Lager: logging with confidence

March 22nd 2013

Andrew Thompson
andrew@basho.com
github.com/Vagabond

But logging is boring, right?

- That doesn’t mean its not important
- When you do need it, you want it to work
- Logs can be customer facing
- Needs good performance and behaviour

A look at error_logger

- 3(?) log levels;info, warning, error.
- Warnings default to errors (see +W)
- gen_event based, async notify()
- Size rotation by log_mf_h.erl

Some problems

- No backpressure, sender doesn’t wait for message to be acked
- No runtime configuration at all
- Format string and arguments are passed to the gen_event and then formatted once per handler – expensive

Error logger file rotation

Consider this directory listing:

```
-rw-r--r-- 1 andrew users 1048500 Mar 15 17:19 1
-rw-r--r-- 1 andrew users 1048500 Mar 15 17:19 2
-rw-r--r-- 1 andrew users 1048500 Mar 15 17:20 3
-rw-r--r-- 1 andrew users 1048500 Mar 15 17:20 4
-rw-r--r-- 1 andrew users  565100 Mar 15 17:20 5
-rw-r--r-- 1 andrew users       1 Mar 15 17:20 index
```

How do you find the latest file?

Cat the index file: ^E

You have to look at the filesize/date or read the index file as decimal, not ASCII.

Hugely confusing for customers.

More rotation fun

- Disk is filling up, rotation size is large (like 10mb)
- “Just delete the logfile”
- No check for deletion/move of logfile, will keep writing to deleted file
- Erlang doesn’t support kill -HUP

But wait, there's more

- OTP errors include the total process state, as well as the last message received
- If you're transcoding a 10mb video in a process' state, error_logger will print the entire thing for you
- But, it'll probably OOM first

Result

- error_logger, through no fault of your own, will OOM your node in production
- Google and erlang-questions are littered with horror stories
- This is unacceptable for most deployments

We can do better

Riak_err

- Written by Basho hacker Scott Lystig Fritchie in 2010
- Replaces error_logger handlers with a custom one that truncates large input

However...

- Uses erts_debug:flat_size to detect large terms, doesn’t work for off-heap binaries
- The formatting it does for oversize messages is hard to read
- Still no solution for large mailboxes

Meanwhile, in 2009

- I was writing callcenters in upstate NY (Ruby and then Erlang)
- Wrote logging libraries for both
- Erlang logger tried to handle OOM similarly to riak_err

Cpxlog

- Async gen_event
- error_logger handler that uses trunc_io to limit size
- Added a crazy regexp based format string tokenizer
- File backend supported multiple files with different log levels

Cpxlog continued

- Levels could be changed at runtime
- Also had a syslog backend, which I wrote a port driver for
- Could enable debug messages per-module

But...

- Was under an obnoxious license (CPAL, ugh!)
- Changed jobs and went to write jQuery (javascript, ugh!!)
- Got hired at Basho. Support complained about logs, a lot. Showed them cpxlog, they told me to write them a logger for Riak.

And so, Lager was born

Builds on the experience of cpxlog and riak_err, but also learns from their mistakes

Philosophies

- Logging should NEVER crash the node
- Logs should be customer readable
- Rotation should be easy
- Log levels should be granular and flexible

Why reinvent the wheel?

- Hard to patch the ‘philosophy’ of a library
- Nothing else seemed to share my ideals of logging
- Guaranteed path to internet fame and fortune

Lager features

- Forked trunc_io and io_lib (EQC!)
- Rewrites common OTP errors into ‘english’
- Uses the 7 syslog levels, configurable at runtime
- Parse-transform that captures callsite metadata

Bonus features

- Ability to ‘route’ log messages
- Internal size and date rotation
- Supports external rotation
- File, console & syslog backends
- Lots of community provided backends; SMTP, AMQP, Loggly, CouchDB(!), MongoDB(!!), etc.

Example of ‘english’ messages

From:

```
=ERROR REPORT==== 17-Mar-2013::22:47:18 ===
** Generic server crash terminating
** Last message in was badmatch
** When Server state == {}
** Reason for termination ==
** {{badmatch,{}},
    [{crash,handle_call,3,[{file,"test/crash.erl"},{line,55}]},
     {gen_server,handle_msg,5,[{file,"gen_server.erl"},{line,588}]},
     {proc_lib,init_p_do_apply,3,[{file,"proc_lib.erl"},{line,227}]}]}
```

To:

```
22:47:38.934 [error] gen_server crash terminated with reason: no match of right
hand value {} in crash:handle_call/3 line 55
```

'Full' message is also written to crash.log

Tracing example

Send all debug messages from 'flux_capacitor' module to the console, regardless of the minimum loglevel:

```
lager:trace_console([{module, flux_capacitor}], debug).
```

Or, trace to a brand new file:

```
lager:trace_file("/tmp/flux_capacitor_debug", [{module, flux_capacitor}], debug).
```

You can also trace to an existing lager file backend and trace on any custom metadata you annotate your messages with.

A new challenger appears

- error_logger, obviously
- log4erl, inspired by log4j
- alogger, from the first spawnfest
- elog, from Inaka
- elogger, by BlueTail(via jungerl)
- fast_log, by Opscode

log4erl

- Sync gen_event, error_logger handler
- Uses log4j appender/formatter design
- Console, file, smtp, xml, syslog backends, and more
- 5 log levels, change on the fly via recompilation

Alogger

- Async gen_server with handler modules, error_logger handler
- File, console, syslog and scribe
- Handler modules can be sync or async (eg. file)
- ‘flows’, inspired lager ‘traces’
- Recompiles to change config

fast_log

- Async gen_event, no error_logger
- Only backend is sync disk_log
- Has overload protection, won’t send to gen_event if mailbox full
- Not documented, but code is nice and readable

elog

- Registered process per-loglevel
- Async logging
- Backends are per-loglevel(?) also with a default one(?)
- Had a hard time configuring it
- Uses riak_err, but not for truncating output(?)

elogger

- disk_log backend for error_logger
- Uses undocumented disk_log APIs
- disk_log:do_sync() per message
- Not a real OTP app, reads config from current application (infuriating!)

Vagabond/logbench on GitHub

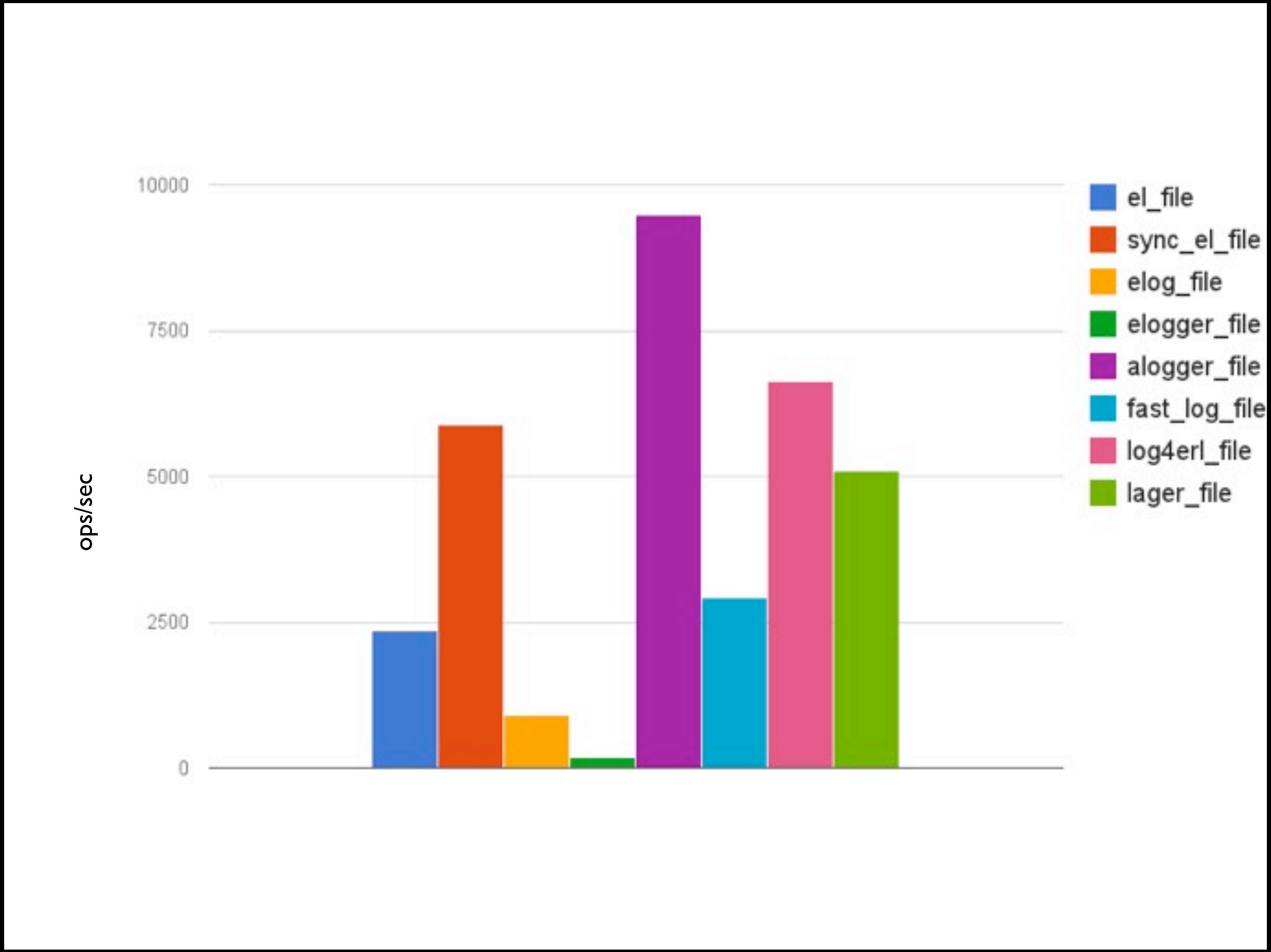
- Logging benchmark framework
- Supports all mentioned loggers, plus a fork of error_logger that runs in sync mode
- Can generate different sizes of messages
- Number of workers is configurable

Methodology

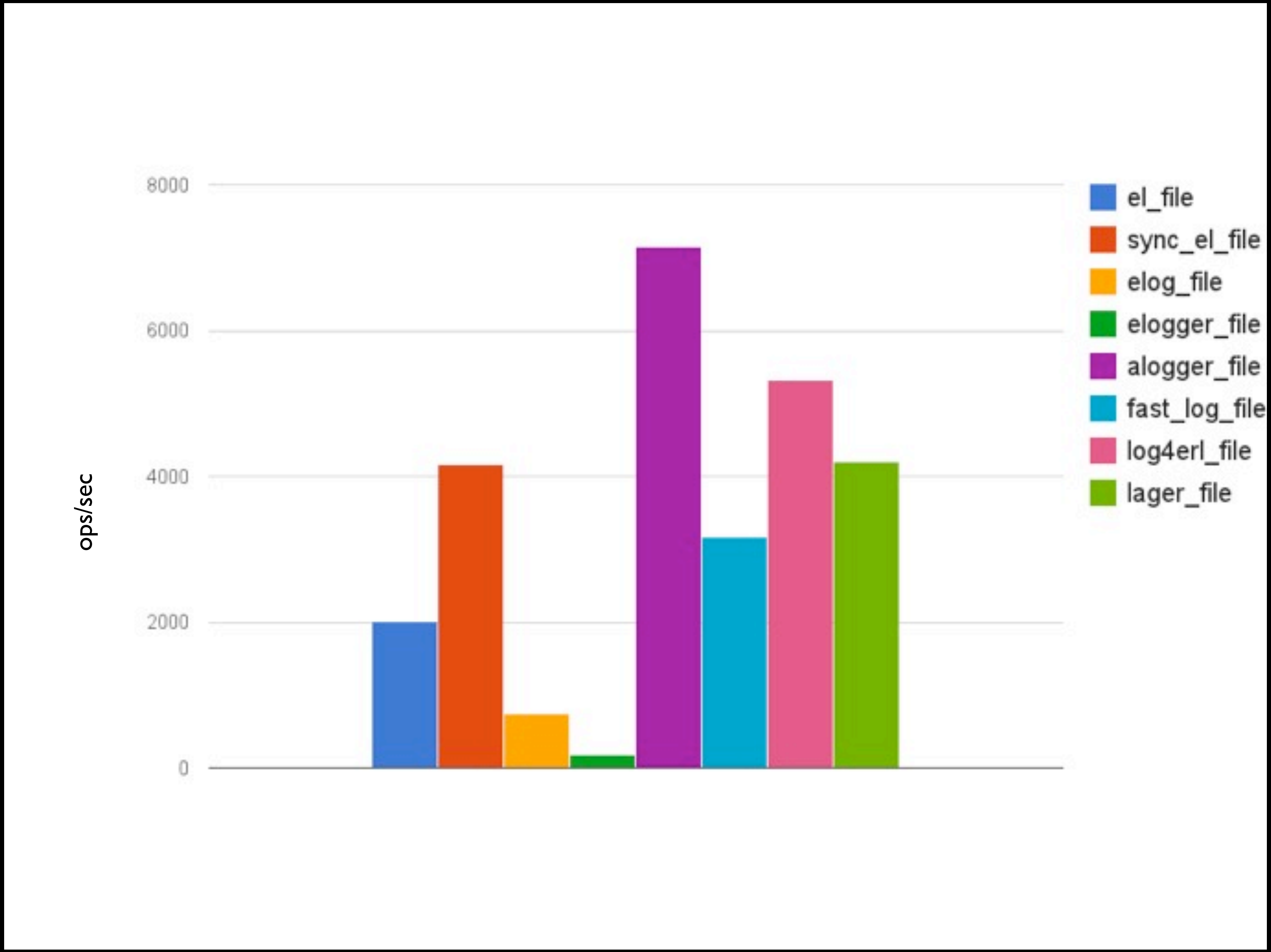
- 3 phases of benchmark, setup, run and finish
- Finish waits for all log messages to actually be processed
- Results are in operations/second, including finishing time
- 3 runs, median value selected

Other notes

- I disabled the large mailbox check in fast_log, so it'd actually log the same amount of messages as the others
- alogger uses async all the way down, to ensure that the logfiles are actually written, finish step calls disk_log:sync()



10,000 simple, 1 worker



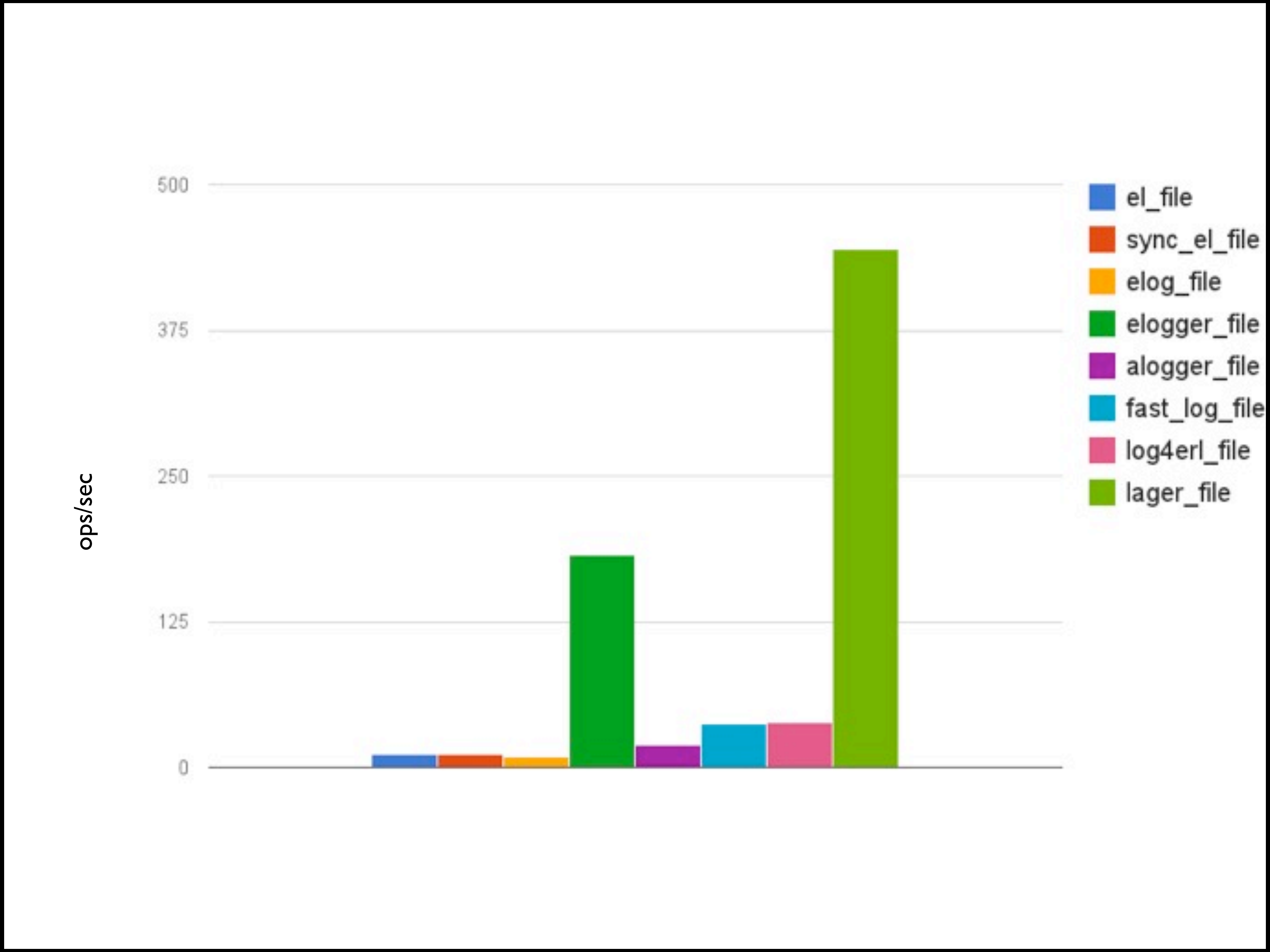
10,000 small, 1 worker

Conclusions

- Alogger’s completely async pipeline is fast
- Large mailboxes are bad, compare error_logger to sync_error_logger
- Lager’s constant rotation checking comes at a price

More conclusions

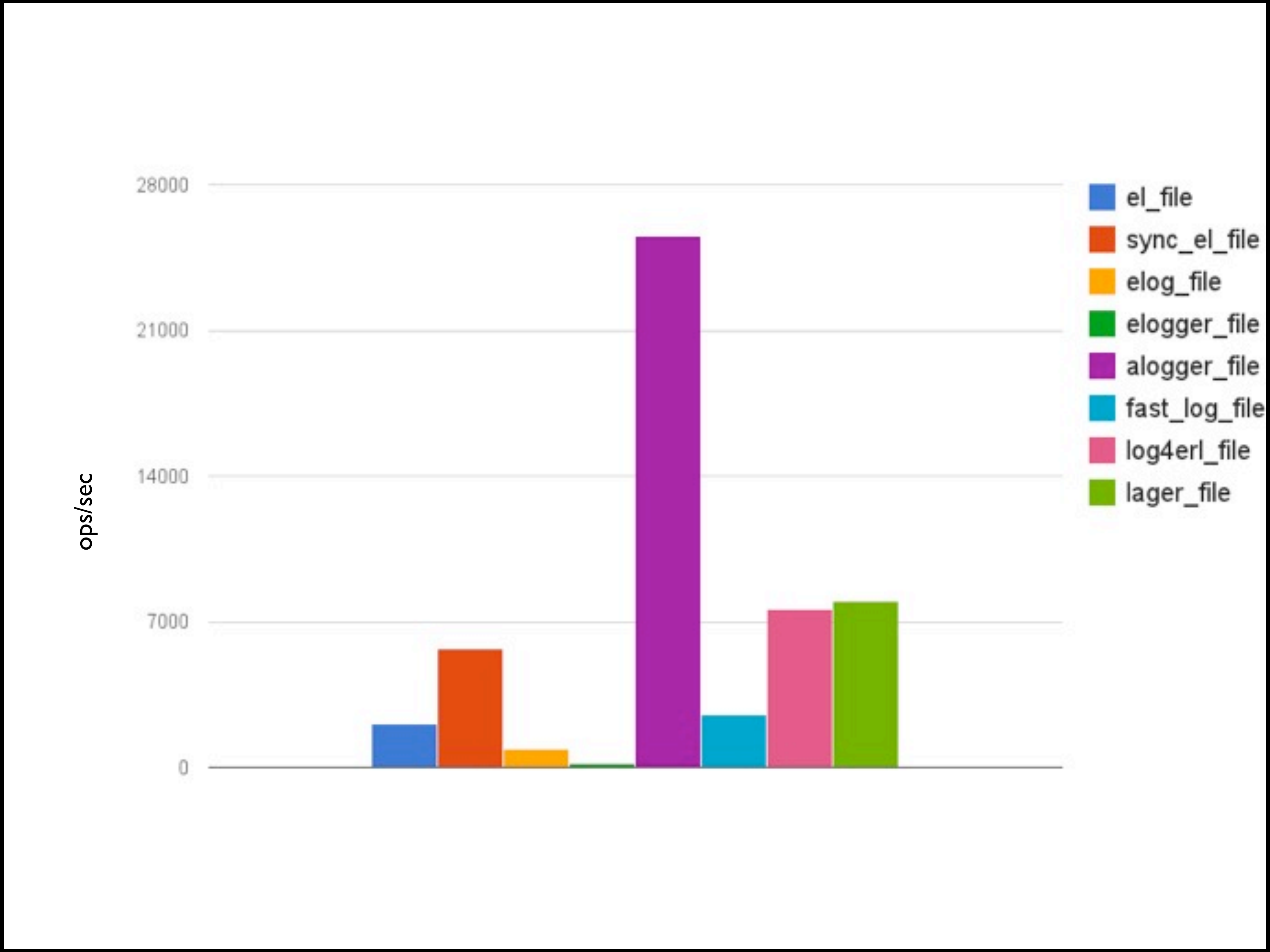
- elogger’s constant syncs are expensive
- Not sure why elog does so badly



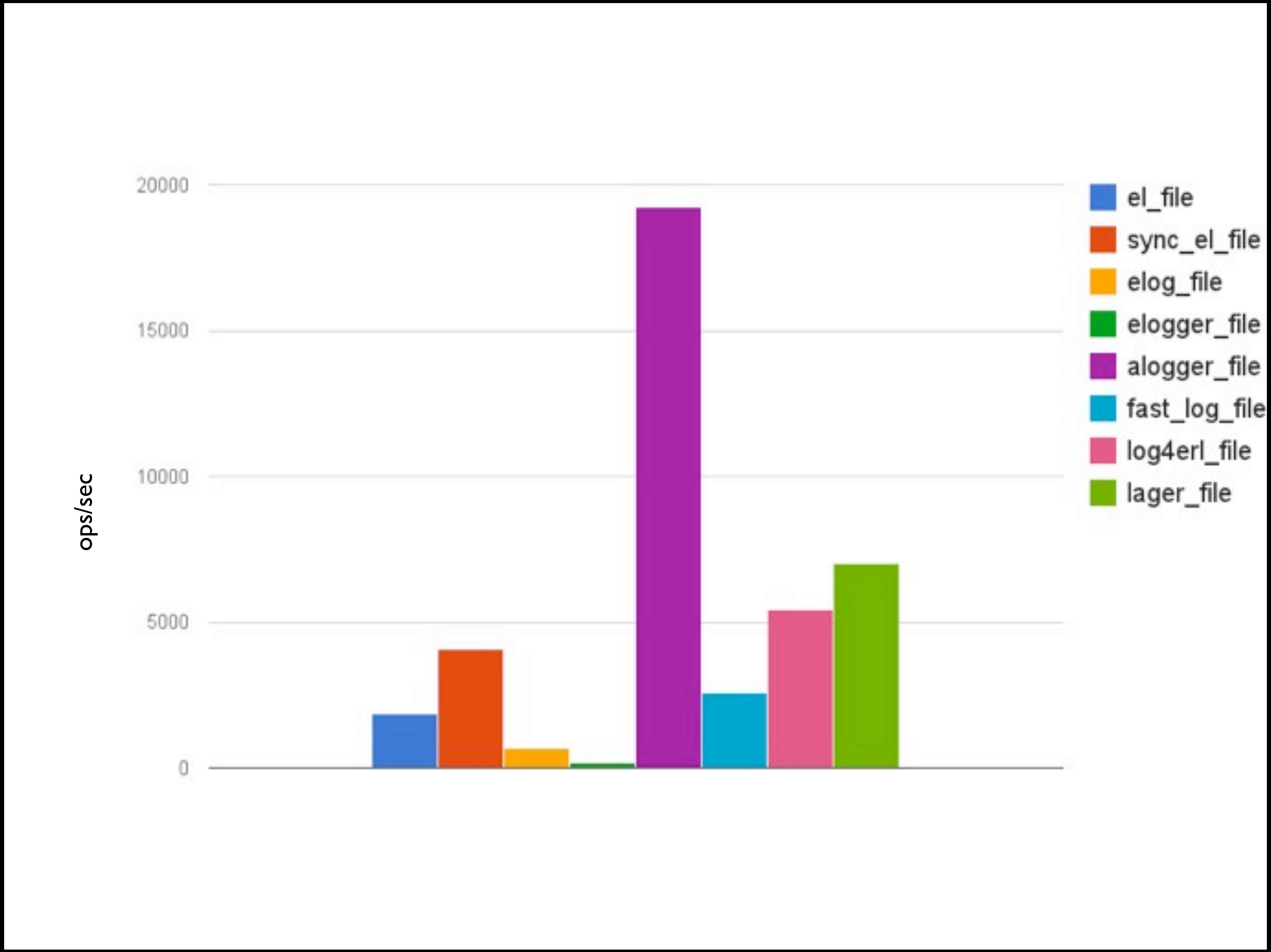
10,000 large, 1 worker

Conclusions

- Most loggers can’t handle these large messages (~80kb), log 10s of messages/sec
- elogger has protection against large binaries
- If a string is used, things get even worse



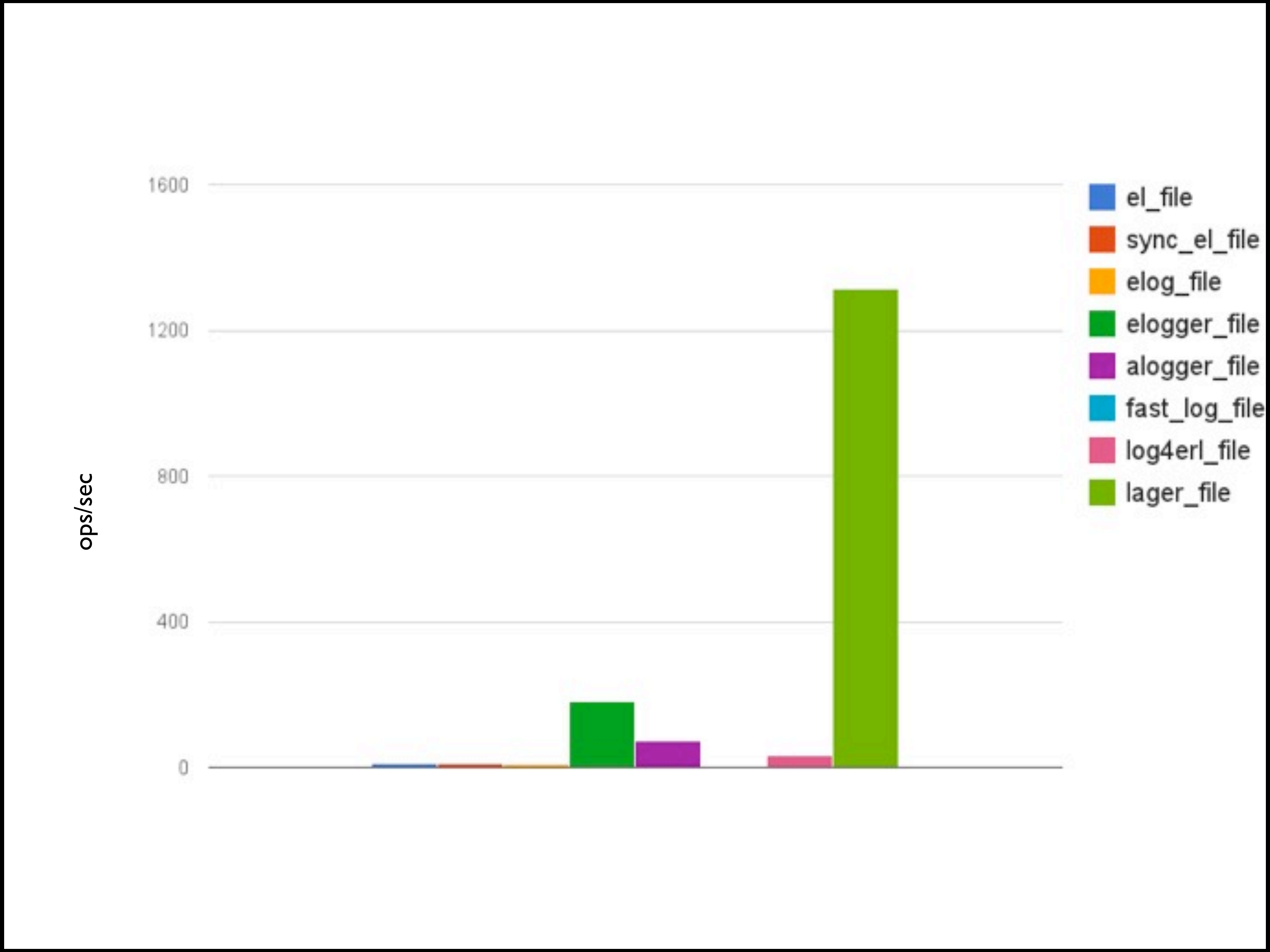
10,000 simple,4 workers



10,000 small,4 workers

Conclusions

- Alogger wins big here because it is completely async and it does the formatting in the calling process – very parallel



10,000 large,4 workers

Crash and burn, again

- alogger and fast_log OOMed, and we were logging off-heap binaries
- fast_log even OOMed before I disabled the large mailbox check
- Lager’s performance is about 3x the single worker test
- elogger manages to hold out

Lager 2.0.0

RC1 tagged today.

New Features

- Metadata is passed to backends*
- Mochiglobal->ETS for config
- Syslog comparison flags: <info, !=warning, etc
- Webmachine/Cowboy error messages
- Application and pdict in metadata

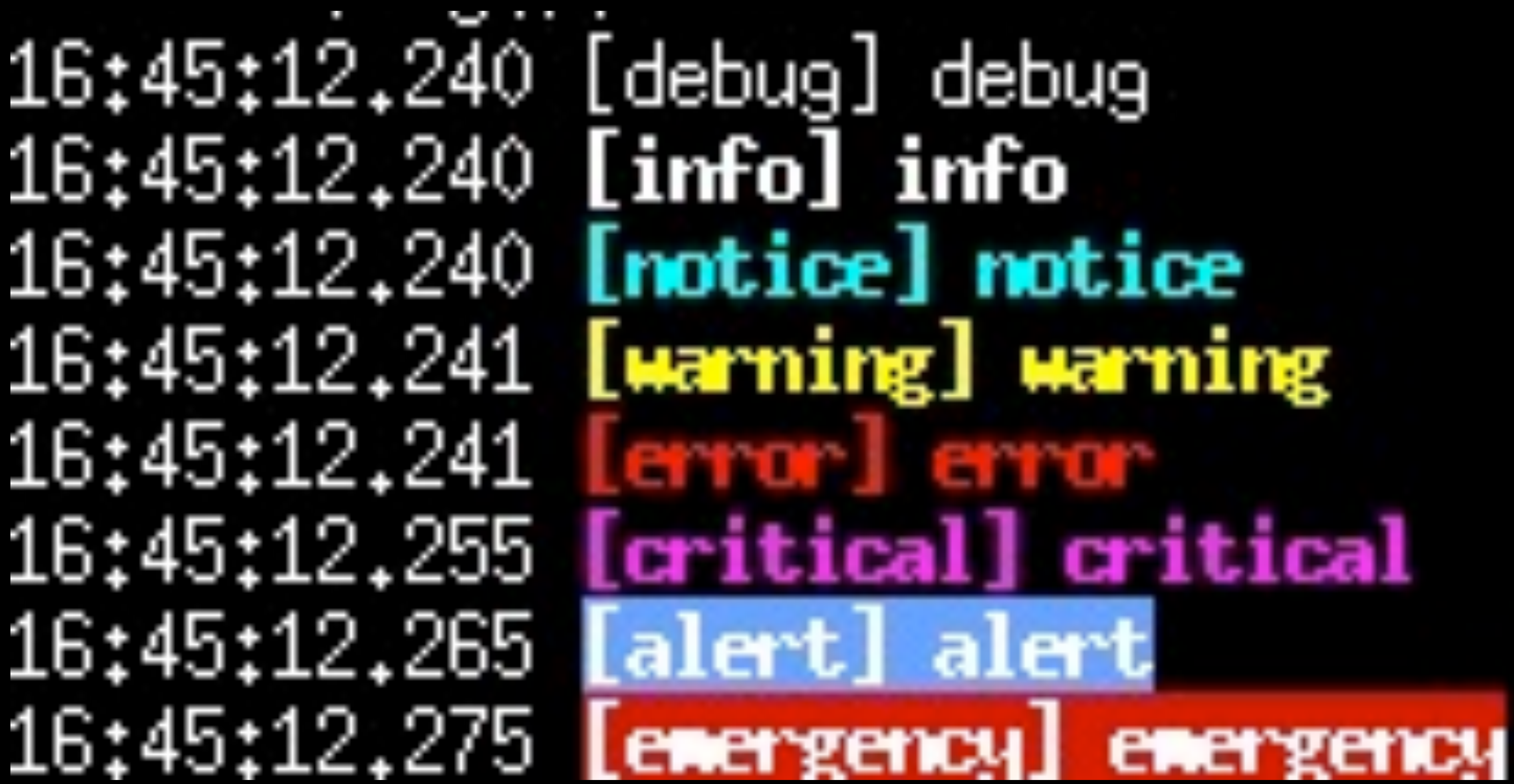
Record printing

```
lager:info("My state is ~p", [lager:pr(State, ?MODULE)]).
```

- Prints

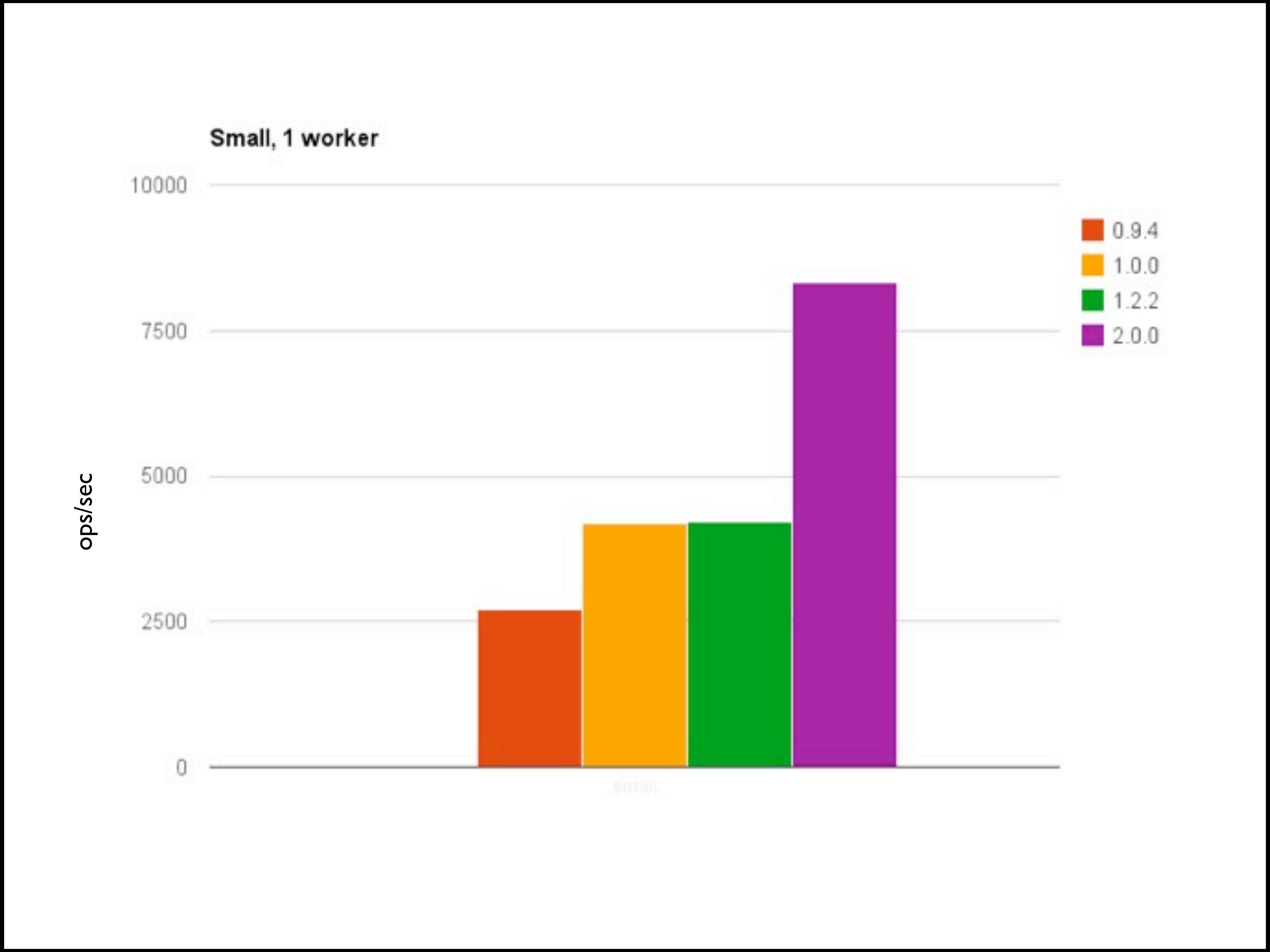
```
00:16:37.888 [info] My state is #state{lions=[simba],tigers=
[tigger],bears=[winnie,yogi],oh_my=true}
```

- Works by spotting records at compile time in the parse transform and storing record info in the module's attributes.
- As long as you know where the record came from, and that module was compiled with lager, you can print it.



```
16:45:12.240 [debug] debug
16:45:12.240 [info] info
16:45:12.240 [notice] notice
16:45:12.241 [warning] warning
16:45:12.241 [error] error
16:45:12.255 [critical] critical
16:45:12.265 [alert] alert
16:45:12.275 [emergency] emergency
```

Console colors, needs R16 – thanks DeadZen

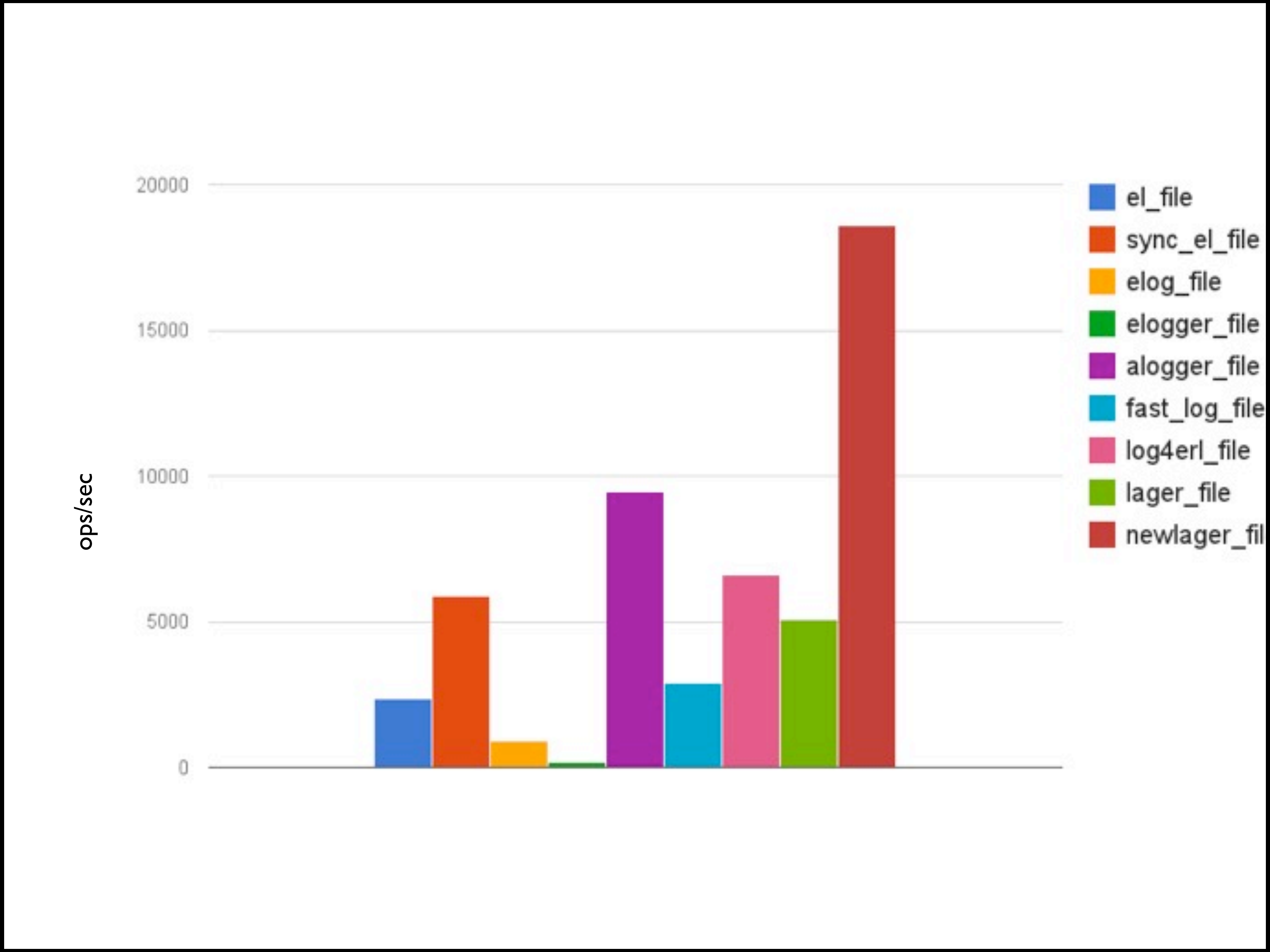


Performance by version

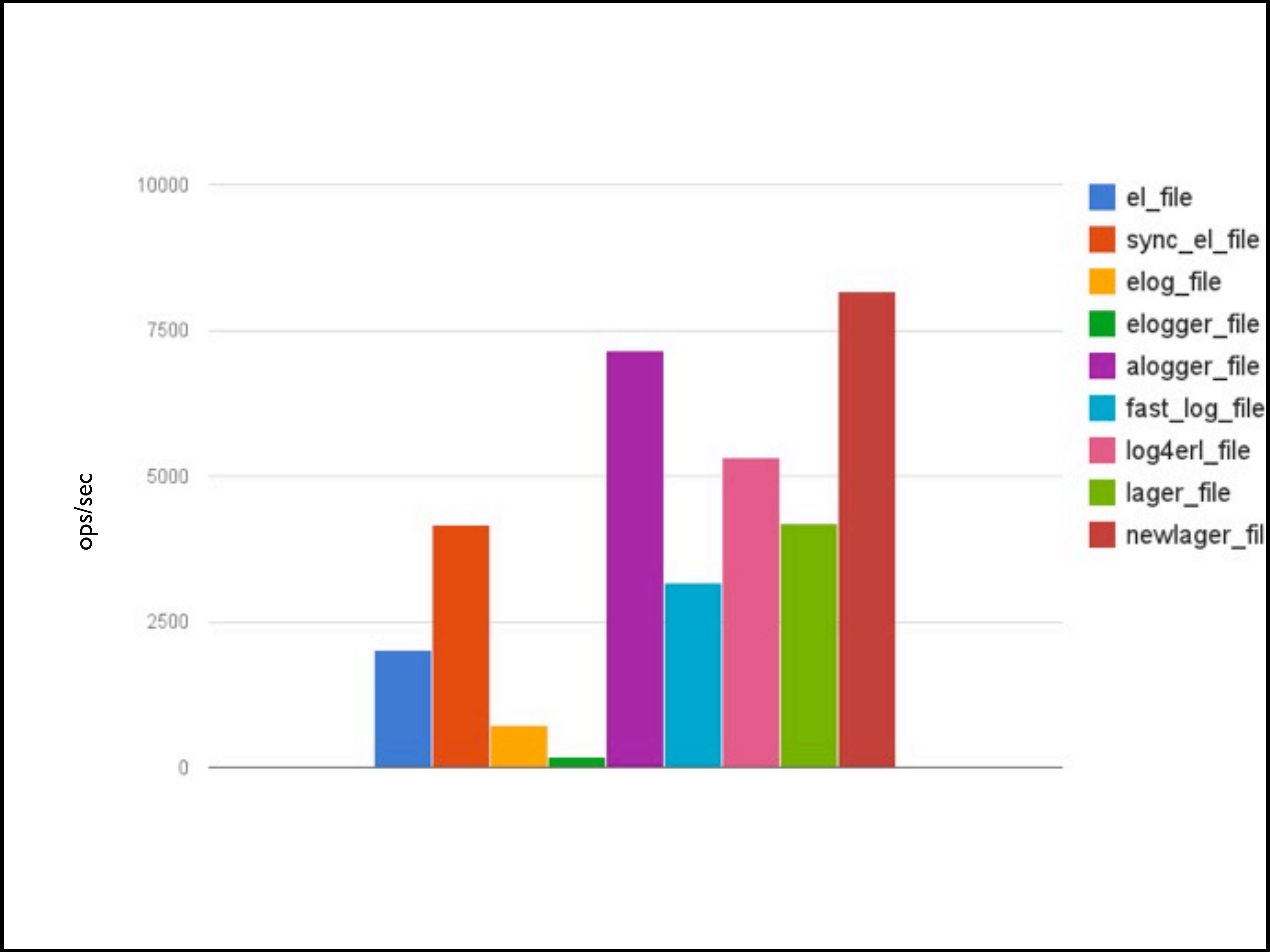
But how?

- Reduce paranoid external checking for file rotation/deletion to once a second, by default
- Make sync/async messaging adaptive based on mailbox size

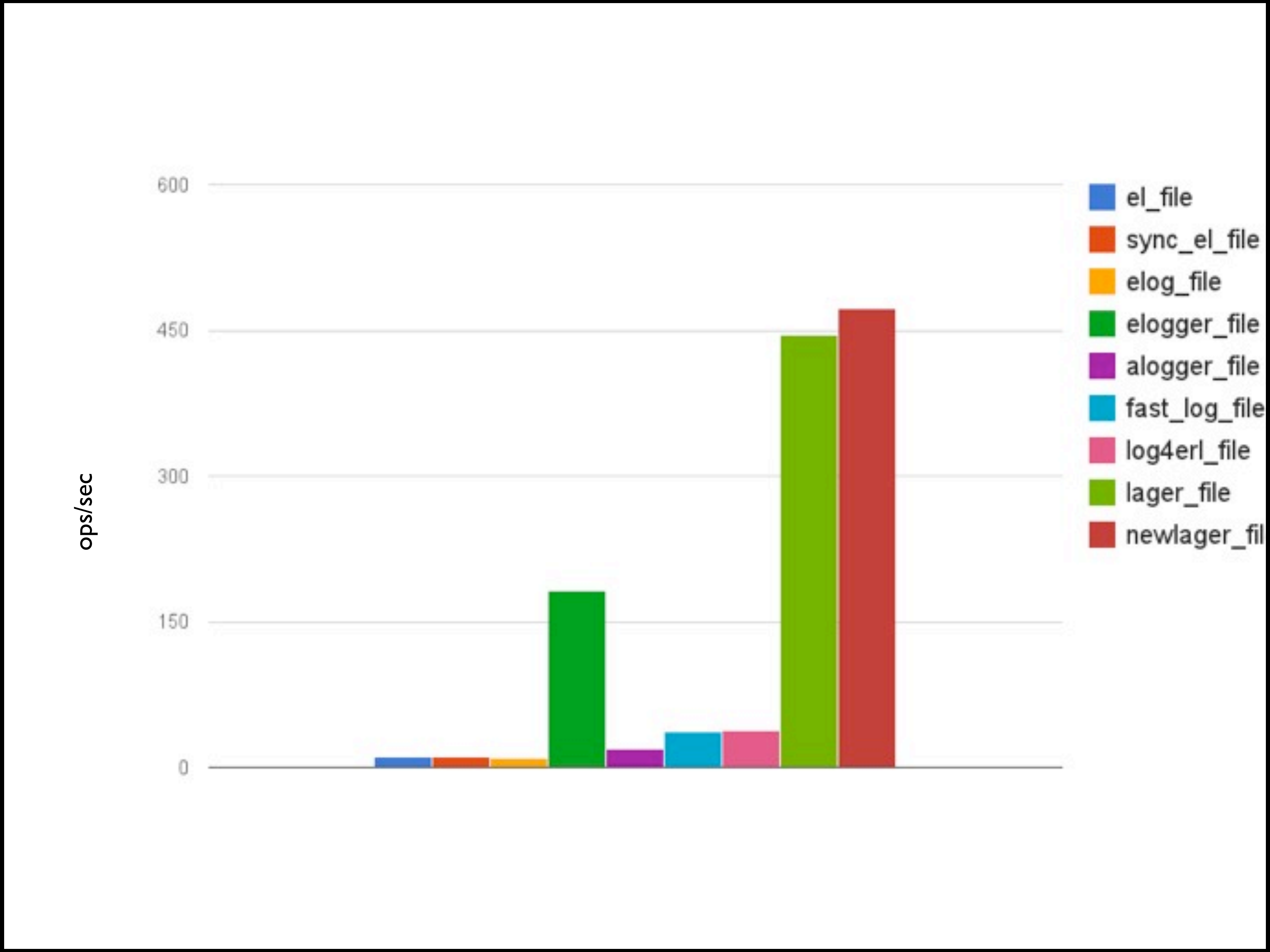
Chart it again, Sam



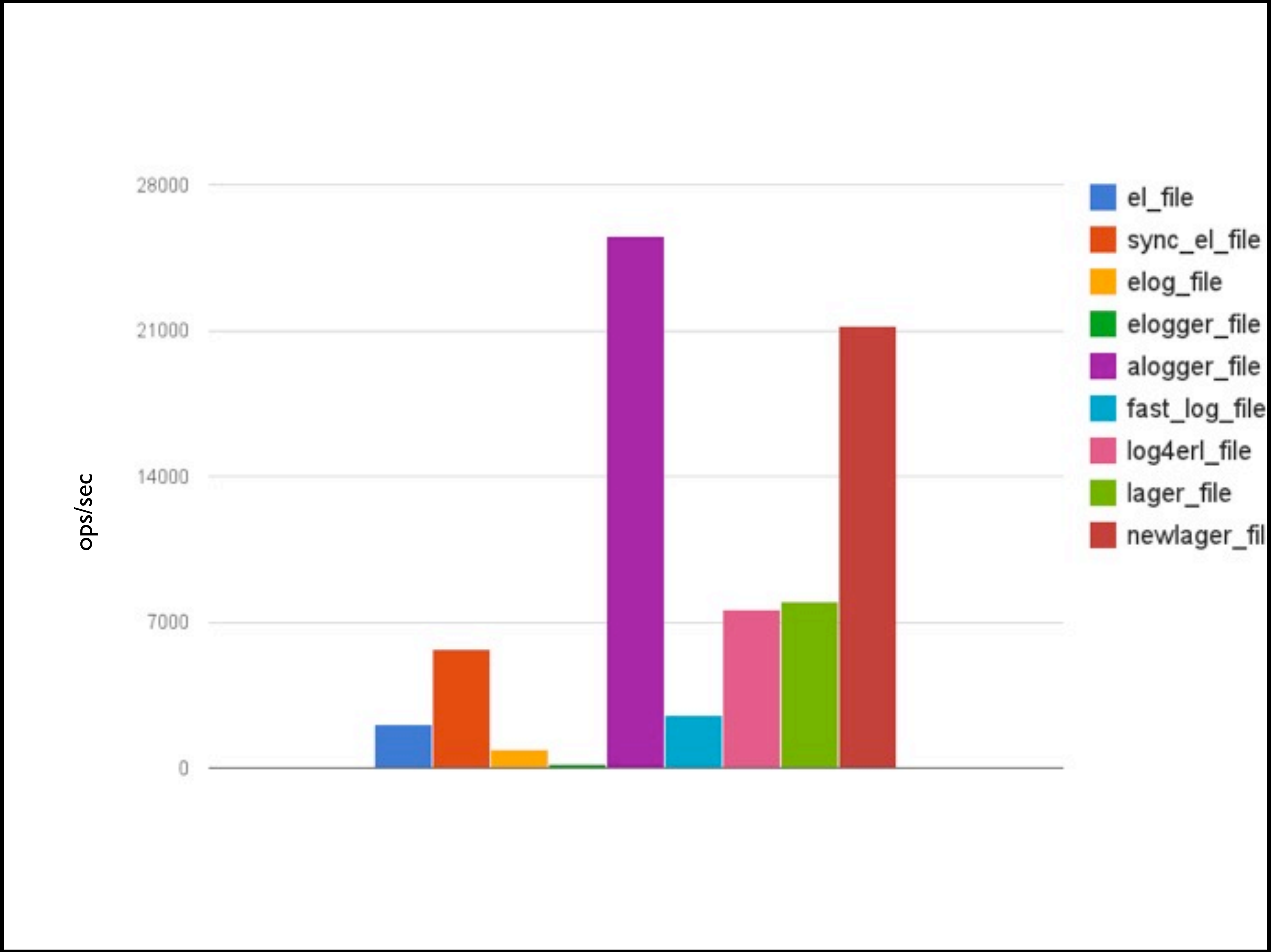
10,000 small, 1 worker



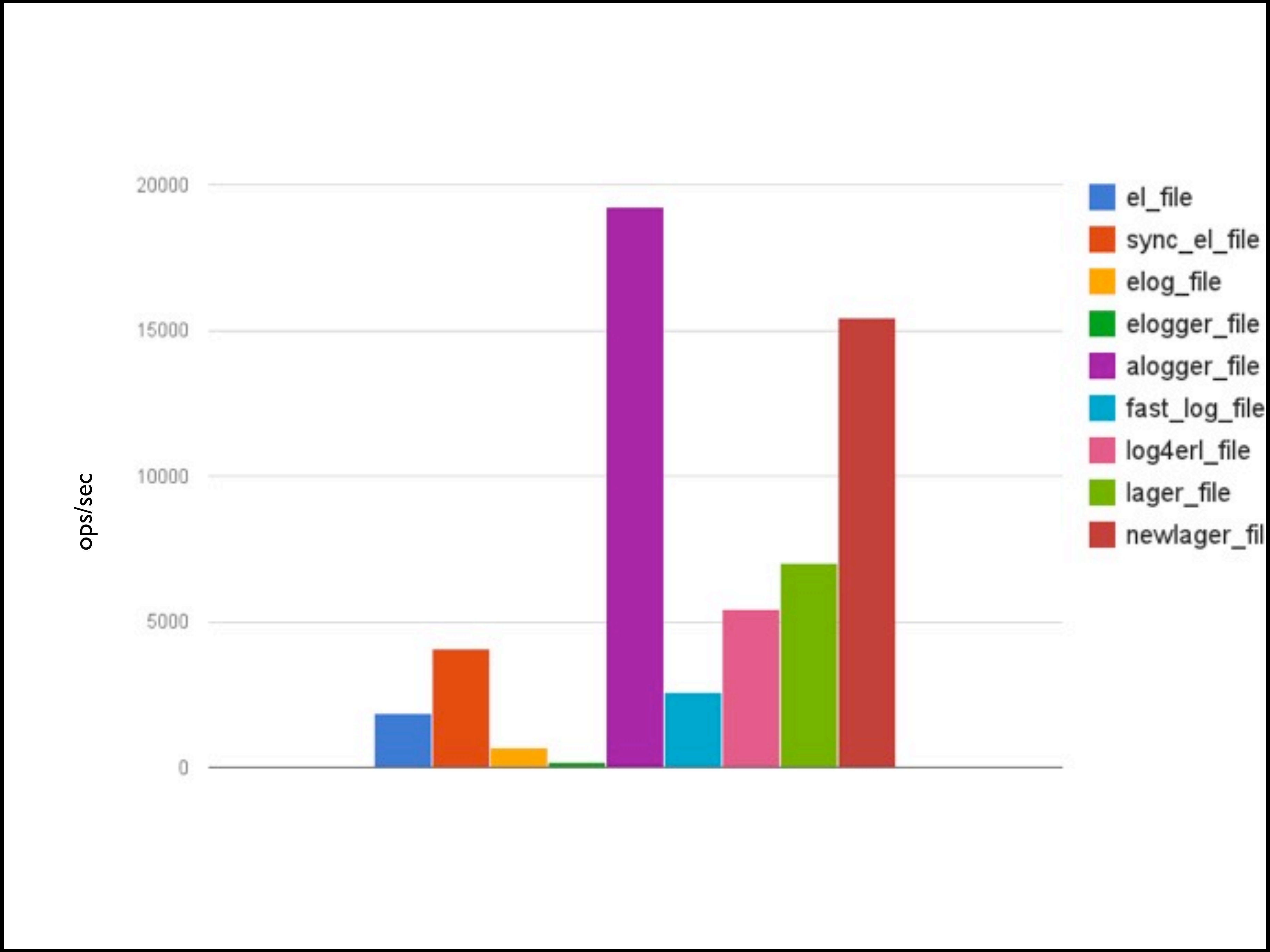
10,000 simple, 1 worker



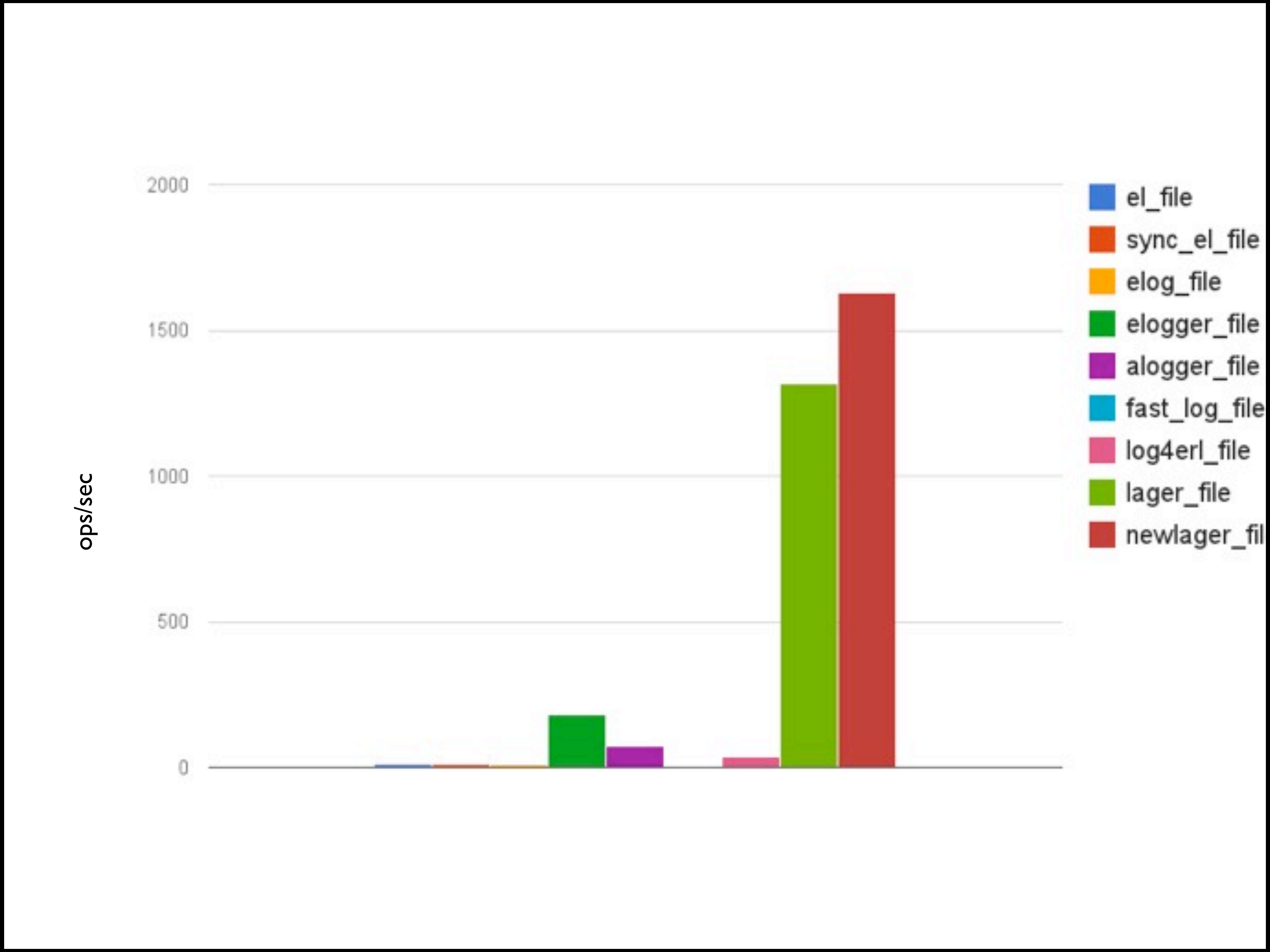
10,000 large, 1 worker



10,000 simple,4 workers



10,000 small,4 workers



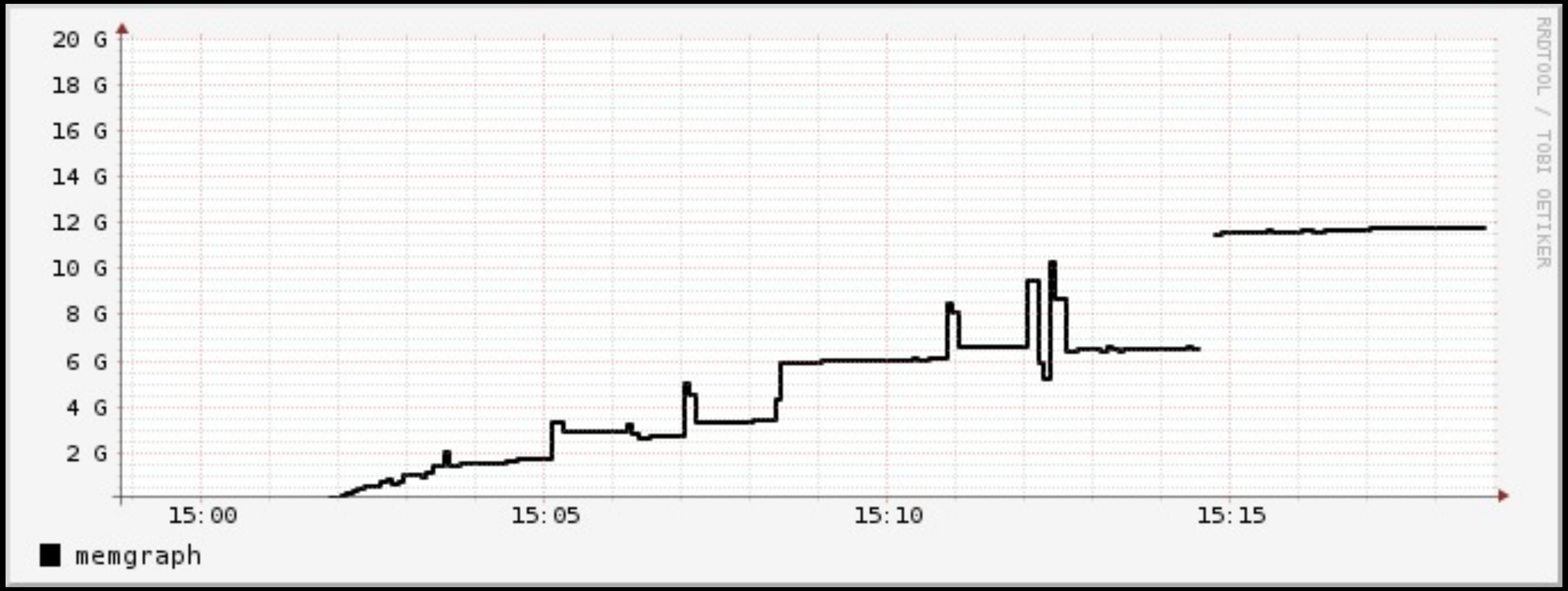
10,000 large,4 workers

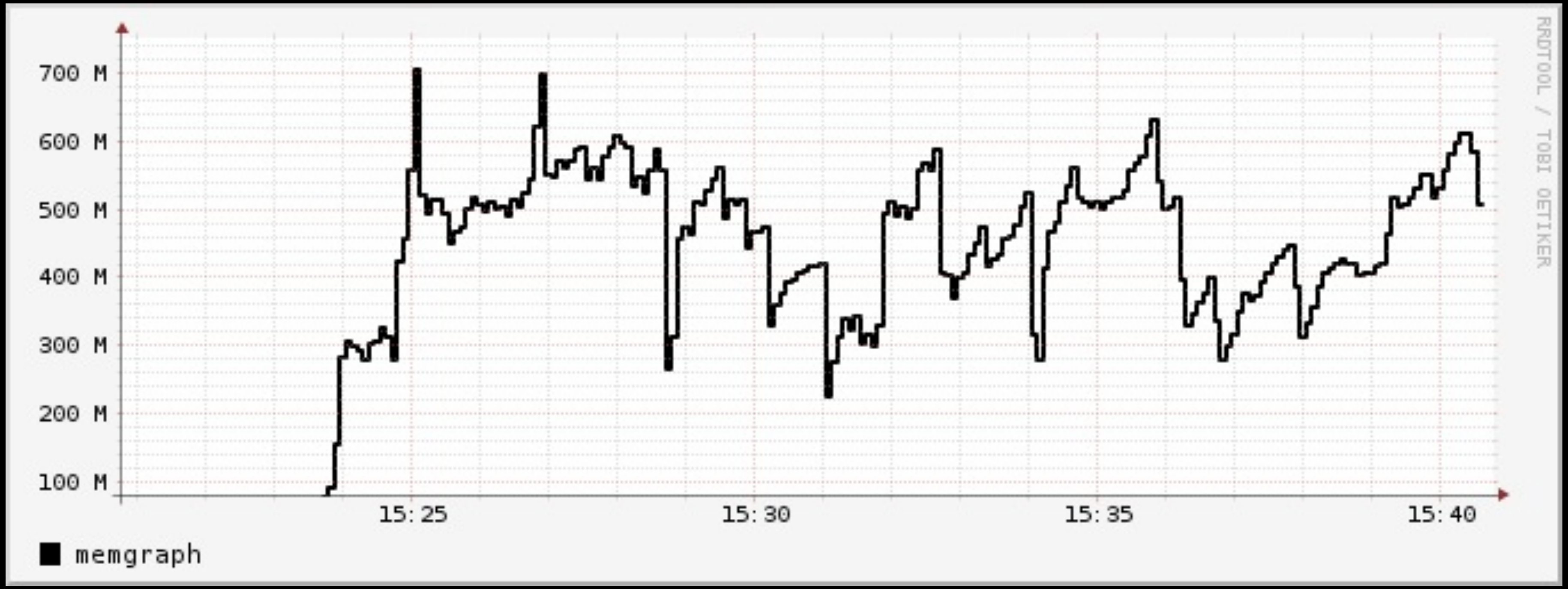
But, error_logger is still async

- As long as you’re using lager, those messages can’t crash your node
- But, OTP errors can still go crazy and crash your node
- This is still a big problem

Cascading-failures

- Written by Fred Hebert, modeled on real world failures
- github.com/ferd/cascading-failures
- Starts a bunch of workers that reference an ETS table, then kills the table owner, mayhem ensues





15 minute run, mem graph

How

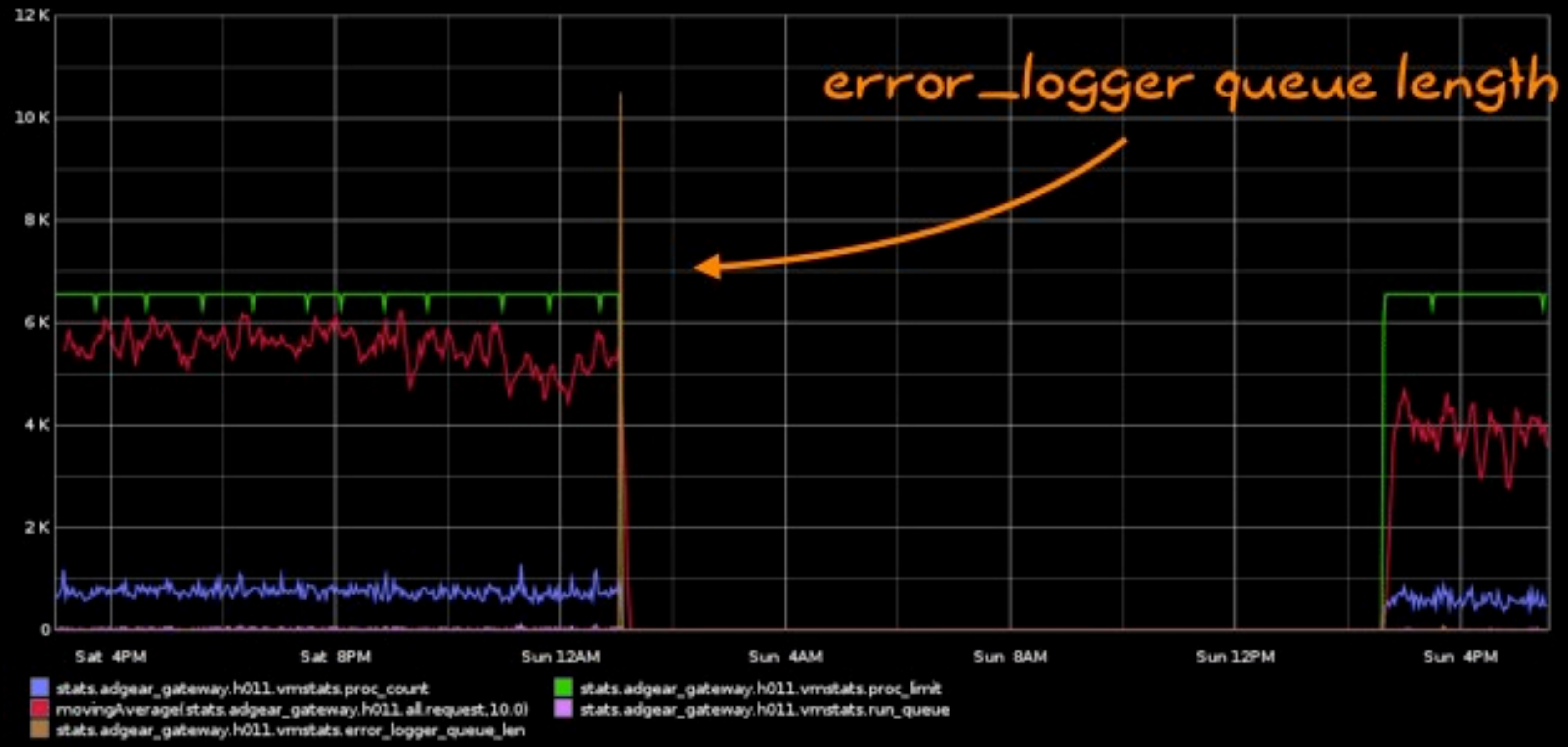
- Lager 2.0 has optional high water mark in error_logger handler
- Once hit, no more messages that second will be logged, it will try to discard them instead
- We do drop messages, but in these cases the messages are often the same

But...

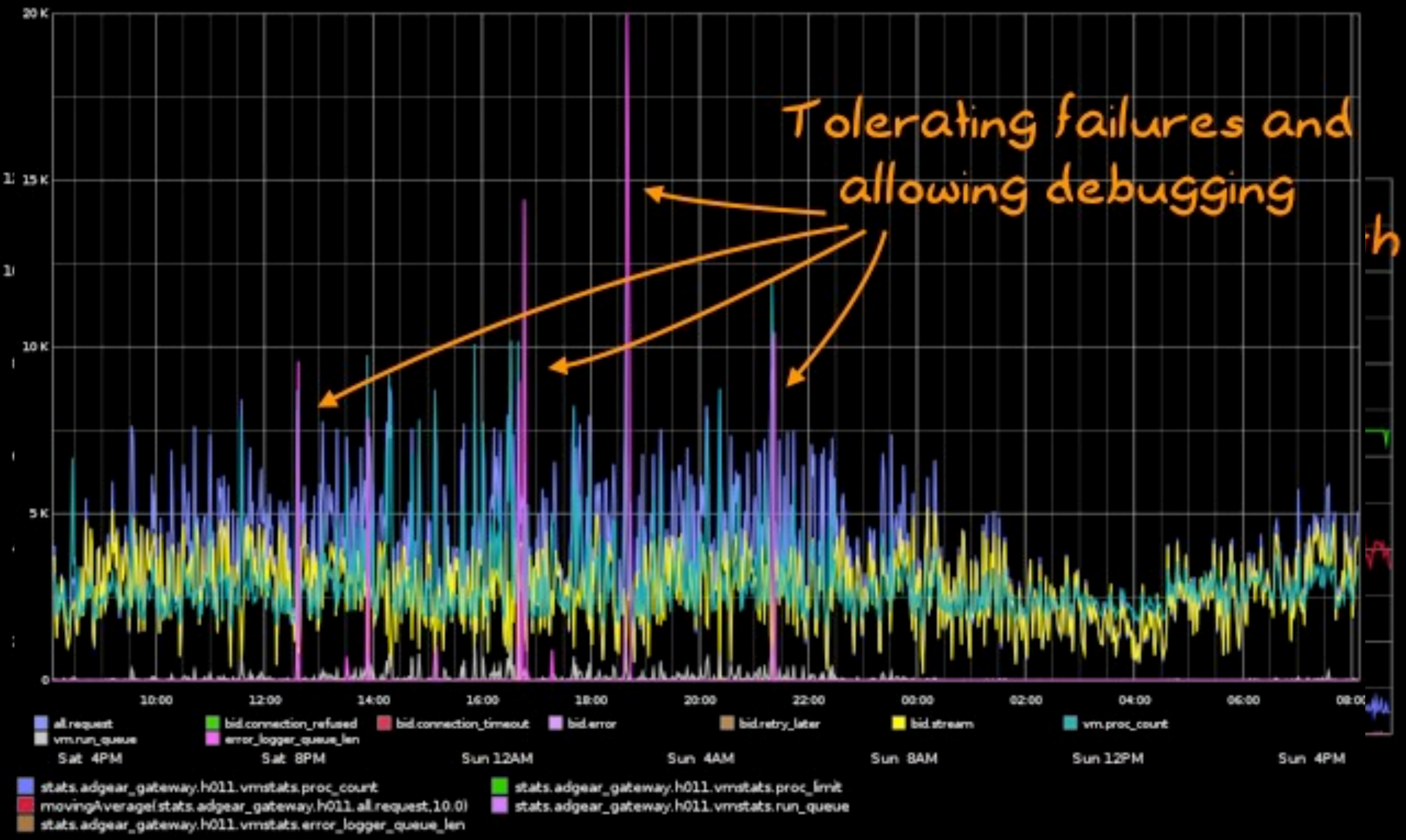
- Still can’t prevent large messages being sent to error_logger
- Use format_status/2 callback for OTP
- When sending large messages, try to use binaries

Lager in production

- Riak since 1.0
- Zotonic since 0.8.0
- AdGear
- Lots more, not sure I can mention
- 19th most watched erlang repo on github, almost 2 years old



error_logger at AdGear



lager at AdGear

Special thanks

- Steve Vinoski for a lot of initial design ideas
- Fred Hebert for sharing his production experience and ideas
- Everyone at Basho for their ideas and reviewing my incessant PRs
- All the users and contributors

Questions?