Ufuk Kayserilioglu @paracycle

Dec 21 • 28 tweets • paracycle/status/1605706226007941122

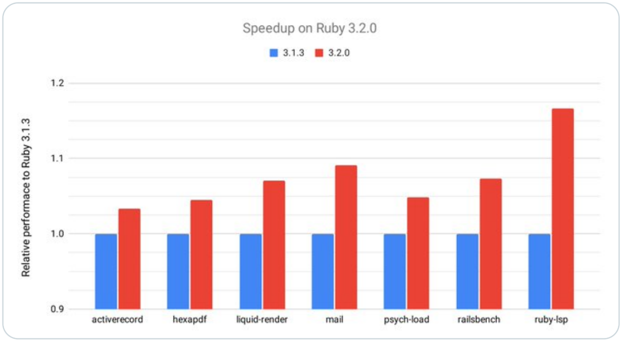
I'm so hyped as we get closer & closer to the Christmas Day release of Ruby 3.2. Actually, there are so many great things in 3.2 that we couldn't wait for release day, but already deployed it for all Shopify storefront requests.

Allow me to do a deep dive on all the details 🧵

tobi lutke
@tobi · **Follow**

All storefront requests are now served by the latest version of ruby with YJIT enabled! We are seeing ~10% speedups across the board.

YJIT that has been developed by @Love2Code's and team at Shopify.



8:10 PM · Dec 21, 2022

Read the full conversation on Twitter

363 Reply Copy link

Read 7 replies

The headline feature that we're all very excited about is YJIT. The YJIT version in Ruby 3.2 is now labeled as production-ready, since we've been able to solve the biggest production drawback, which was the large memory overhead of the previous YJIT release.

Now YJIT uses less memory overall (so much so that we reduced the default executable memory size config from 256MB to 64MB). It also lazily allocates the memory that it needs to use, so even if you have a smaller app, you won't have to tune your parameters any longer.

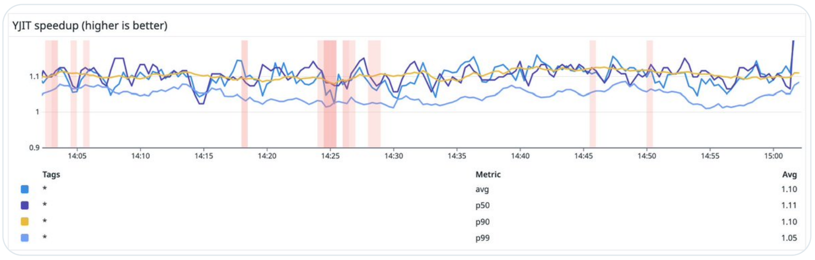
Moreover, there is now Code GC, which invalidates and garbage collects generated machine code that is no longer valid or needed. This allows applications that fill the full executable memory size to reclaim some memory so that they can continue to benefit from the JIT.

There has to be trade-off, right? Well, no! Despite using memory more efficiently, YJIT is now even faster than the previous YJIT, and is up to 40% faster than CRuby on railbench. 🎉 You can see more details at speed.yjit.org

On top of all this, the team rearchitected the JIT, so that the backend that generates machine code is separated from the frontend, allowing the team to add an ARM backend to complement the x64 backend that already existed. Now you can run YJIT on your RPis or Graviton servers.

The YJIT team set an ambitious roadmap at the start of the year, decided on a Rust rewrite not on the roadmap, grew the team & still delivered all the items by end of Nov. This's been an amazing effort from @Love2Code, @alanwusx, @codefolio , @kddnewton , @k0kubun & @jimmyhmiller

All that work allowed us to speedup our storefront total web request time by 10% on average, which is including all the time the web server is blocked on IO, for example, waiting for data from the DB, which YJIT obviously can't make any faster.



Tags	Metric	Avg
*	avg	1.10
*	p50	1.11
*	p90	1.10
*	p99	1.05

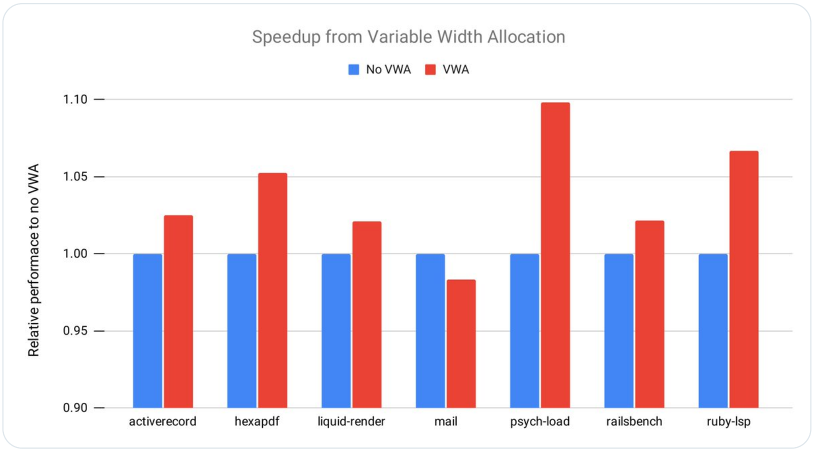
However, YJIT is not the only exciting thing in this release. Ruby 3.2 is also the first release that will come with Variable Width Allocation (VWA) enabled by default. VWA was a project we started 2 years ago to improve memory locality in the Ruby virtual machine.

Up to Ruby 3.1, the Ruby VM would always allocate Ruby objects using fixed 40 byte slots. It would use some of the 40 bytes for bookkeeping, and could store some data in the remaining 20 odd bytes. However, most objects store more data than that.

So, Ruby would have to allocate extra memory from the system to store the extra data there. That extra memory location would be far from the location of the object slot, which meant that extra memory reads would need to be done to read objects, which is inefficient.

VWA project implemented multiple sized slots to allow bigger objects to store their data right in the slot. This improves memory locality & allows object access to be faster. Initial work for VWA was done in Ruby 3.1 but behind a compile flag; with 3.2, it becomes the default.

And what's the upshot of all of this? Glad you asked! Compared to no VWA, the default Ruby 3.2 performance with VWA on realistic benchmarks is between 2% to 10% better.



This work was done by [@peterzhu2118](), [@eightbitraptor]() and [@tenderlove]() and if you want to learn more, you should definitely watch this RubyKaigi 2021 talk by Peter and Matt:

What problems are we trying to solve?
RubyKaigi Takeout 2021 #rubykaigi
sponsored by

https://www.youtube.com/embed/7C3bdT6Ri2Q

This brings us to the final piece of the puzzle to speed up Ruby, which is Object Shapes. Object Shapes is a technique to efficiently store properties of objects. The technique goes all the way back to Smalltalk and is currently being used by V8, TruffleRuby and other VMs.

The idea is to treat an object as a bag of properties, and to consider objects that have defined the same properties in the same way to have the same "shape". What I mean by properties is "state" that an object holds, like instance variables or even the frozen status.

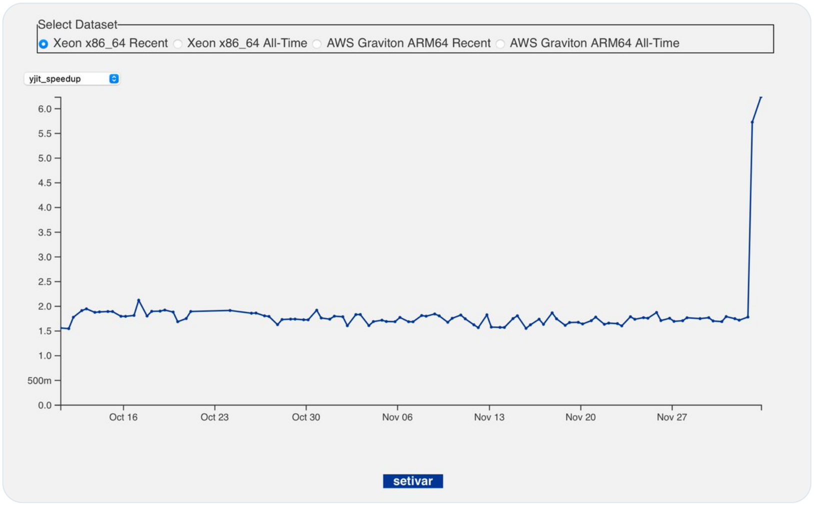
By representing objects using such shapes, it is possible to implement optimizations that are not available otherwise. [@ChrisGSeaton]() had given an excellent RubyKaigi 2021 talk explaining the idea: https://chrisseaton.com/truffleruby/rubykaigi21/

Working on YJIT, @Love2Code realized that if Ruby were to implement this Object Shapes technique, we could improve instance variable access for both the interpreter and the JIT. She explained some of that idea in her RubyKaigi 2021 talk:

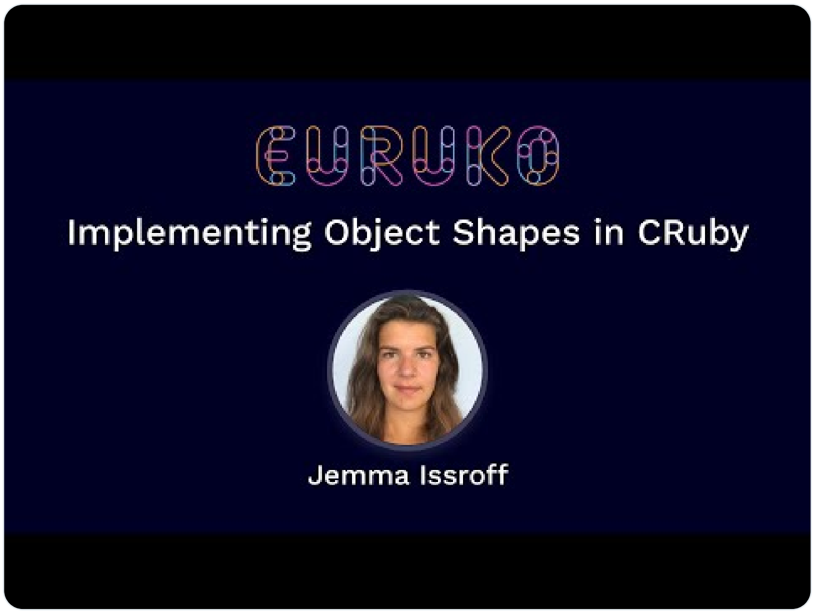
https://www.youtube.com/embed/PBVLf3yfMs8?start=1164

(timestamped link)

Motivated by this direction, we setup a small team with @JemmaIssroff and @tenderlove to work on implementing Object Shapes earlier this year. The result have been stunning. The final state of the work is able to speed up instance variable writes by 6x with YJIT! 🚀



In order to understand how this is implemented, or how it leads to performance gains, I recommend you watch the excellent talk by Jemma from Euruko 2022:



https://www.youtube.com/embed/2aVyTtxs0GU

We still have more ideas to implement on top of the Object Shapes work in Ruby 3.2, next year.

Phew, this is already a long thread, but the performance gains are not the only things that excite me about this upcoming release.

So, what more is there? Stability!

We'd always been ambitious at moving to new versions of Ruby at Shopify, but it hadn't been easy over the years.

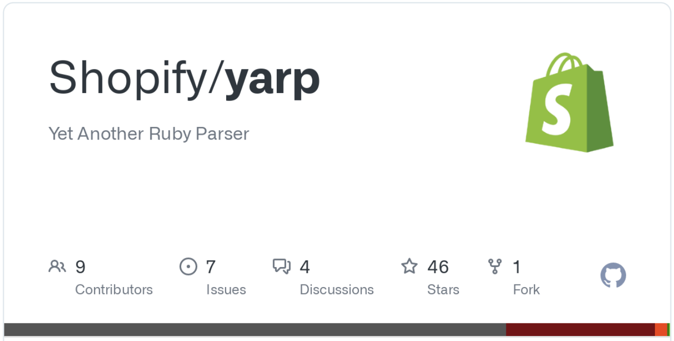
Last year @_byroot lead an effort to use our large platform for the benefit of the stability of Ruby 3.1 and it paid off. We were able to run Ruby 3.1.0 on our Core monolith merely 18 days after release, and actually 3 days after we were back from the holidays.

This year, we doubled down on this approach. Since the beginning of the year, we've been running our Core monolith CI against the nightly checkout of Ruby HEAD. If there were any crashes we would either investigate them or report them upstream.

So, throughout this year we've been investing into the stability of Ruby. Starting early September 2022, we increased our efforts even more by running a Ruby HEAD checkout every hour on our Core monolith CI. We also setup an early access ARM cluster to test the new YJIT backend.

All this work identified some obscure bugs that we fixed before release so that we can run Ruby 3.2 days after release again. This time, though, we ended up shipping 3.2 to our storefronts before the release, with massive gains to our performance and for the merchants we serve.

Of course, we are not done yet and we have ambitious plans for future releases as well. @kddnewton has just open-sourced the initial version of the new Ruby parser that he's been working on for the past 4 months:

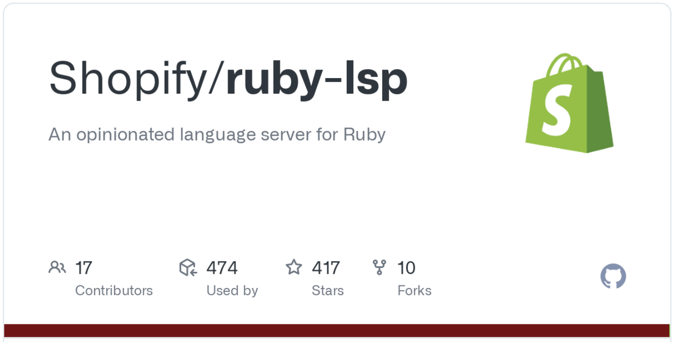

Shopify/**yarp**

Yet Another Ruby Parser

9 Contributors · 7 Issues · 4 Discussions · 46 Stars · 1 Fork

GitHub - Shopify/yarp: Yet Another Ruby Parser
Yet Another Ruby Parser. Contribute to Shopify/yarp development by creating an account on GitHub.
https://github.com/Shopify/yarp

This new parser will take some time to reach completion, but we are expecting to reap early benefits from it by employing it in our developer tools like the Ruby LSP server that our Ruby DX team has been working on:


Shopify/**ruby-lsp**

An opinionated language server for Ruby

17 Contributors · 474 Used by · 417 Stars · 10 Forks

GitHub - Shopify/ruby-lsp: An opinionated language server for Ruby
An opinionated language server for Ruby. Contribute to Shopify/ruby-lsp development by creating an account on GitHub.
https://github.com/Shopify/ruby-lsp

Of course, there will be more improvements to YJIT, GC and Object Shapes as well. As we clarify the roadmaps of all the teams, we will share more details regarding all the cool things our Ruby Infrastructure team at Shopify will be tackling in the new year. Stay tuned! 📻

• • •