Custom Types II
Garbage collection

Ryan Eberhardt and Julio Ballista
April 20, 2021

Logistics

- Week 3 exercises due Thursday
  - Please let us know if you get stuck / feel confused! We want you to sleep!
- Using myth? See announcements channel
- Participation incentive: At the end of the quarter, I'll randomly select at least 3 people that participated 10 times throughout the quarter, and I'll make you a custom mug or pot (see @pottedpeasceramics)
  - Asking or answering a question in lecture (out loud, or in the chat) or on Slack all count as participation

Today

- Wrapping up our basic linked list implementation
- Contrasting Rust with other memory-safe languages

Linked list live coding

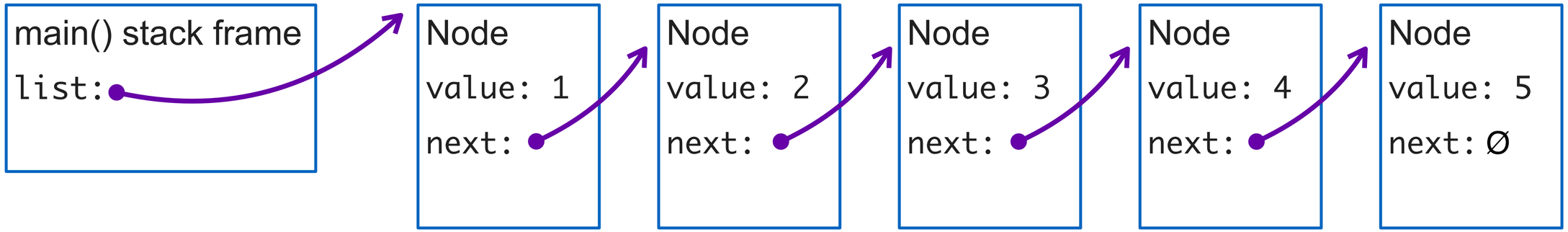
But wait… Why is this *so* much harder than it is in Python?

Garbage collection

Garbage collection

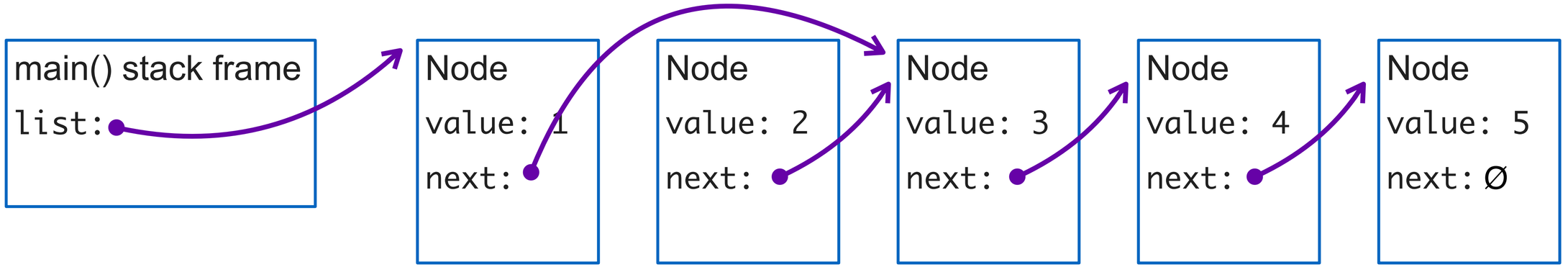
- C/C++ have a problem: When should you free your memory?
  - In complicated codebases, it's very easy to have memory leaks, double frees, or use-after-frees
- Rust: Use a fancy type system to denote who is responsible for freeing memory, and let the compiler check that everything looks right
  - Still difficult to program. You're constantly thinking about who has ownership of what
- Much older approach: Garbage collection
  - When writing your program, don't worry about freeing memory
  - When running your program, the runtime will observe when memory is no longer being used, and will free it for you

Tracing garbage collection



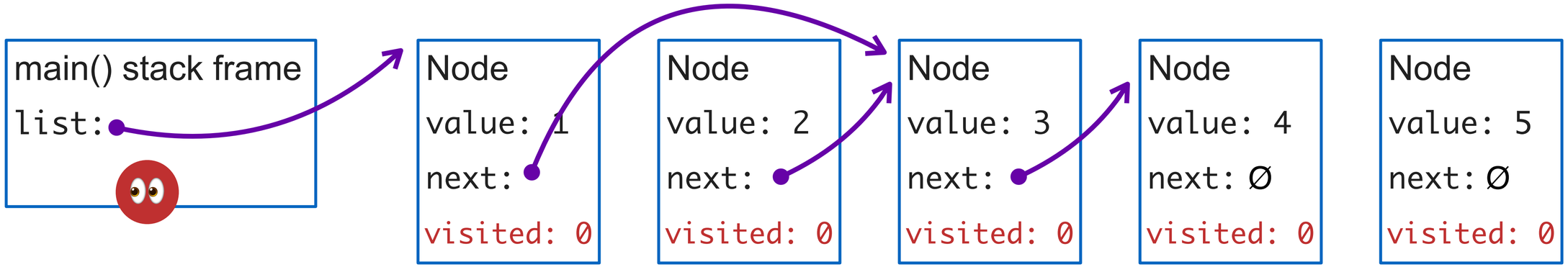
Remove 2nd node

Tracing garbage collection



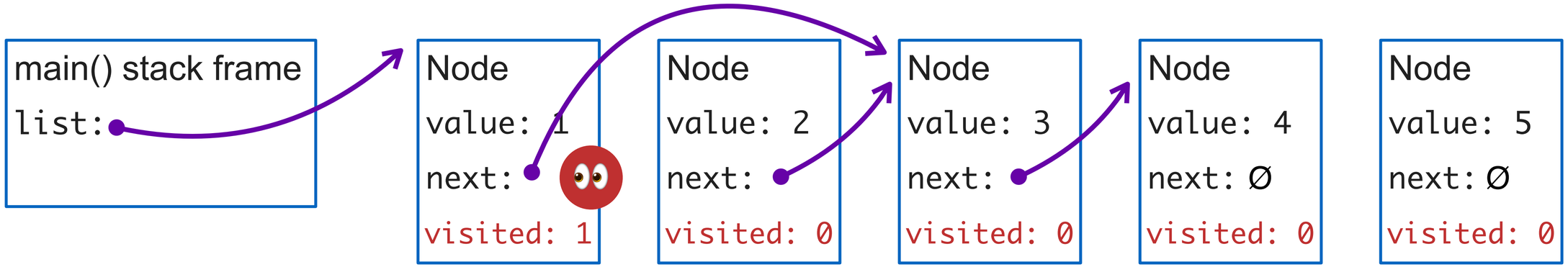
Remove last node

Tracing garbage collection



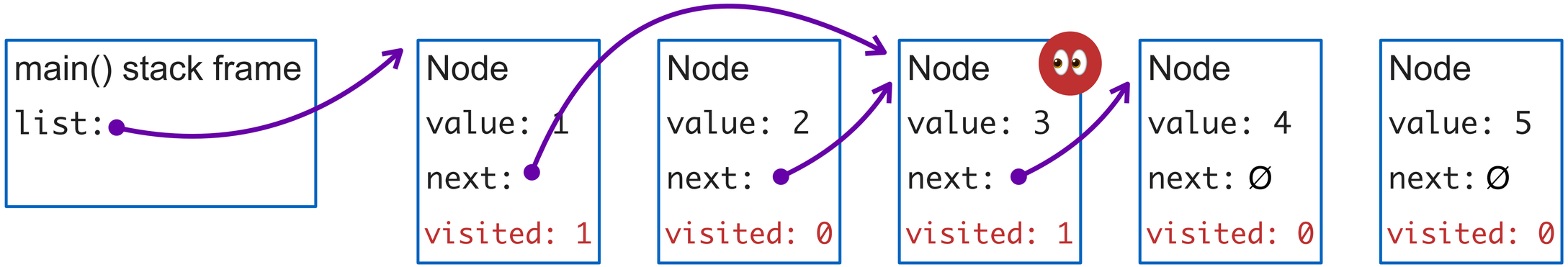
!! Pause execution, begin GC

Tracing garbage collection



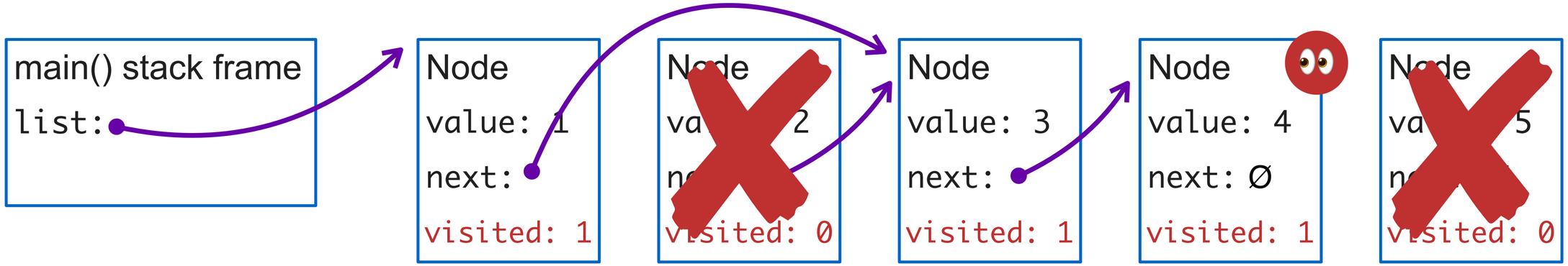
!! Pause execution, begin GC

Tracing garbage collection



!! Pause execution, begin GC

Tracing garbage collection



!! Pause execution, begin GC

Dear X,

I am looking forward to meeting you, and to a great year in Kimball!

Please consider an idea that I think will make life just a tiny bit better for everyone in the dorm this year.

Last year, a number of us noticed that some people in the dorm were pretty messy, their rooms were a mess, and trash piled up.

It turns out that not only are these trash piles unpleasant, but they can be a hazard, potentially even to others.

According to *The Cardinal Safety Letter* (2012) :

“We have seen some fairly impressive mountains of trash overflowing the little dorm room trash cans. This is not sanitary in the least!”

We also know that exhortations to clean up too often fall on deaf ears.

… more pleas follow …

The good news is that we have a completely painless solution that will be totally inclusive, promote a clean dorm, reduce stress, and engages with Stanford's goal of sustainability.

It's all set to go, just pending your go-ahead.

I will collect the trash from each room in Kimball every week* to help everyone maintain a clean living environment. For less than $.50 per student per weekday, we'll take out everyone's trash for all 10 weeks of the quarter.* By using dorm funds, it doesn't really cost anyone anything, yet we all benefit.

It's a great use of dorm funds, because it'll benefit every member of the dorm equally, which is exactly what dorm funds are for.

It's free for the residents: all we have to do is tie our bags of trash, place them outside our doors by midnight on Sunday, and I'll pick them up on Monday – providing a clean start to the week. Students will be saved the hassle and unpleasantness of completing this tiresome chore, and none of us will have to put up with the messy consequences of piles of trash in dorm rooms.

For just $25 per student, the entire dorm's trash is taken care of for the entire quarter.

A twist

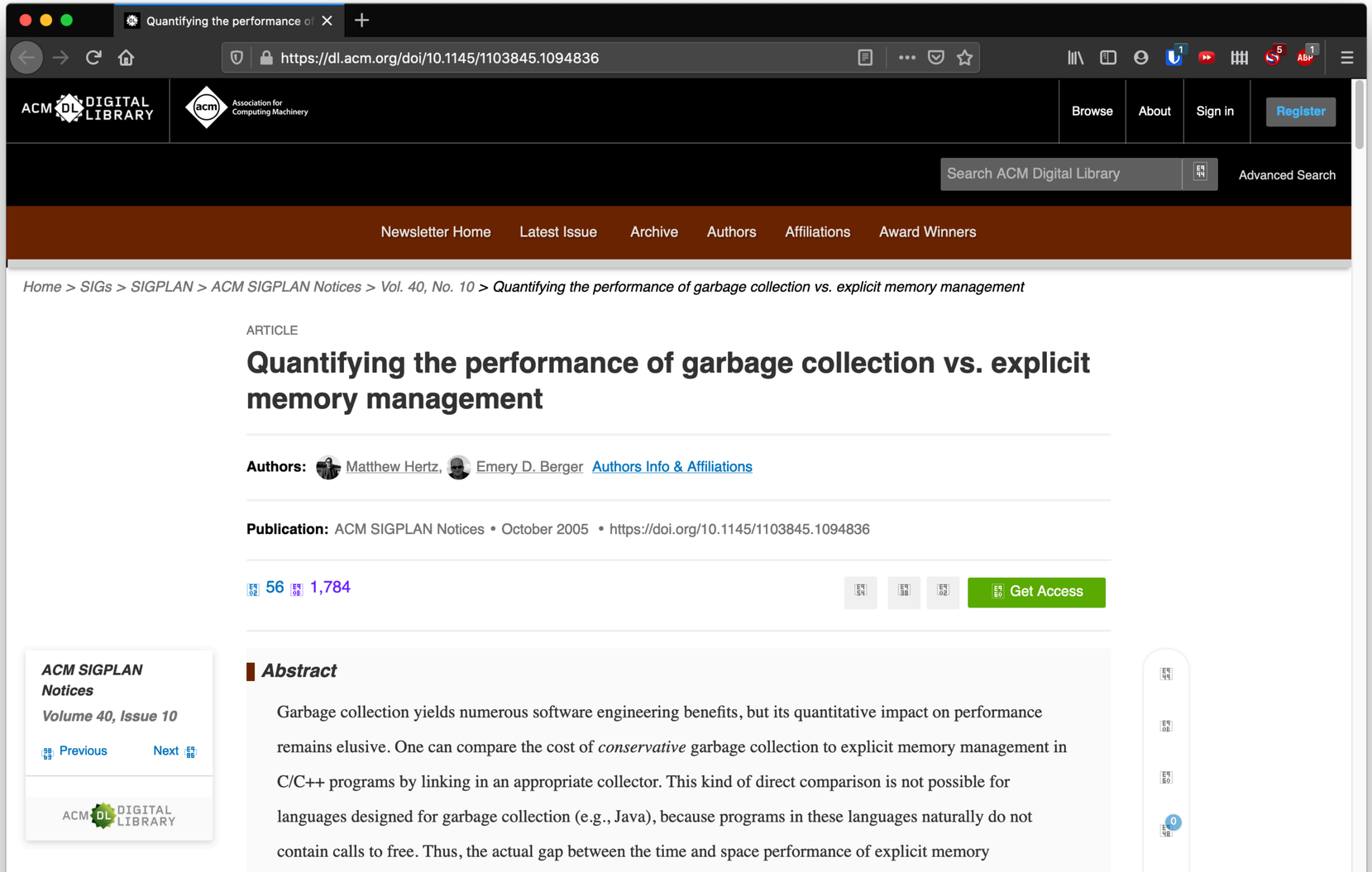
- Instead of putting your trash outside, leave it inside your room
- The GC will come knocking when it’s time to clean up

Downsides of garbage collection

- Expensive
  - No matter what type of garbage collection is used, there will always be nontrivial memory overhead
- Disruptive
  - Drop what you're doing — it's time for GC!
- Non-deterministic
  - When will the next GC pause be? Who knows! Depends on how much memory is being used
- Precludes manual optimization
  - In some situations, you may want to structure your data in memory in a specific way in order to achieve high cache performance
  - GC can't know how you will use memory, so it optimizes for the average use case

GC is expensive

https://dl.acm.org/doi/10.1145/1103845.1094836



GC is expensive

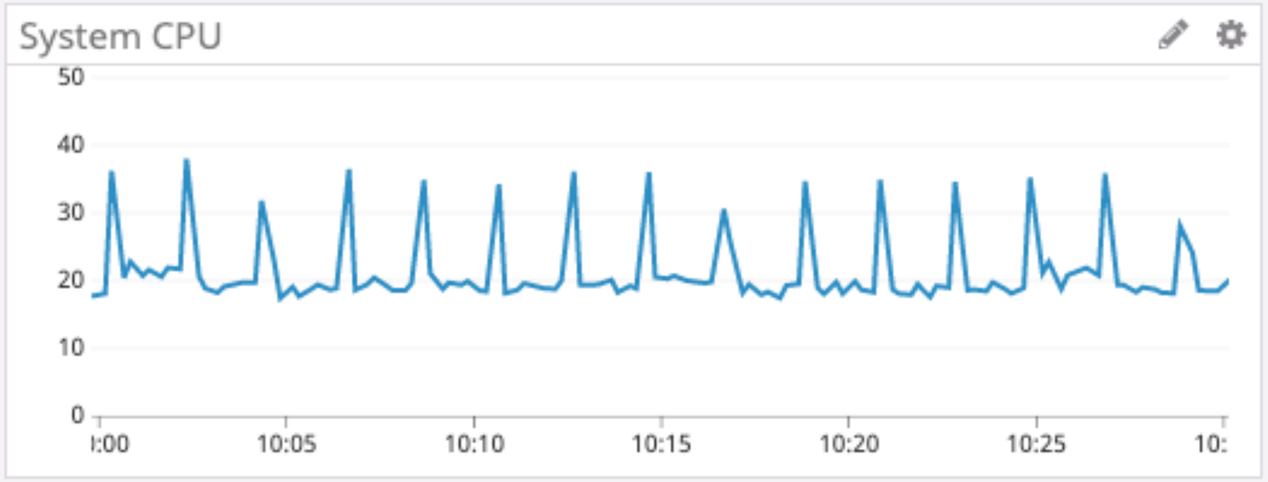
https://dl.acm.org/doi/10.1145/1103845.1094836

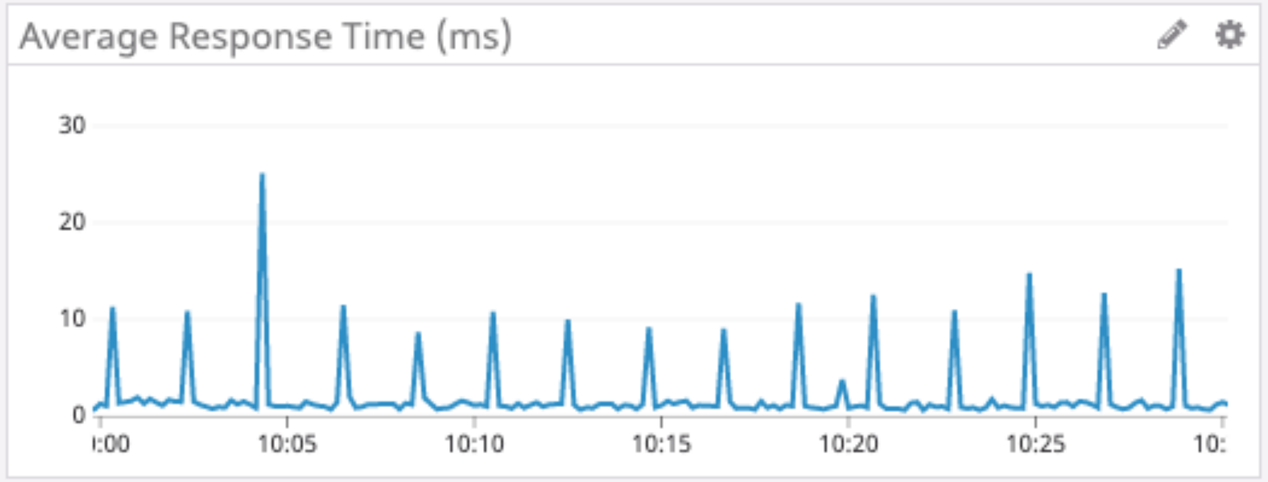
> *With five times as much memory, an Appel-style generational collector with a non-copying mature space matches the performance of reachability-based explicit memory management. With only three times as much memory, the collector runs on average 17% slower than explicit memory management. However, with only twice as much memory, garbage collection degrades performance by nearly 70%. When physical memory is scarce, paging causes garbage collection to run an order of magnitude slower than explicit memory management.*

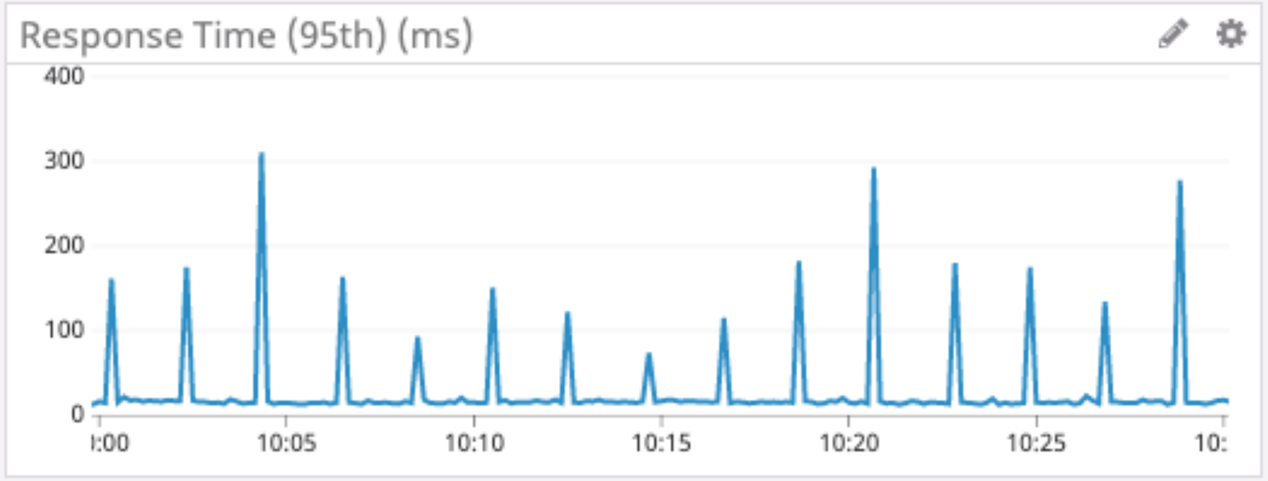
"Quantifying the performance of garbage collection vs. explicit memory management," Hertz and Berger

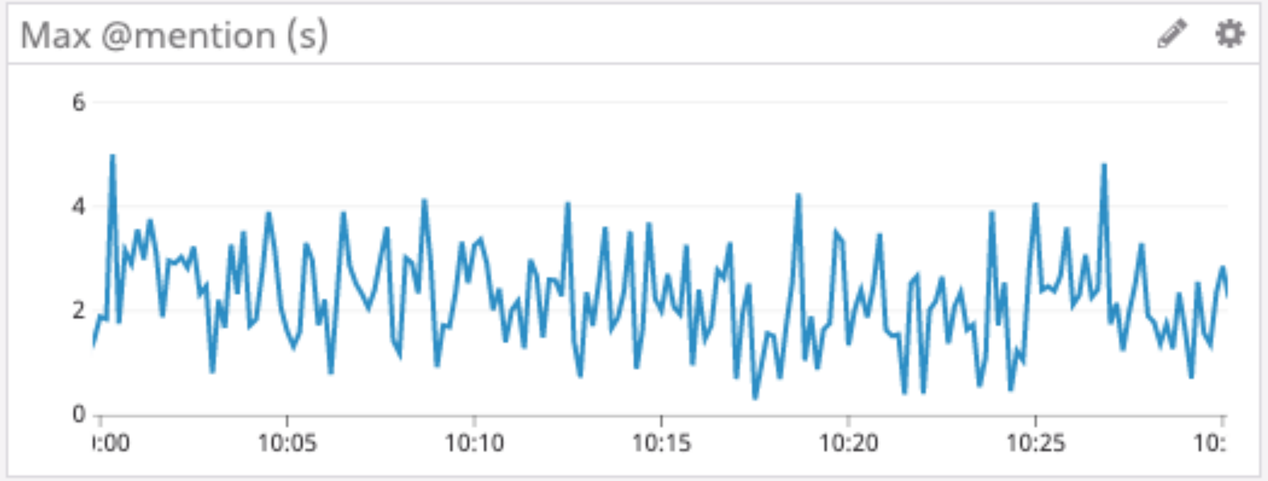
Why Discord is switching from Go to Rust

Jesse Howarth Follow
Feb 4 · 10 min read









Note latency spikes every 2 minutes

LinkedIn Engineering:

“In our production environments, we have seen unexplainable large STW pauses (> 5 seconds) in our mission-critical Java applications.”

https://engineering.linkedin.com/blog/2016/02/eliminating-large-jvm-gc-pauses-caused-by-background-io-traffic

Latency matters

- User interfaces
- Games
- Self-driving cars
- Payment processing
- High frequency trading

Takeaways

- Use GC languages when it makes sense, but know their limits
  - It doesn’t matter how much memory you save if it takes you so long to develop your app that no one uses it
  - You can always rewrite certain components in other languages if efficiency becomes a problem
- In resource-constrained or latency-sensitive environments, GC may not be a viable option

Rust is about more than just memory safety

- Garbage collection solves memory safety, but does not address other types of errors
  - Might not have memory leaks, but can still have file descriptor leaks, database handle leaks, etc.
  - Can still have memory errors caused by race conditions in multithreaded programs
- Recall that Rust's fancy type system is designed to help us communicate our expectations (pre/postconditions, etc) so that the compiler can sanity check us
- This helps with all sorts of different safety and correctness issues!