
log(Event) ->
  gen_event:sync_notify(lager_event, Event).
```

Lager Overload Protection

Lager 2.x

```
log(Event) ->
  case lager_config:get(async, false) of
      true  -> gen_event:notify(lager_event, Event);
      false -> gen_event:sync_notify(lager_event, Event)
end;
```

async flag automatically toggled based on the mailbox size
async messaging used until the message exceeds **async_threshold**
sync messaging used after the the threshold is passed
async messaging reverted when size is back below **async_threshold** - **async_threshold_window**

“The problem with the current behaviour is it just pushes the problem onto the rest of the app. Sure, you don't have queuing in lager's mailbox, but now the rest of the app is slowed down instead. This isn't the right tradeoff for me at all. Logging should have minimal impact on the performance of the rest of the system, but instead we're effectively blocking at every log statement.”

“So, assuming you're logging at a rate faster than you can actually write those logs to disk/syslog/whatever, your choice is either: slow down the rest of the app to compensate or let the mailbox balloon and have logging slow to a crawl anyway and possibly OOM the node.”

https://github.com/basho/lager/pull/113

SO WE DID SOME PROFILING.

An Interesting Behaviour

- After stopping the load generator lot of activity is visible in htop
- Activity is restricted to one single core
- dtop shows that lager_event is busy
- no i/o wait is visible in the system

- Most of the time is spent in file:write/2 -

```
redbug:start("file:write->return",
             [{print_msec, true}, {arity, true}]).
```

A single file write operation takes 12 ms
And we have 400.000 messages to log
That's more than 1 hour to catch up

Lesson #2

THE COST OF

FILE:WRITE/2

is directly proportional to the length of the message queue of the writing process

(at least in R15B03-1)

SELECTIVE RECEIVE

```
execute_request(Pid, Request) ->
  Mref = erlang:monitor(process, Pid),
  Pid ! {io request, self(), Request},
  receive
    {io_reply, From, Reply} ->
      ...
```

io.erl

A WORKAROUND (the make_ref trick)

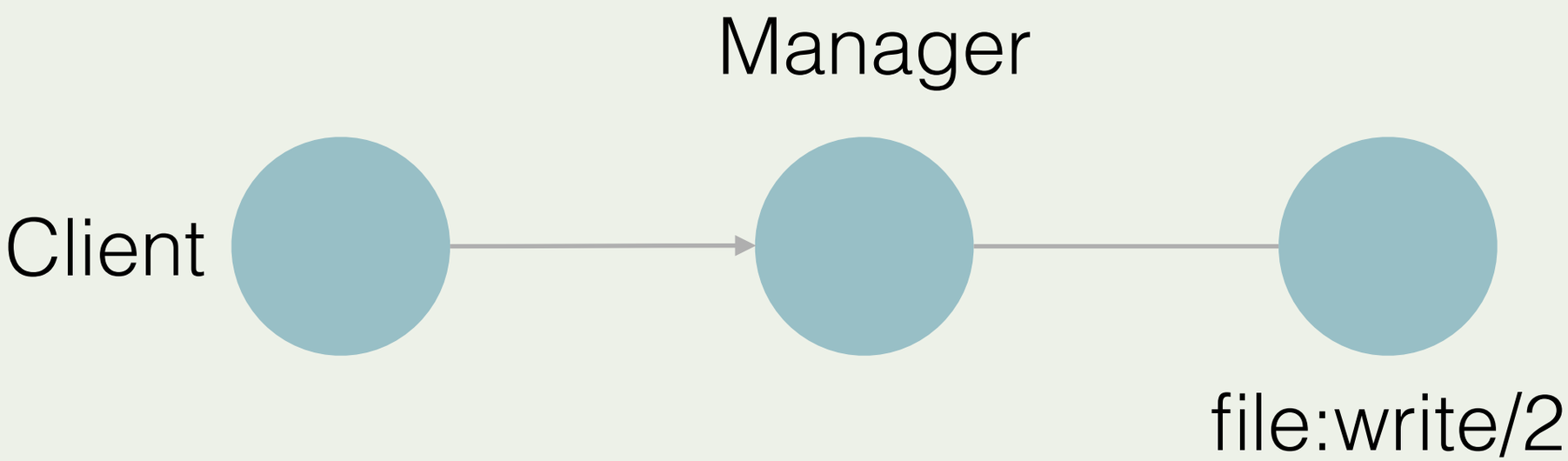
OTP-8623 == compiler erts hipe stdlib ==

Receive statements that can only read out a newly created reference are now specially optimized so that it will execute in constant time regardless of the number of messages in the receive queue for the process.

See gen:do_call/4 for an example of a receive statement that will be optimized.

Erlang R14A (June 2010) Release Notes

THE ALTERNATIVE: A MIDDLEMAN PROCESS



- We went from 12 ms to <1 ms per write -

https://github.com/klarna/lager_middleman_backend

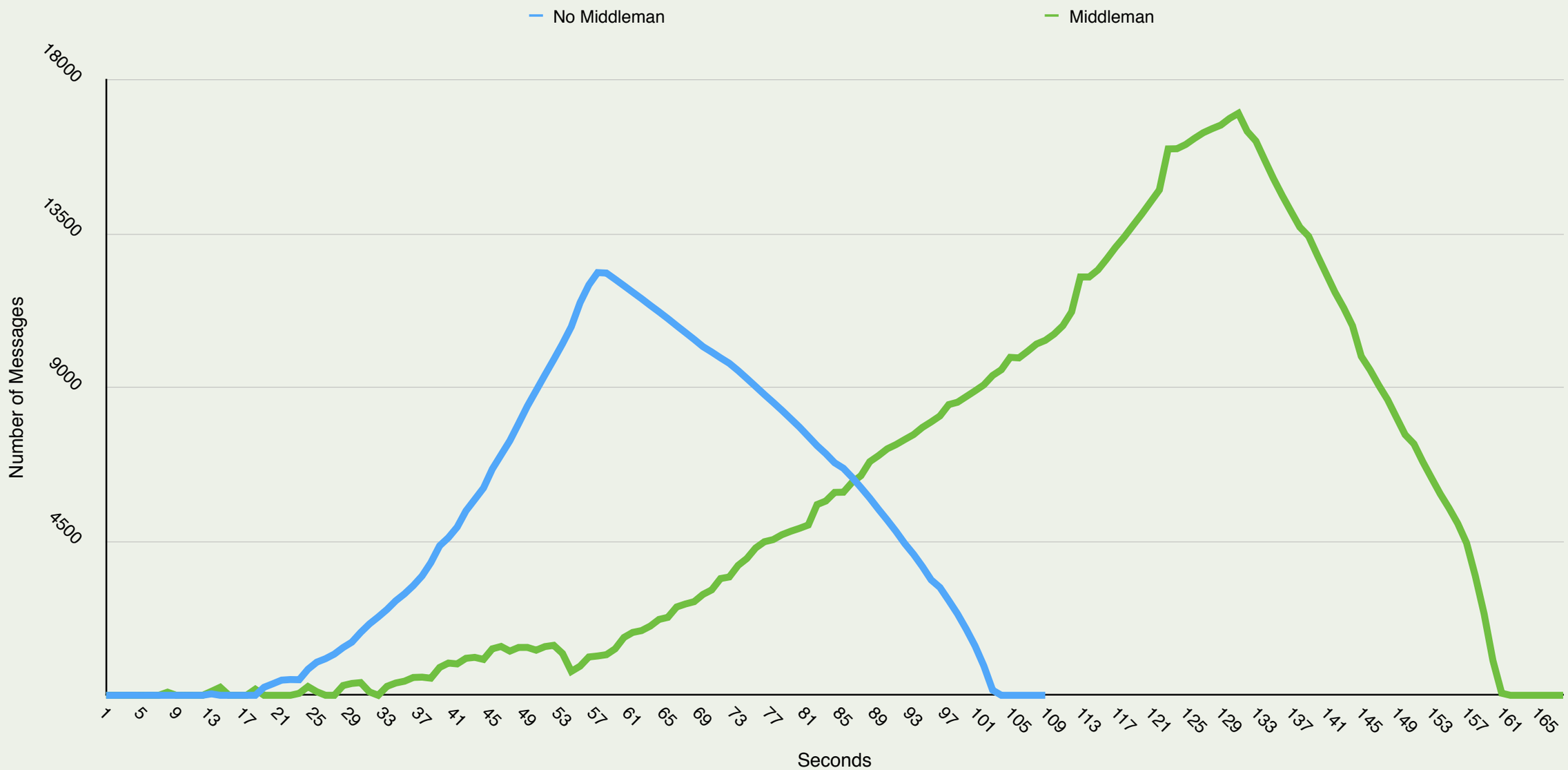
PROVE IT WORKS!

STRESS TEST SCENARIO

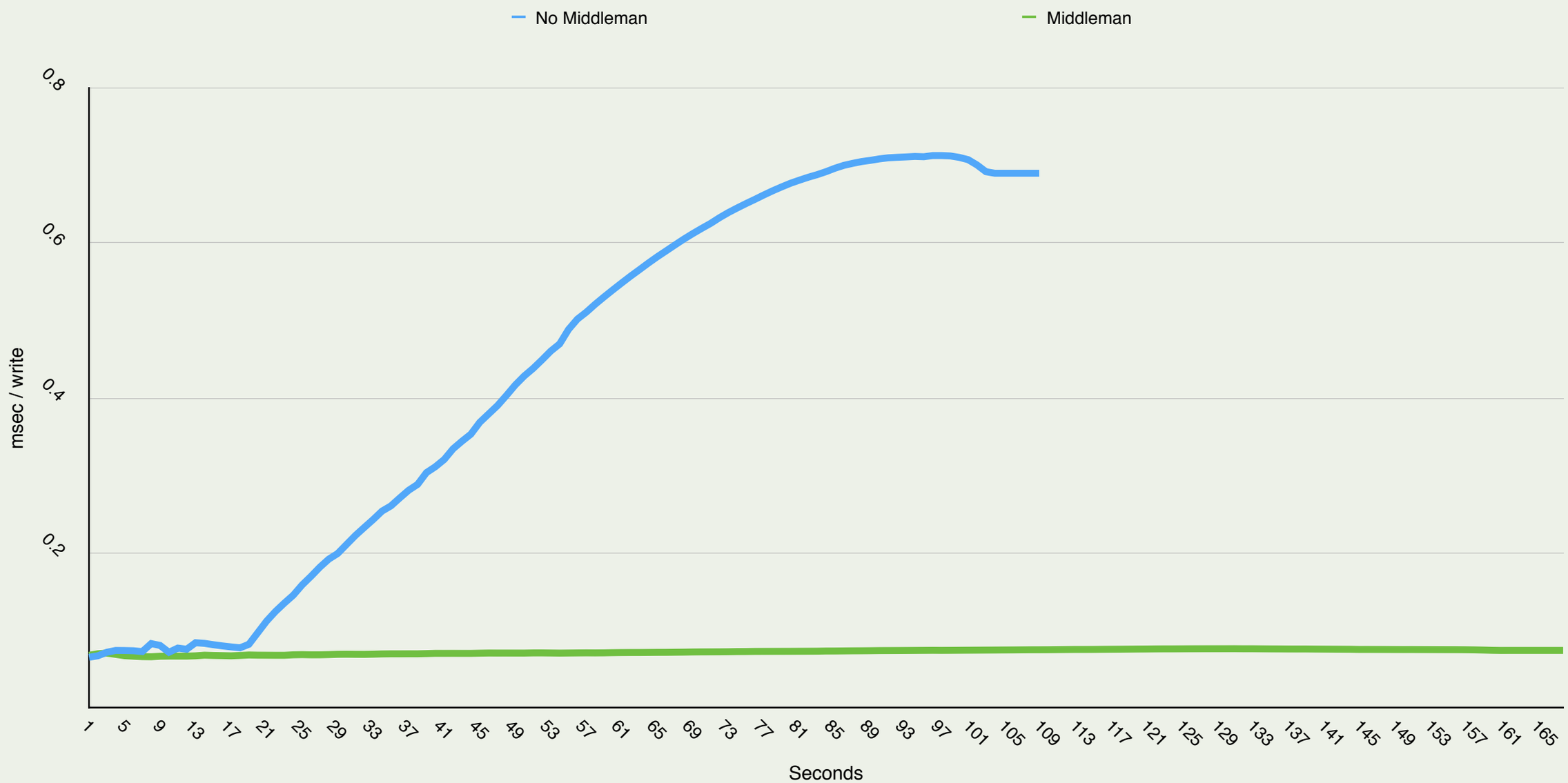
3 File handlers (debug, warning, error)
6 Concurrent workers
100 Log messages / sec
1Kb Message size
10 Minutes (interrupted)

- 1800 file write operation / second -

MESSAGE QUEUE GROWTH



FILE WRITE TIME



Questions?

Klarna

Erlang SOLUTIONS