Perl::Formance Rekapitulacija 2019

Steffen Schwigon (renormalist)
Dresden.pm

PerlCon 2019
Rīga, Latvia, 7-9 August

A new mega shark threatens to destroy humanity. The government
creates an exact robotic copy of the shark, either equal to or greater than the original.
Now they must fight to the death while people and whole cities get in the way.

— Mega Shark vs. Mecha Shark

Perl::Formance in 2019

About

Approach

PerlStone

Numbers 2019

About

Benchmark Perl 5

- not CPAN, Perl 6, architectures, compiler options, ...

Micro vs. Macro benchmarks

Continuously measure vs. “Big once-in-a-year shots”

- Perl Toolchain Summit (formerly known as Perl QA Hackathon)

More:

- http://perlformance.net (slides at the bottom)
- http://renormalist.github.io/Benchmark-Perl-Formance/res/yapc_eu_2015_perlformance2015.pdf

Approach

Build all Perl versions from git

Keep CPAN stable

- sync local cpan mirror once before starting The Big Run
- rebuild all cpan deps from that
  - distroprefs, patchperl, desperately twiddle with latest quirks

Store benchmark results

- own benchmark data schema, database and infrastructure
- http://www.benchmarkanything.org/

Keep benchmarks stable for reliable numbers

- Benchmark::Perl::Formance::Plugin::PerlStone2015

Approach (2)

Perl Toolchain Summit (Perl QA Hackathon)

- thanks to the organizers and sponsors
- http://act.qa-hackathon.org/pts2019/

Bring everything back into shape

- Perl
- CPAN
- Perl::Formance, BenchmarkAnything

Run benchmarks

- back in the prosperous days on dedicated baremetal server
- nowadays on my laptop

PerlStone

Keep benchmarks stable, yet allow maintenance?

Comprehensive benchmark suite (Best-Of Perl::Formance plugins)

Hard-versioned benchmark plugin, i.e., PerlStone2015

PerlStone2015 contains

- replicas of separate existing Perl::Formance plugins
- micro benchmarks along chapters in Camel Book
  - (not much implemented)

My personal point of interest...

- "...if I had more time"(TM)

Substantial rework? → fork into PerlStone2021

Numbers 2019

See yourself

- http://perlformance.net
- http://renormalist.github.io/Benchmark-Perl-Formance/charts/pts2019-f/index.html
  → results from April 2019 at the “Perl Toolchain Summit 2019”
- shortly before 5.30 release

Remember - Smaller is better!

- as I measure execution time

Numbers 2019

Stupid, over-generalized, un-quotable summary:

- Perl once got slower (back in the 5.16 to 5.22 days)
- since v5.22 gets notably faster
  - on data structures, execution overhead, common programming patterns, edge cases
  - → helps algorithmic code
  - → Programming Language Shootouts
- gets slightly slower
  - on regular expressions
  - → can still put down other optimizations
- **5.30 is the fastest modern Perl we ever had**

FORK ME ON GITHUB

PERL::FORMANCE

Perl benchmark suite

download .ZIP

download .TGZ

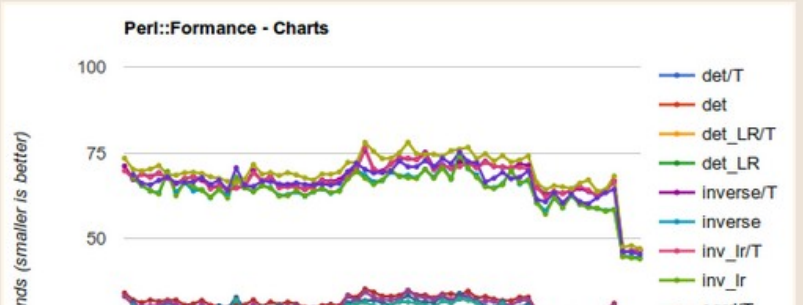
Welcome to Perl::Formance.

This is about benchmarking Perl.

It combines several projects: the actual benchmark suite, tools to bootstrap a complete Perl+CPAN from git, a results database and chart rendering.

Where are the results?

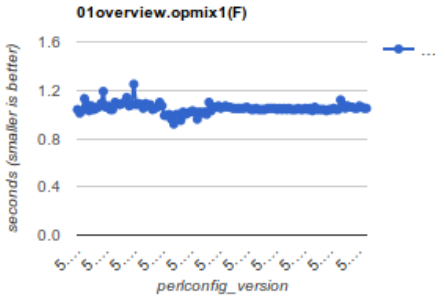
Click here for Perl 5 results:

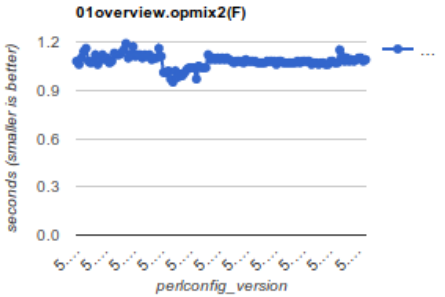


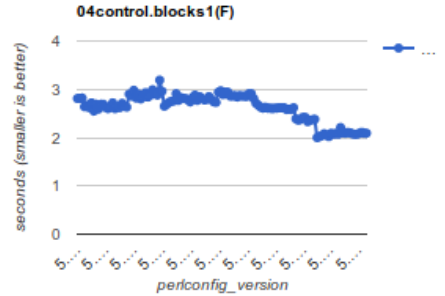
Perl::Formance - pts2019-f

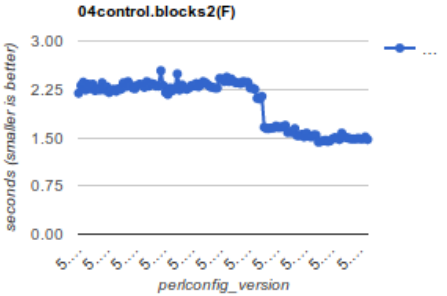
- Raw numbers used on this page with their statistical description
- BenchmarkAnything::Evaluations collecting the statistics
- Benchmark::Perl::Formance plugins on CPAN

- X-axis: perl version
- Y-axis: execution time in seconds (*smaller is better*)









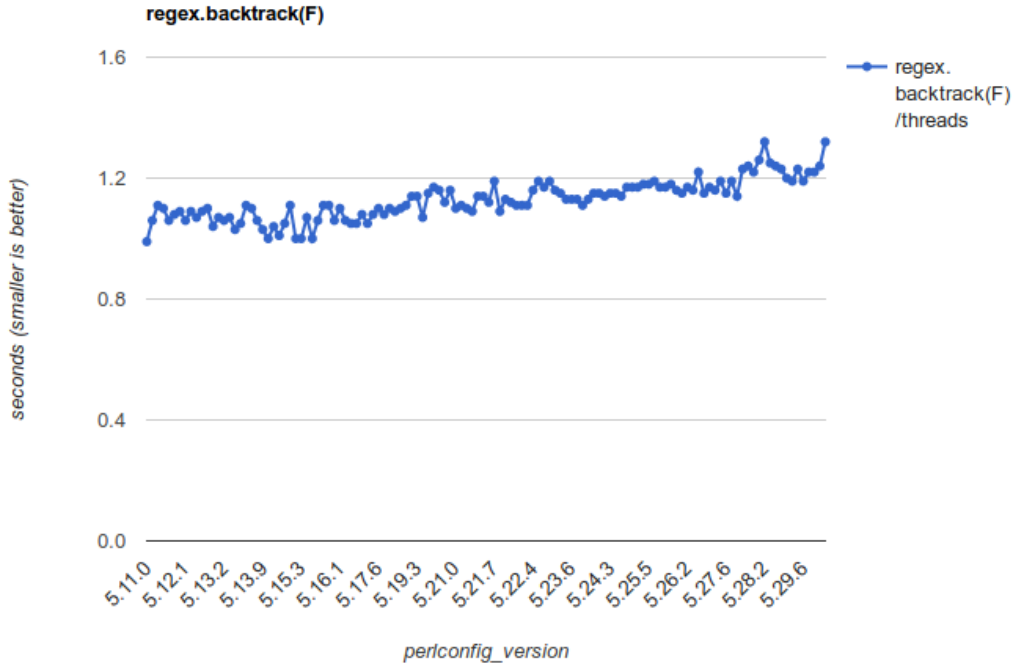
```
Perl::Formance - charts rendered at: Fri May  3 21:43:12 2019

ci95l - confidence intervall 95 lower
ci95u - confidence intervall 95 upper
avg   - average
stdv  - standard deviation

* 01overview.opmix1(F)/threads - perlformance.perl5.PerlStone2015.01overview.opmix1(F)
  01overview.opmix1(F)/threads . 5.28.2  . (ci95l..avg..ci95u) = (1.10 .. 1.12 .. 1.14) +- stdv  0.05 ( 38 points)
  01overview.opmix1(F)/threads . 5.11.0  . (ci95l..avg..ci95u) = (0.99 .. 1.04 .. 1.09) +- stdv  0.03 (  2 points)
  01overview.opmix1(F)/threads . 5.11.5  . (ci95l..avg..ci95u) = (0.99 .. 1.03 .. 1.07) +- stdv  0.02 (  2 points)
  01overview.opmix1(F)/threads . 5.12.5  . (ci95l..avg..ci95u) = (NaN .. 1.19 .. NaN) +- stdv  0.00 (  1 points)
  01overview.opmix1(F)/threads . 5.27.7  . (ci95l..avg..ci95u) = (1.02 .. 1.03 .. 1.04) +- stdv  0.01 ( 15 points)
  01overview.opmix1(F)/threads . 5.13.6  . (ci95l..avg..ci95u) = (1.07 .. 1.08 .. 1.10) +- stdv  0.03 ( 12 points)
  01overview.opmix1(F)/threads . 5.25.2  . (ci95l..avg..ci95u) = (1.04 .. 1.05 .. 1.05) +- stdv  0.01 ( 15 points)
  01overview.opmix1(F)/threads . 5.15.3  . (ci95l..avg..ci95u) = (1.03 .. 1.05 .. 1.07) +- stdv  0.03 (  9 points)
  01overview.opmix1(F)/threads . 5.19.2  . (ci95l..avg..ci95u) = (1.01 .. 1.02 .. 1.04) +- stdv  0.02 ( 12 points)
  01overview.opmix1(F)/threads . 5.27.4  . (ci95l..avg..ci95u) = (1.03 .. 1.04 .. 1.04) +- stdv  0.02 ( 15 points)
  01overview.opmix1(F)/threads . 5.22.0  . (ci95l..avg..ci95u) = (1.04 .. 1.05 .. 1.05) +- stdv  0.01 ( 14 points)
  01overview.opmix1(F)/threads . 5.16.1  . (ci95l..avg..ci95u) = (NaN .. 1.10 .. NaN) +- stdv  0.00 (  1 points)
  01overview.opmix1(F)/threads . 5.22.1  . (ci95l..avg..ci95u) = (1.04 .. 1.05 .. 1.05) +- stdv  0.01 ( 14 points)
  01overview.opmix1(F)/threads . 5.15.2  . (ci95l..avg..ci95u) = (1.04 .. 1.08 .. 1.12) +- stdv  0.07 ( 10 points)
  01overview.opmix1(F)/threads . 5.17.7  . (ci95l..avg..ci95u) = (0.98 .. 1.00 .. 1.02) +- stdv  0.02 (  7 points)
  01overview.opmix1(F)/threads . 5.11.4  . (ci95l..avg..ci95u) = (0.94 .. 1.08 .. 1.22) +- stdv  0.07 (  2 points)
  01overview.opmix1(F)/threads . 5.17.5  . (ci95l..avg..ci95u) = (0.91 .. 0.92 .. 0.94) +- stdv  0.02 (  8 points)
  01overview.opmix1(F)/threads . 5.29.9  . (ci95l..avg..ci95u) = (1.03 .. 1.05 .. 1.06) +- stdv  0.03 ( 21 points)
  01overview.opmix1(F)/threads . 5.15.1  . (ci95l..avg..ci95u) = (1.05 .. 1.09 .. 1.13) +- stdv  0.06 ( 11 points)
  01overview.opmix1(F)/threads . 5.19.7  . (ci95l..avg..ci95u) = (1.00 .. 1.01 .. 1.03) +- stdv  0.03 ( 13 points)
  01overview.opmix1(F)/threads . 5.23.3  . (ci95l..avg..ci95u) = (1.04 .. 1.04 .. 1.05) +- stdv  0.02 ( 14 points)
  01overview.opmix1(F)/threads . 5.15.0  . (ci95l..avg..ci95u) = (1.06 .. 1.08 .. 1.10) +- stdv  0.04 ( 12 points)
  01overview.opmix1(F)/threads . 5.17.8  . (ci95l..avg..ci95u) = (0.93 .. 0.95 .. 0.97) +- stdv  0.02 (  7 points)
  01overview.opmix1(F)/threads . 5.27.3  . (ci95l..avg..ci95u) = (1.03 .. 1.04 .. 1.05) +- stdv  0.02 ( 15 points)
  01overview.opmix1(F)/threads . 5.22.4  . (ci95l..avg..ci95u) = (1.04 .. 1.05 .. 1.06) +- stdv  0.02 ( 14 points)
  01overview.opmix1(F)/threads . 5.27.0  . (ci95l..avg..ci95u) = (1.04 .. 1.05 .. 1.06) +- stdv  0.02 ( 15 points)
  01overview.opmix1(F)/threads . 5.21.4  . (ci95l..avg..ci95u) = (1.05 .. 1.07 .. 1.08) +- stdv  0.01 (  5 points)
  01overview.opmix1(F)/threads . 5.17.1  . (ci95l..avg..ci95u) = (0.98 .. 0.99 .. 0.99) +- stdv  0.01 ( 19 points)
  01overview.opmix1(F)/threads . 5.23.9  . (ci95l..avg..ci95u) = (1.03 .. 1.04 .. 1.05) +- stdv  0.02 ( 14 points)
  01overview.opmix1(F)/threads . 5.24.0  . (ci95l..avg..ci95u) = (1.04 .. 1.05 .. 1.05) +- stdv  0.02 ( 25 points)
  01overview.opmix1(F)/threads . 5.23.2  . (ci95l..avg..ci95u) = (1.04 .. 1.05 .. 1.05) +- stdv  0.01 ( 14 points)
  01overview.opmix1(F)/threads . 5.11.1  . (ci95l..avg..ci95u) = (1.00 .. 1.01 .. 1.03) +- stdv  0.01 (  2 points)
  01overview.opmix1(F)/threads . 5.25.8  . (ci95l..avg..ci95u) = (1.03 .. 1.05 .. 1.06) +- stdv  0.02 ( 15 points)
  01overview.opmix1(F)/threads . 5.22.2  . (ci95l..avg..ci95u) = (1.04 .. 1.05 .. 1.05) +- stdv  0.02 ( 14 points)
  01overview.opmix1(F)/threads . 5.23.8  . (ci95l..avg..ci95u) = (1.03 .. 1.04 .. 1.05) +- stdv  0.01 ( 14 points)
  01overview.opmix1(F)/threads . 5.27.9  . (ci95l..avg..ci95u) = (1.03 .. 1.04 .. 1.06) +- stdv  0.03 ( 15 points)
  01overview.opmix1(F)/threads . 5.24.2  . (ci95l..avg..ci95u) = (1.04 .. 1.05 .. 1.05) +- stdv  0.02 ( 25 points)
```

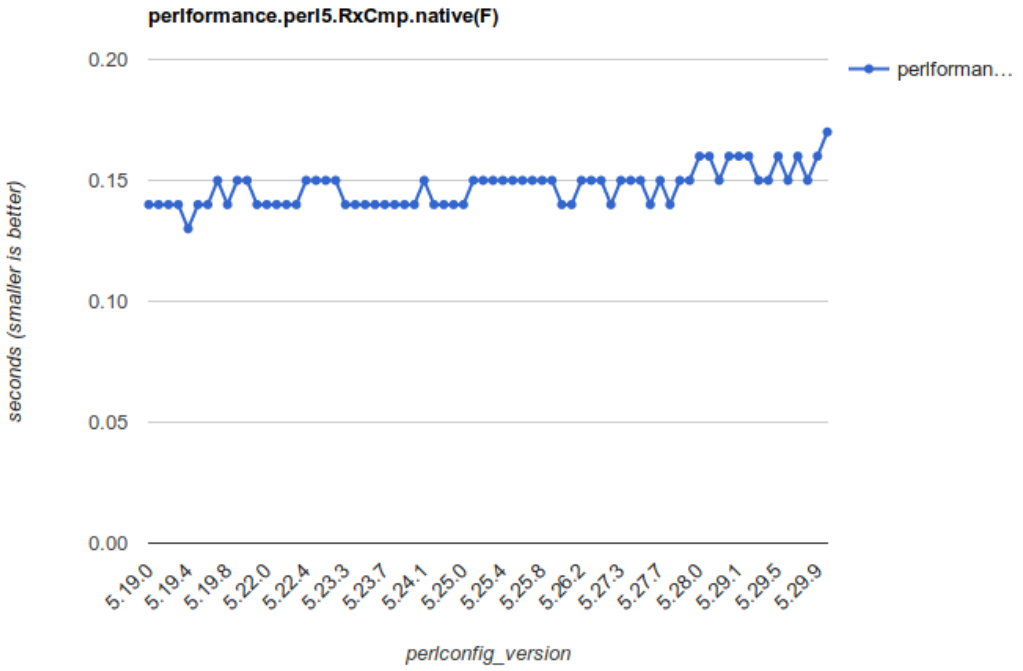
Perl::Formance - pts2019-f - regex.backtrack(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::PerlStone2015::regex



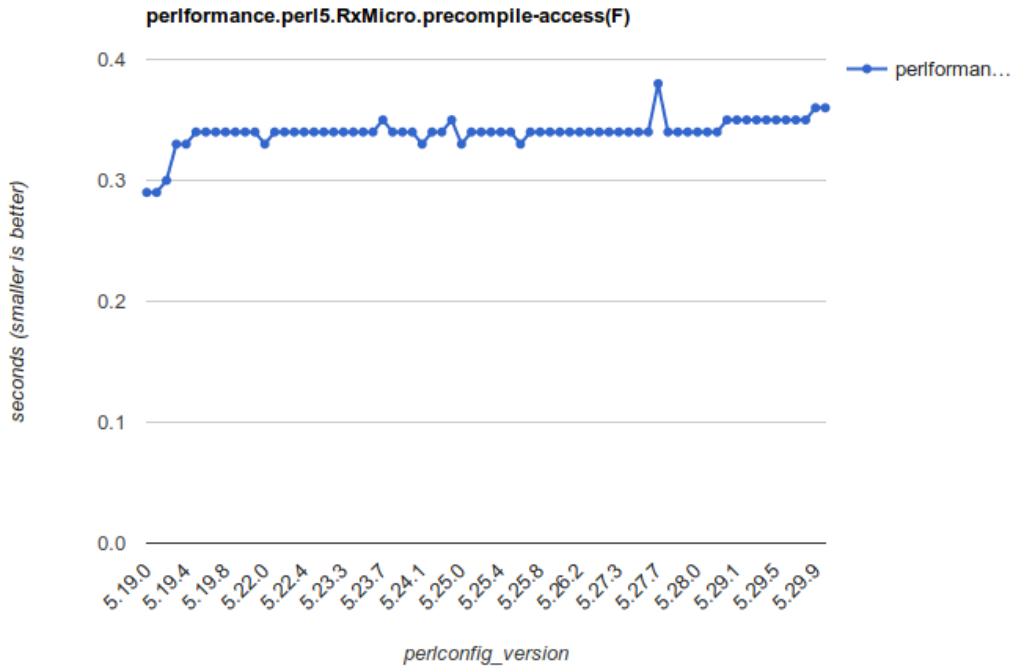
Perl::Formance - pts2019-f - perlformance.perl5.RxCmp.native(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::RxCmp



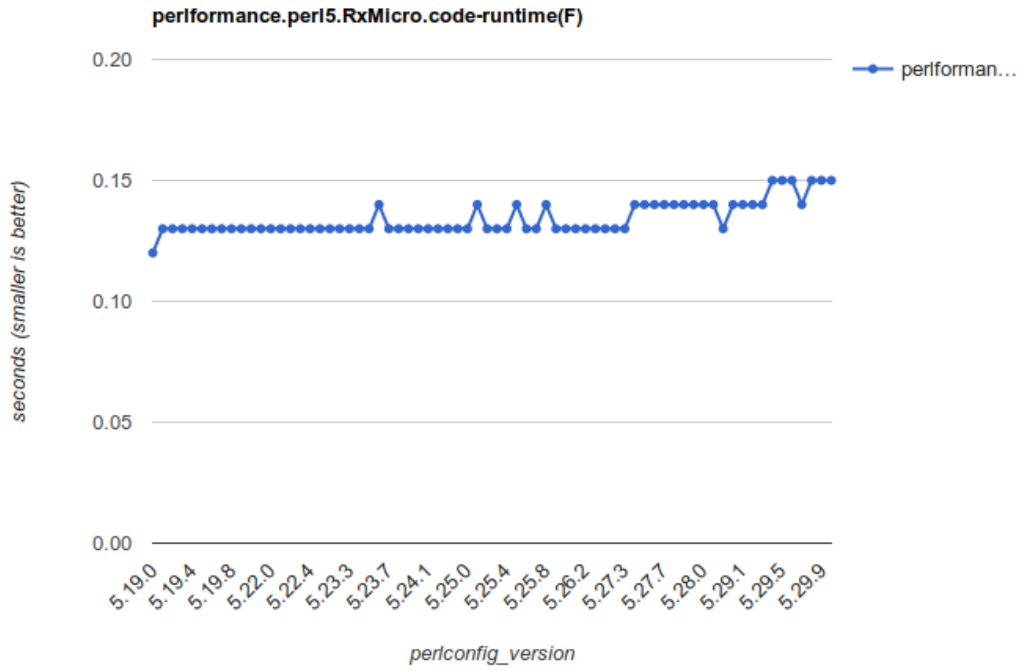
Perl::Formance - pts2019-f - perlformance.perl5.RxMicro.precompile-access(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::RxMicro



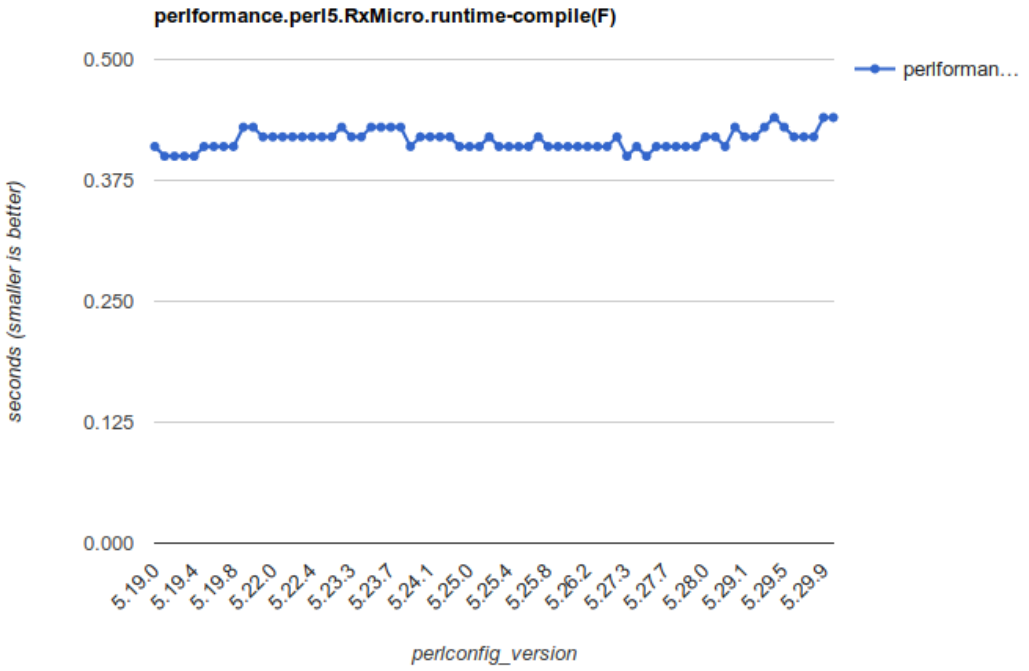
Perl::Formance - pts2019-f - perlformance.perl5.RxMicro.code-runtime(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::RxMicro



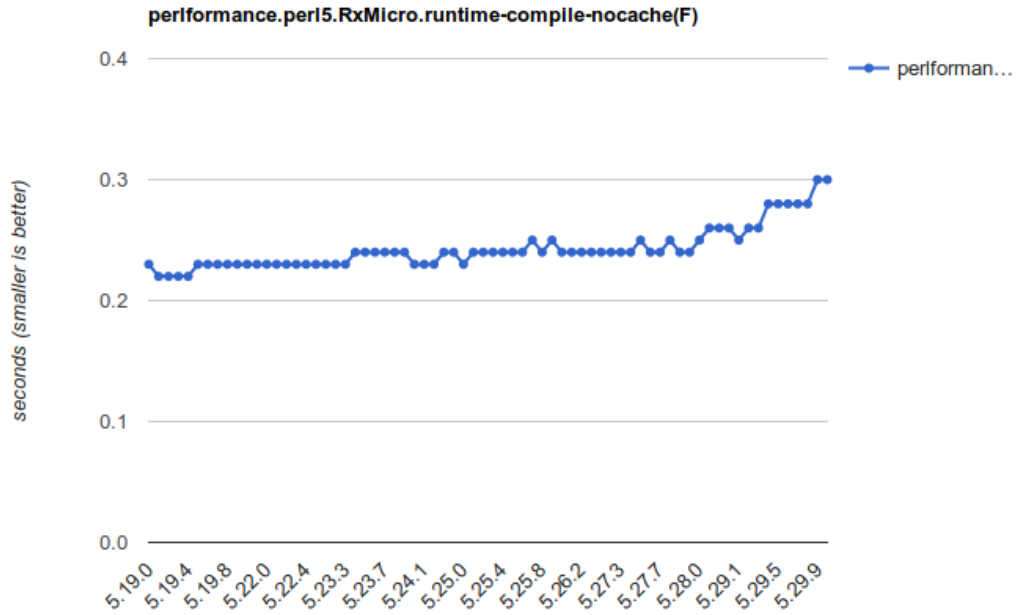
Perl::Formance - pts2019-f - perlformance.perl5.RxMicro.runtime-compile(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::RxMicro



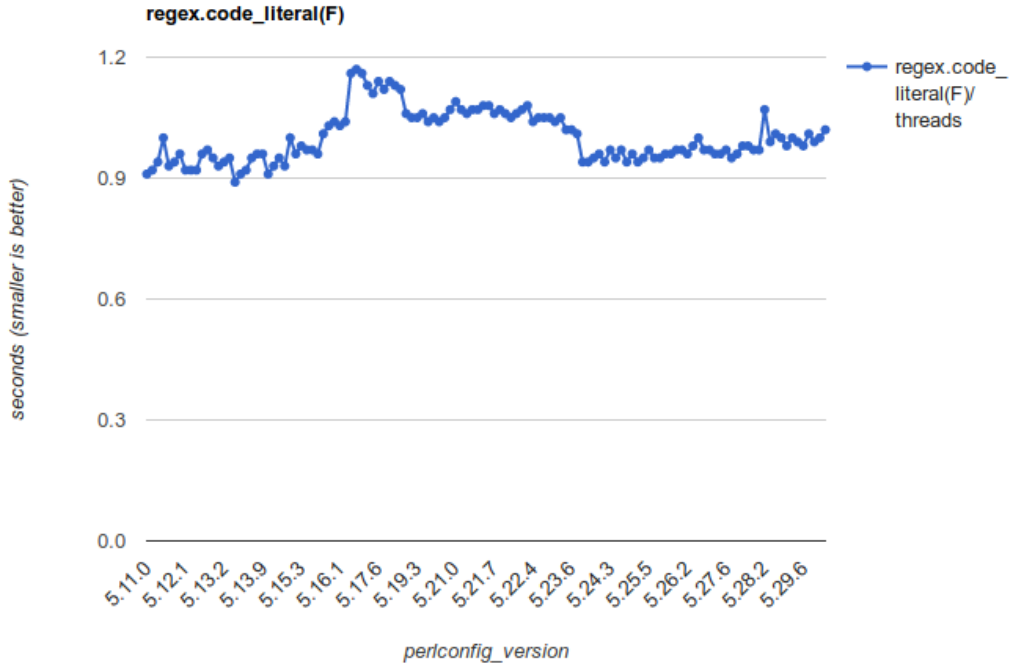
Perl::Formance - pts2019-f - perlformance.perl5.RxMicro.runtime-compile-nocache(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::RxMicro



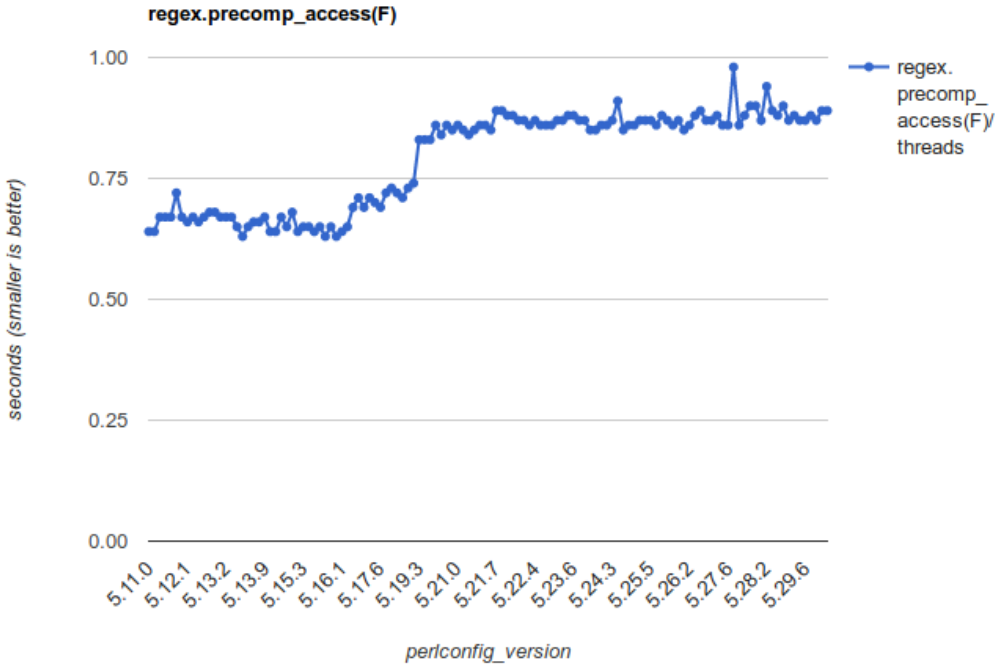
Perl::Formance - pts2019-f - regex.code_literal(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::PerlStone2015::regex



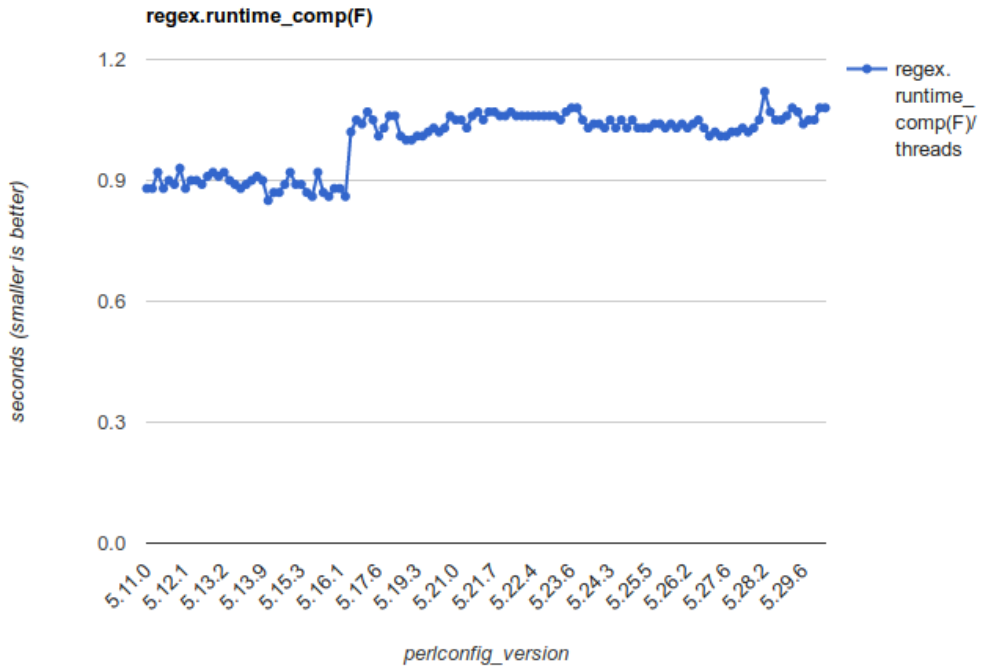
Perl::Formance - pts2019-f - regex.precomp_access(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::PerlStone2015::regex



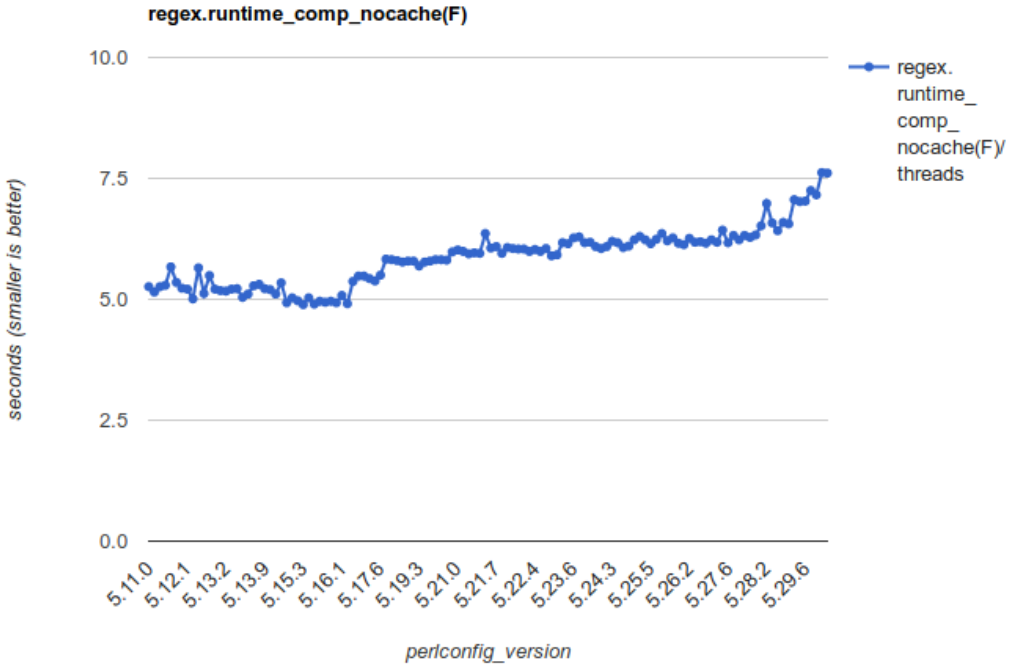
Perl::Formance - pts2019-f - regex.runtime_comp(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::PerlStone2015::regex



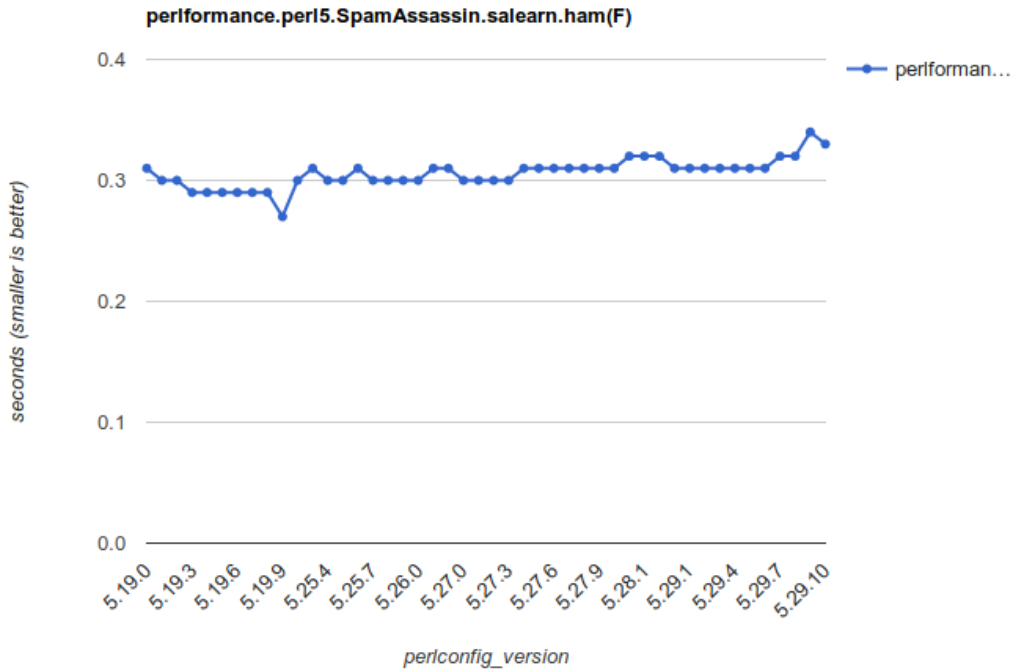
Perl::Formance - pts2019-f - regex.runtime_comp_nocache(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::PerlStone2015::regex



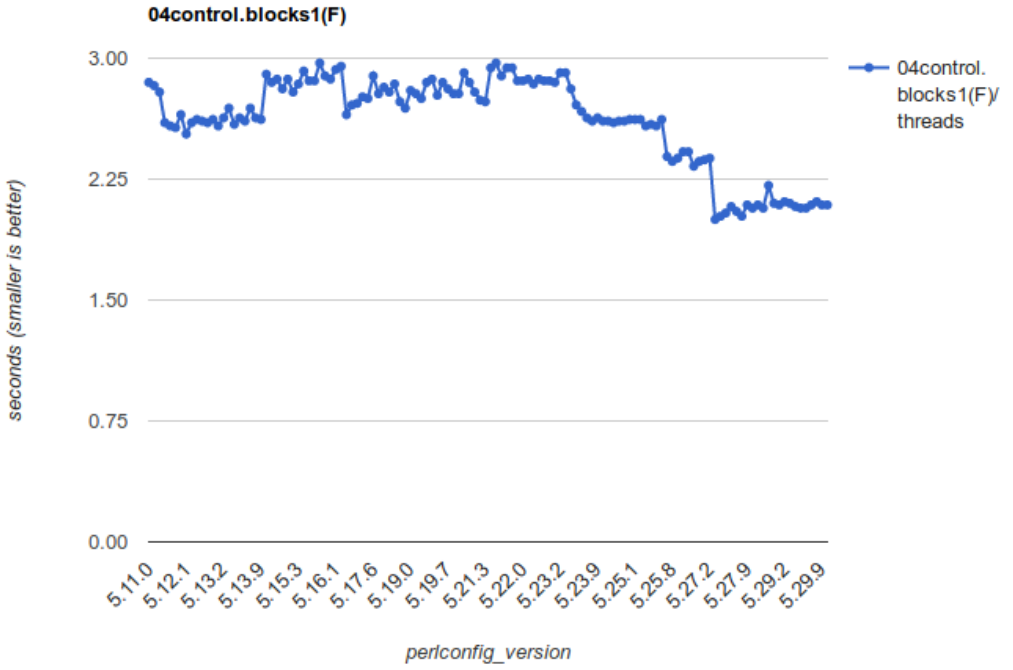
Perl::Formance - pts2019-f - perlformance.perl5.SpamAssassin.salearn.ham(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::SpamAssassin



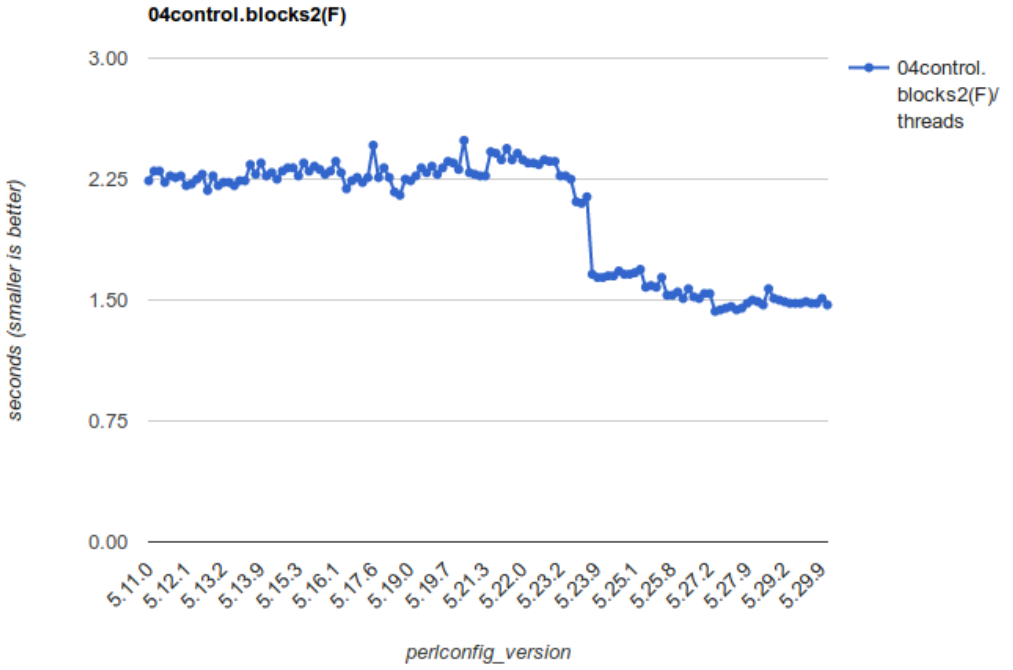
Perl::Formance - pts2019-f - 04control.blocks1(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::PerlStone2015::04control



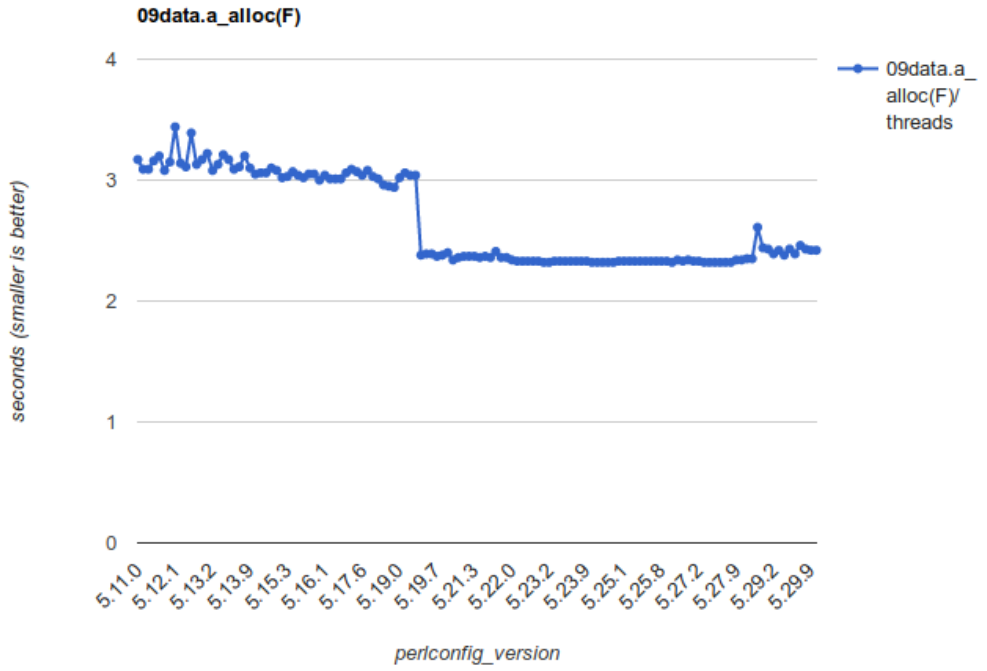
Perl::Formance - pts2019-f - 04control.blocks2(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::PerlStone2015::04control



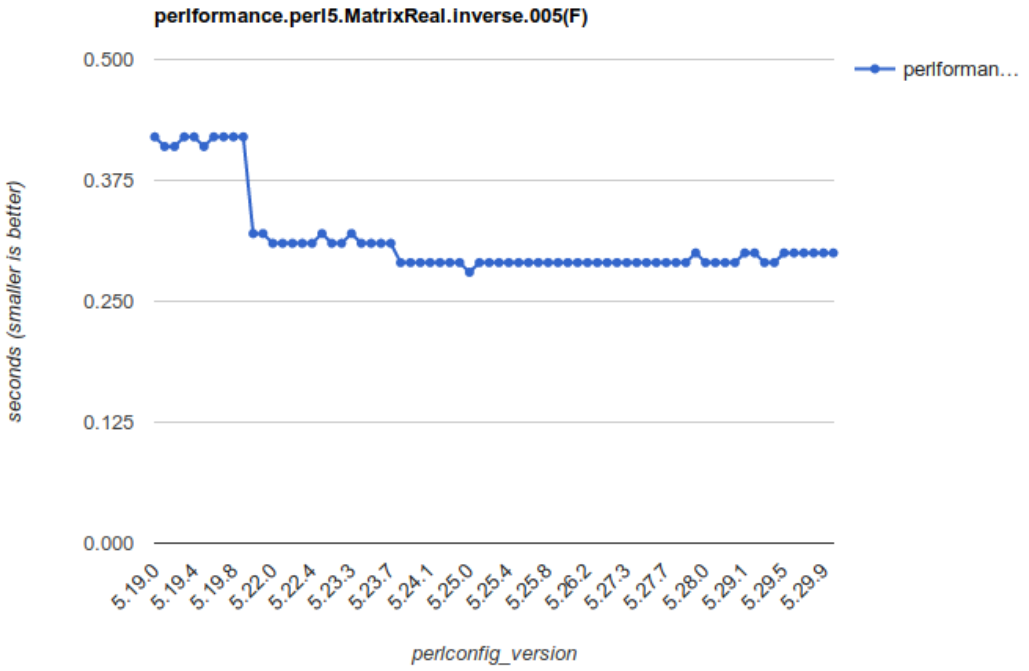
Perl::Formance - pts2019-f - 09data.a_alloc(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::PerlStone2015::09data



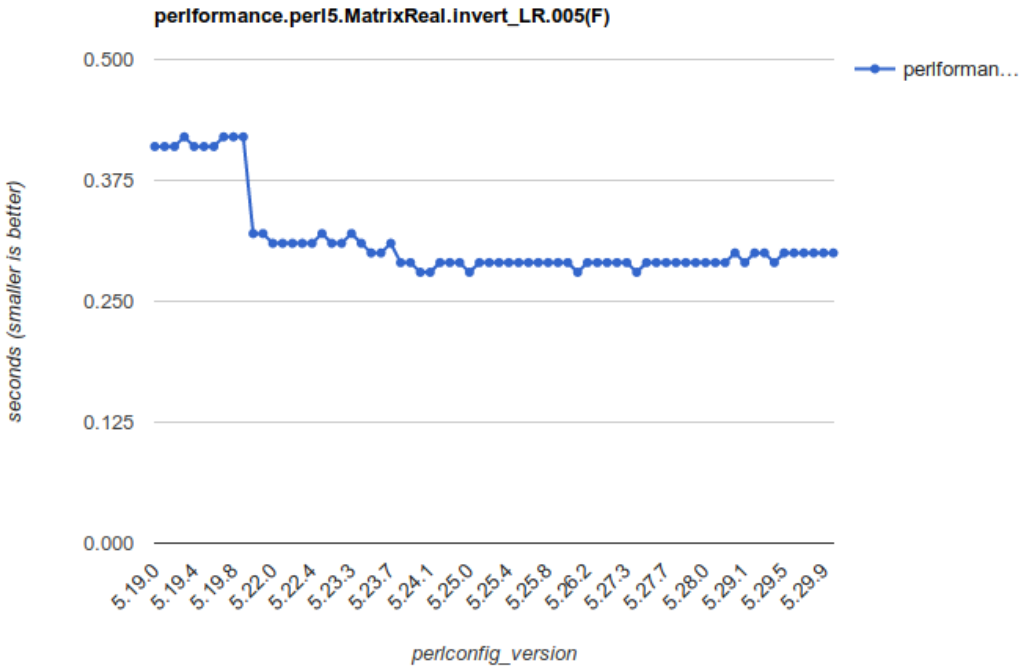
Perl::Formance - pts2019-f - perlformance.perl5.MatrixReal.inverse.005(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::MatrixReal



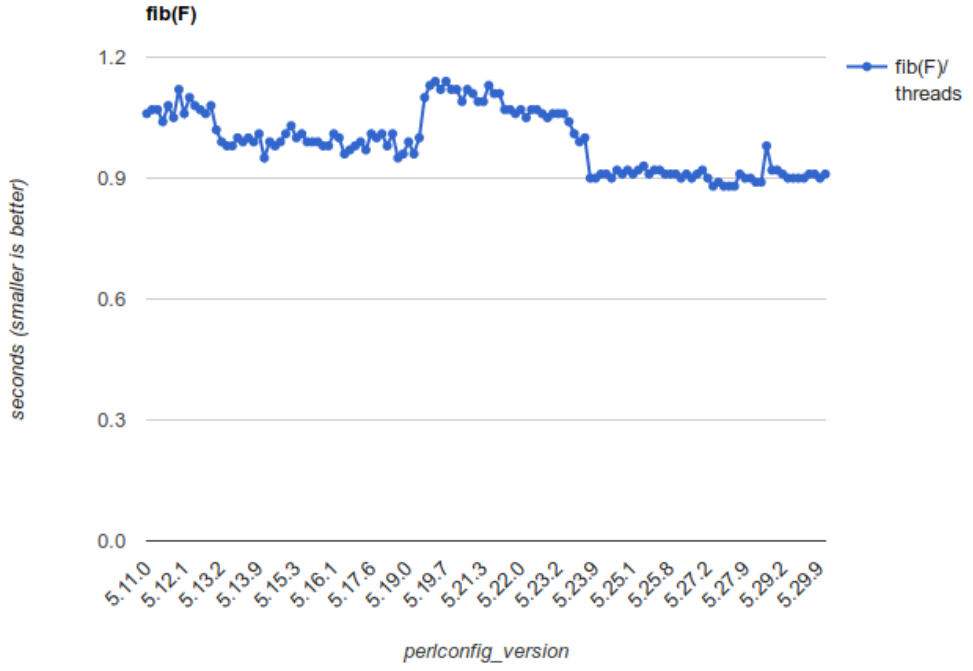
Perl::Formance - pts2019-f - perlformance.perl5.MatrixReal.invert_LR.005(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::MatrixReal



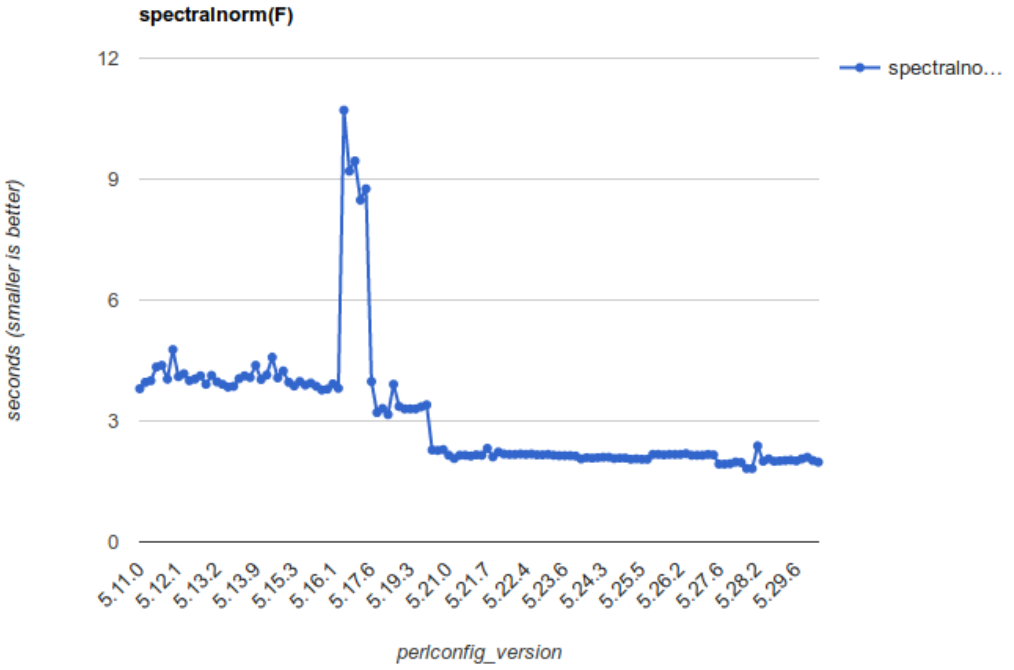
Perl::Formance - pts2019-f - fib(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::PerlStone2015::fib



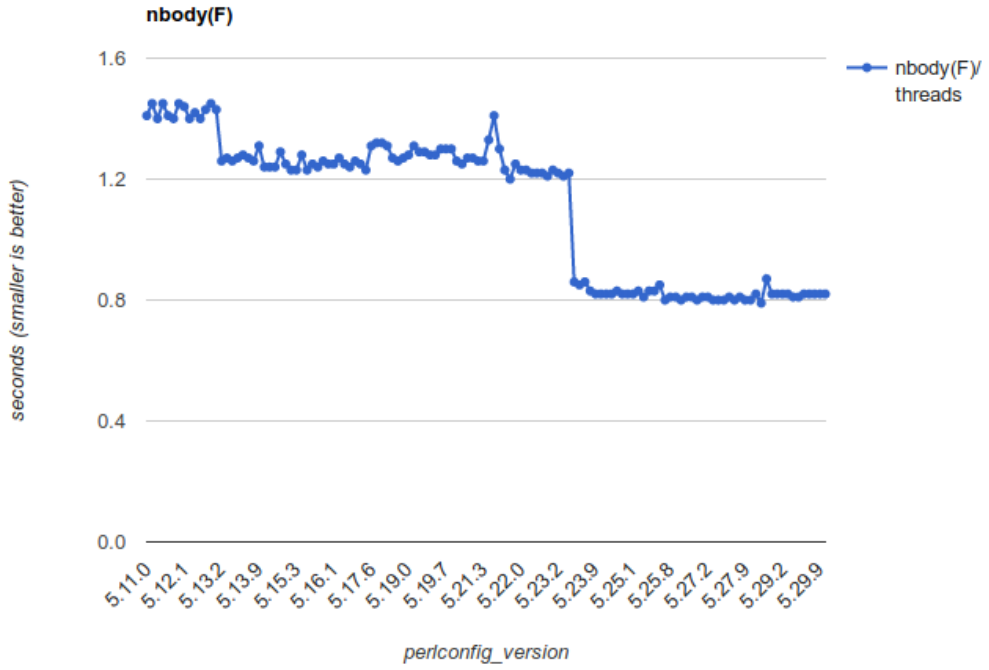
Perl::Formance - pts2019-f - spectralnorm(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::PerlStone2015::spectralnorm



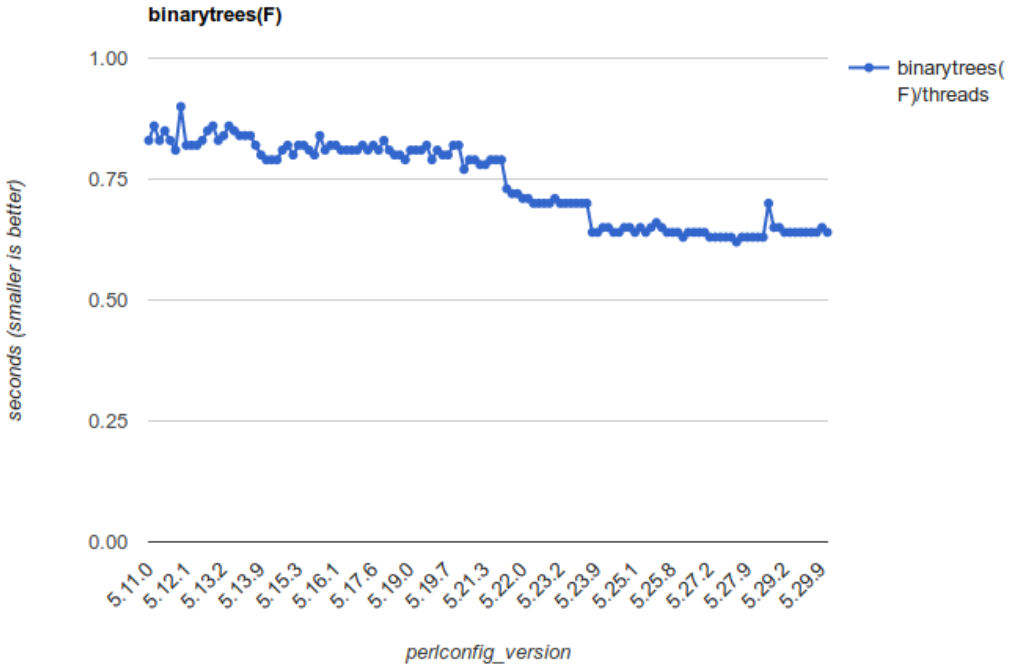
Perl::Formance - pts2019-f - nbody(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::PerlStone2015::nbody



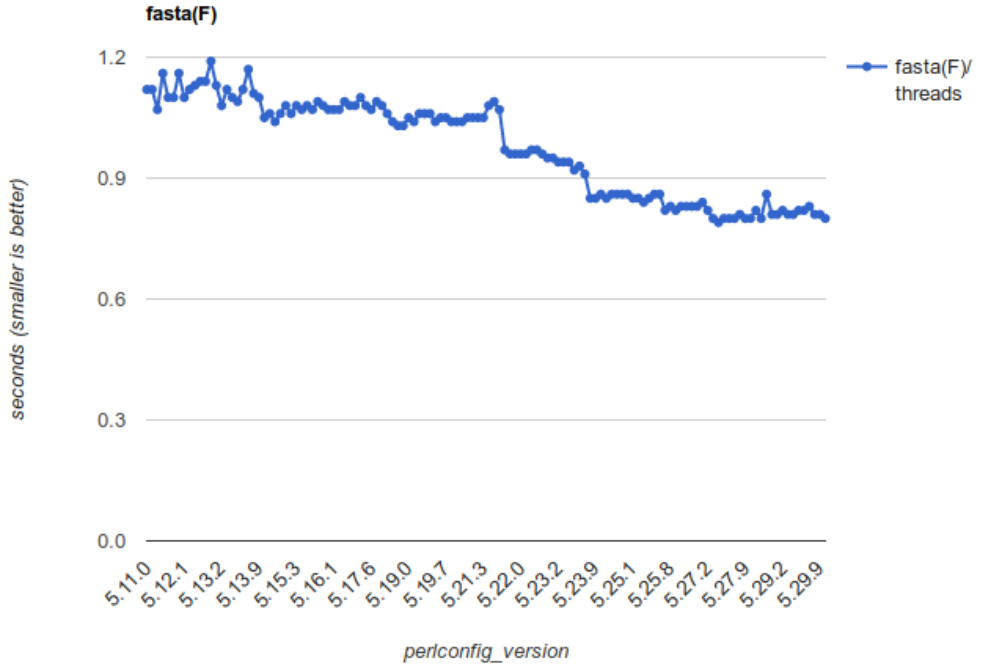
Perl::Formance - pts2019-f - binarytrees(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::PerlStone2015::binarytrees



Perl::Formance - pts2019-f - fasta(F)

- Back to dashboard
- Raw numbers: here
- Benchmark code: Benchmark::Perl::Formance::Plugin::PerlStone2015::fasta



Summary

Perl 5 once fast.

Then feature-rich but slower.

Now feature-rich and fast again.


DRESDEN PERL MONGERS
use style;