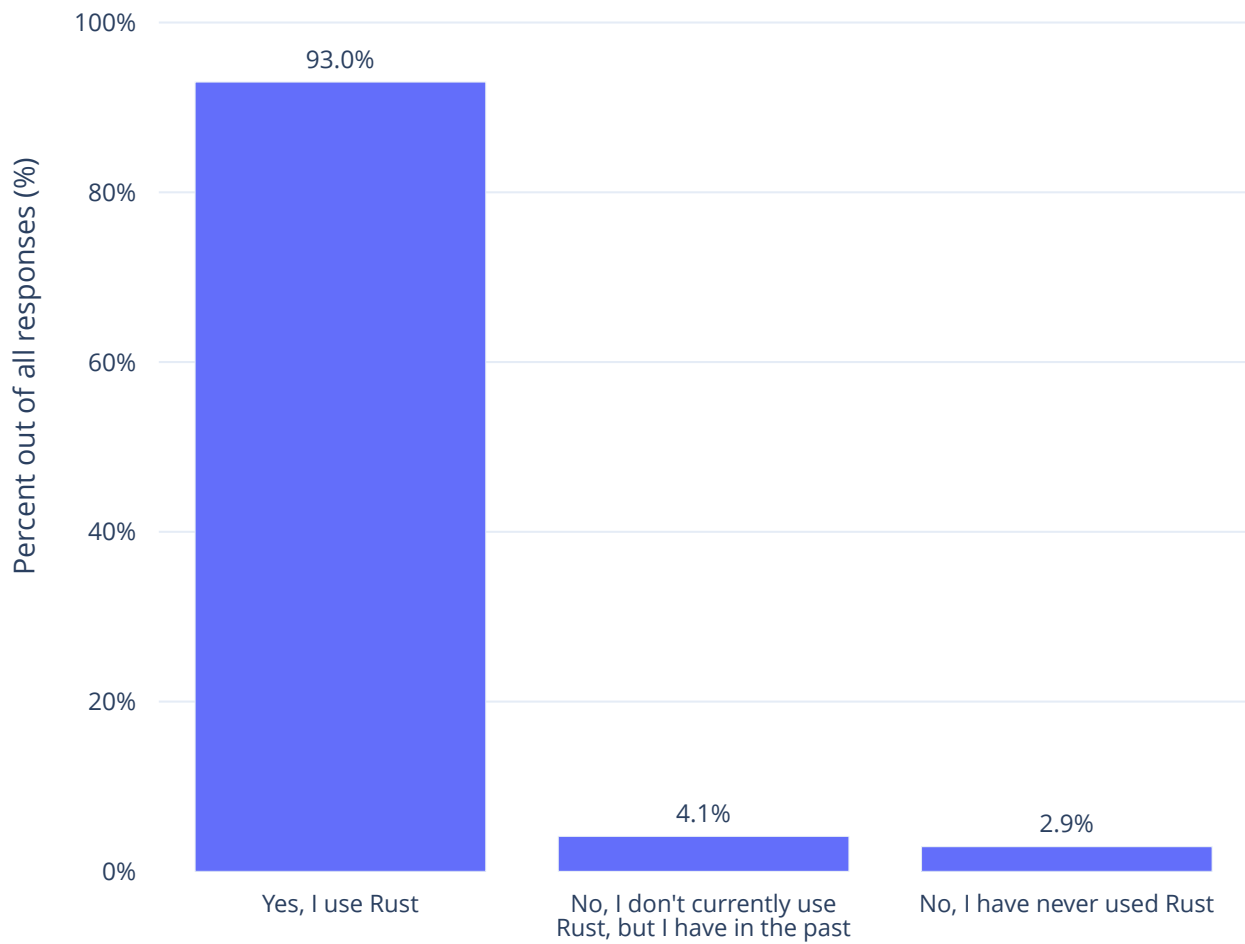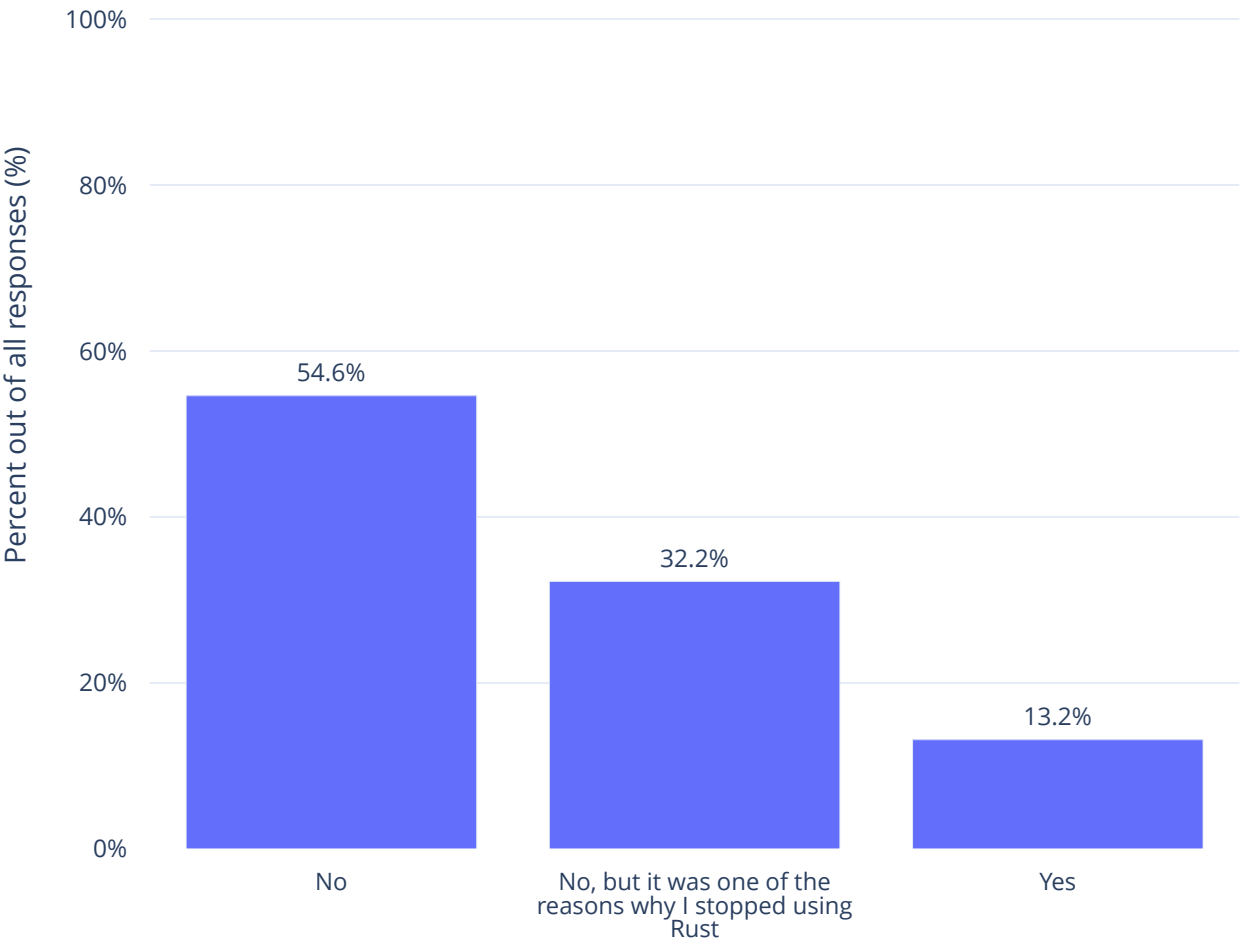# Compiler performance survey 2025 report

Do you use Rust?

(total responses = 3720, single answer)
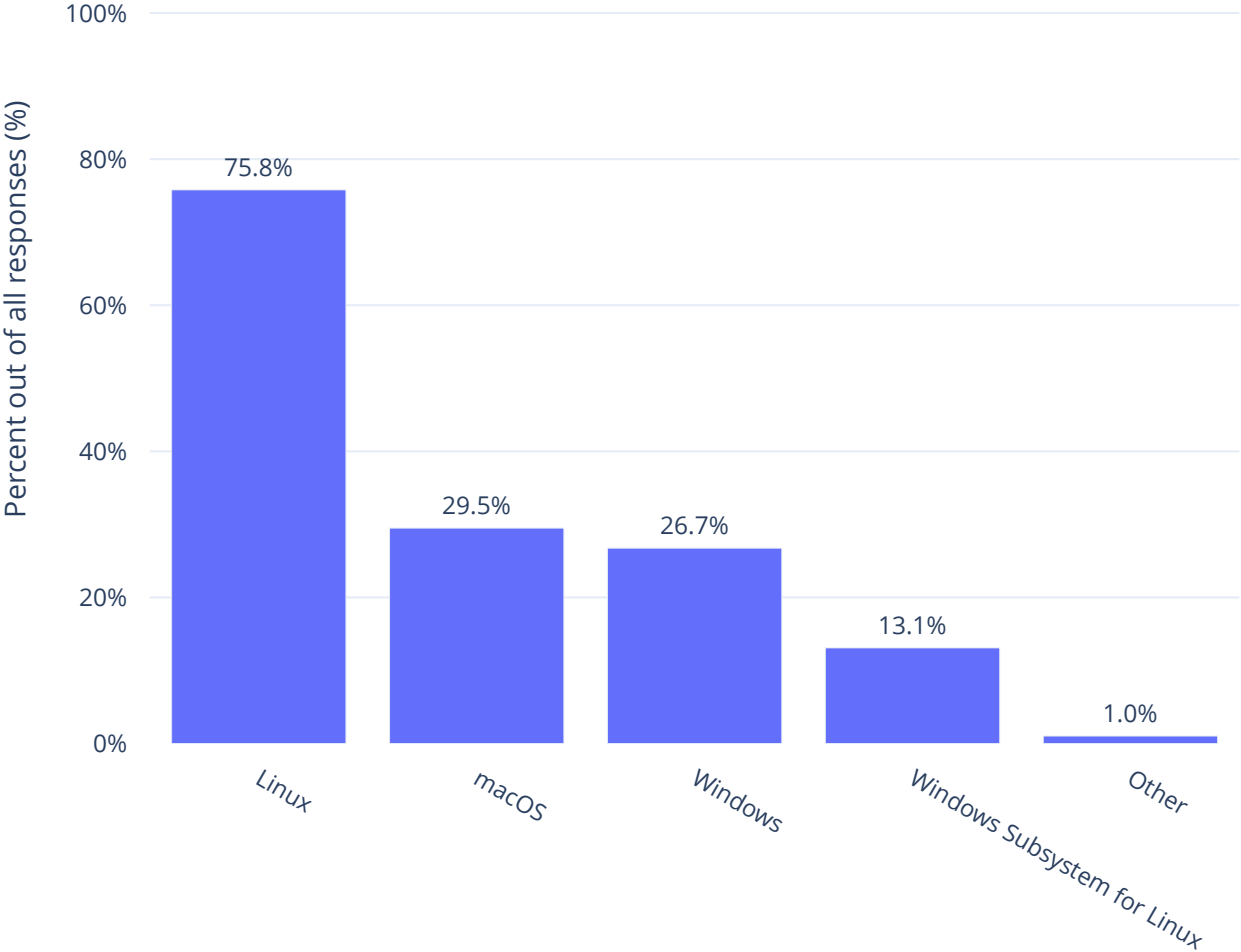
Were long compilation times the primary reason why you stopped using Rust?

(total responses = 152, single answer)

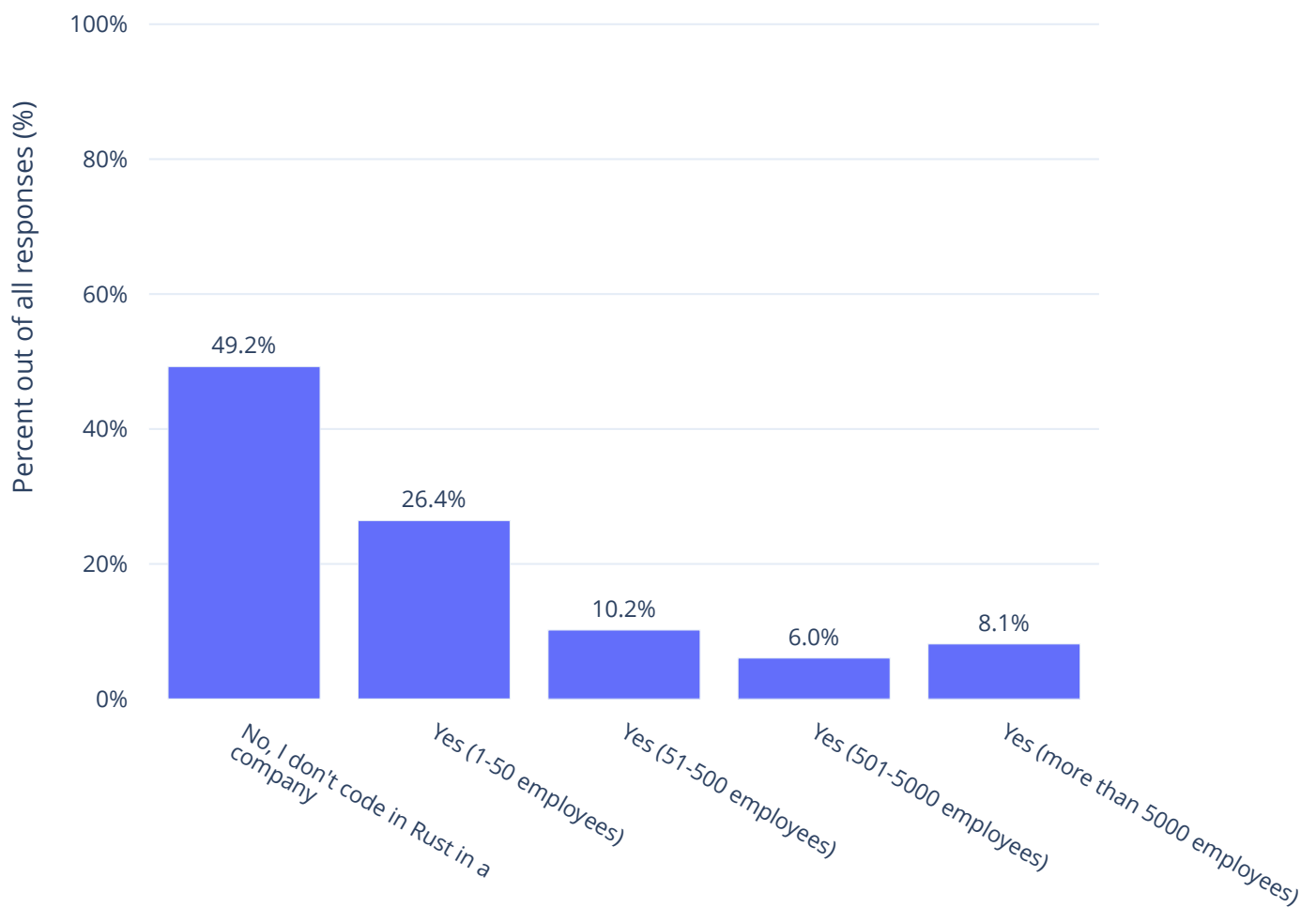## Which operating systems do you use regularly for Rust development?

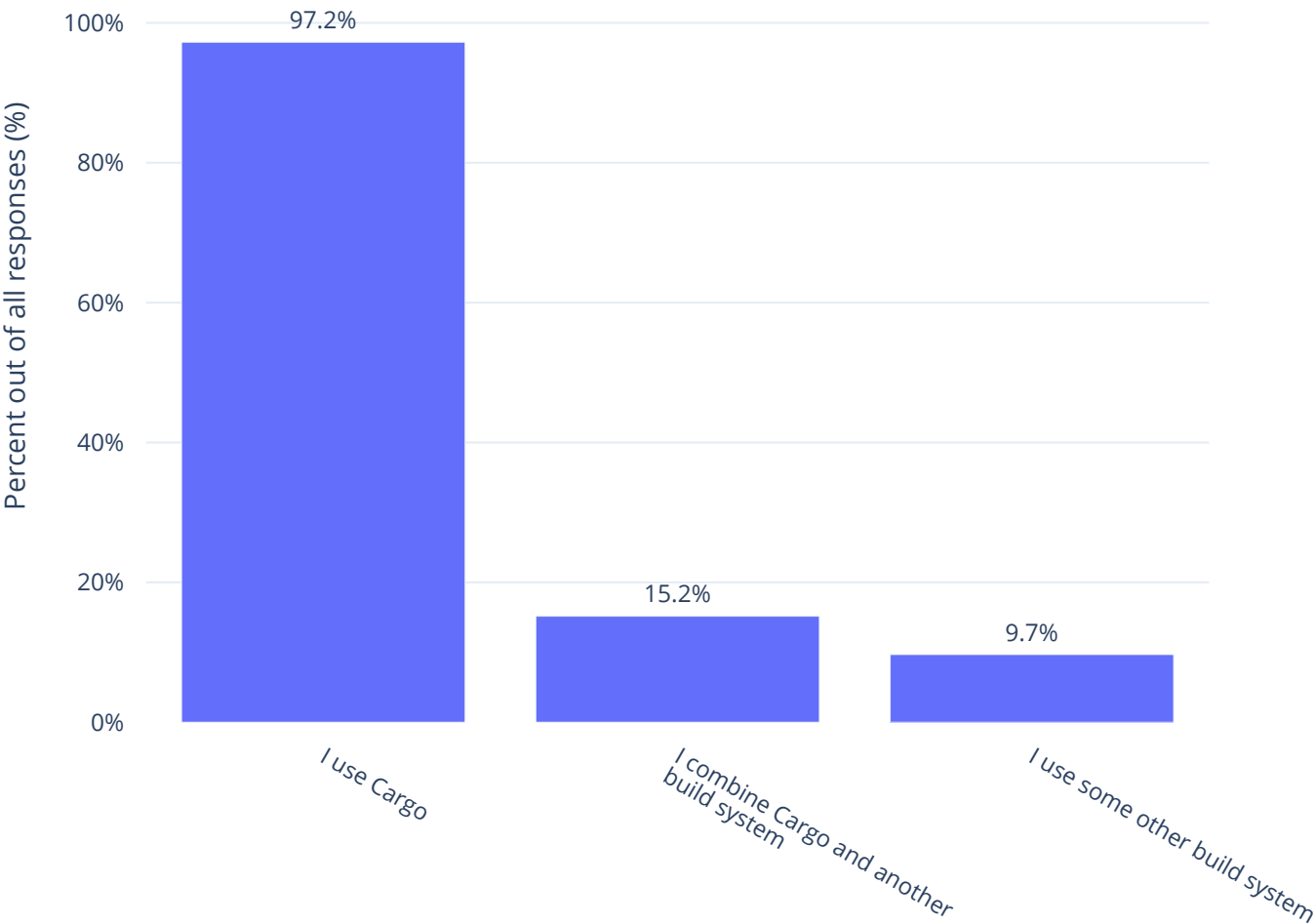(total responses = 3496, multiple answers)

Do you code in Rust at a company?
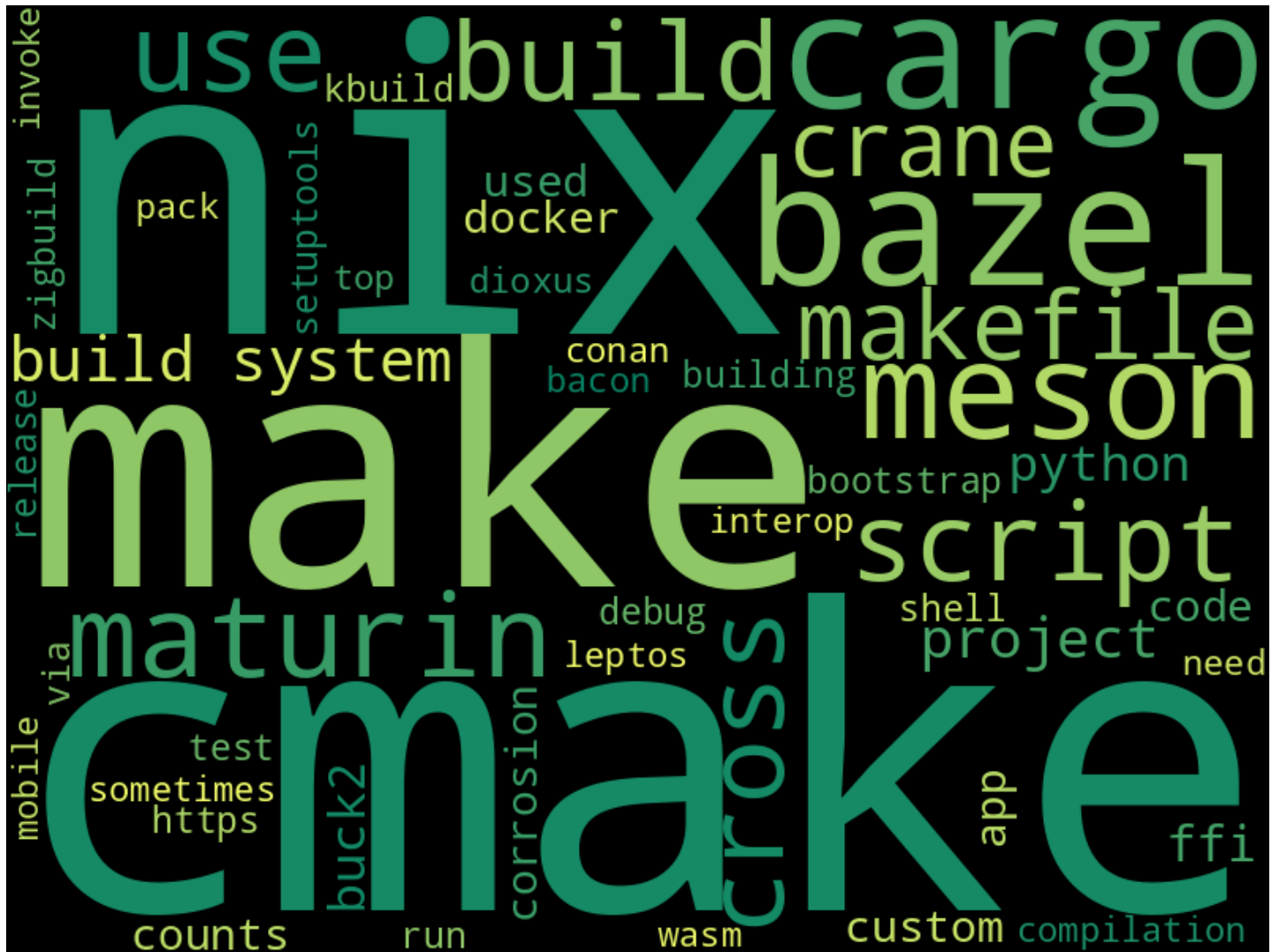
(total responses = 3492, single answer)

Which build systems do you use to build Rust code?
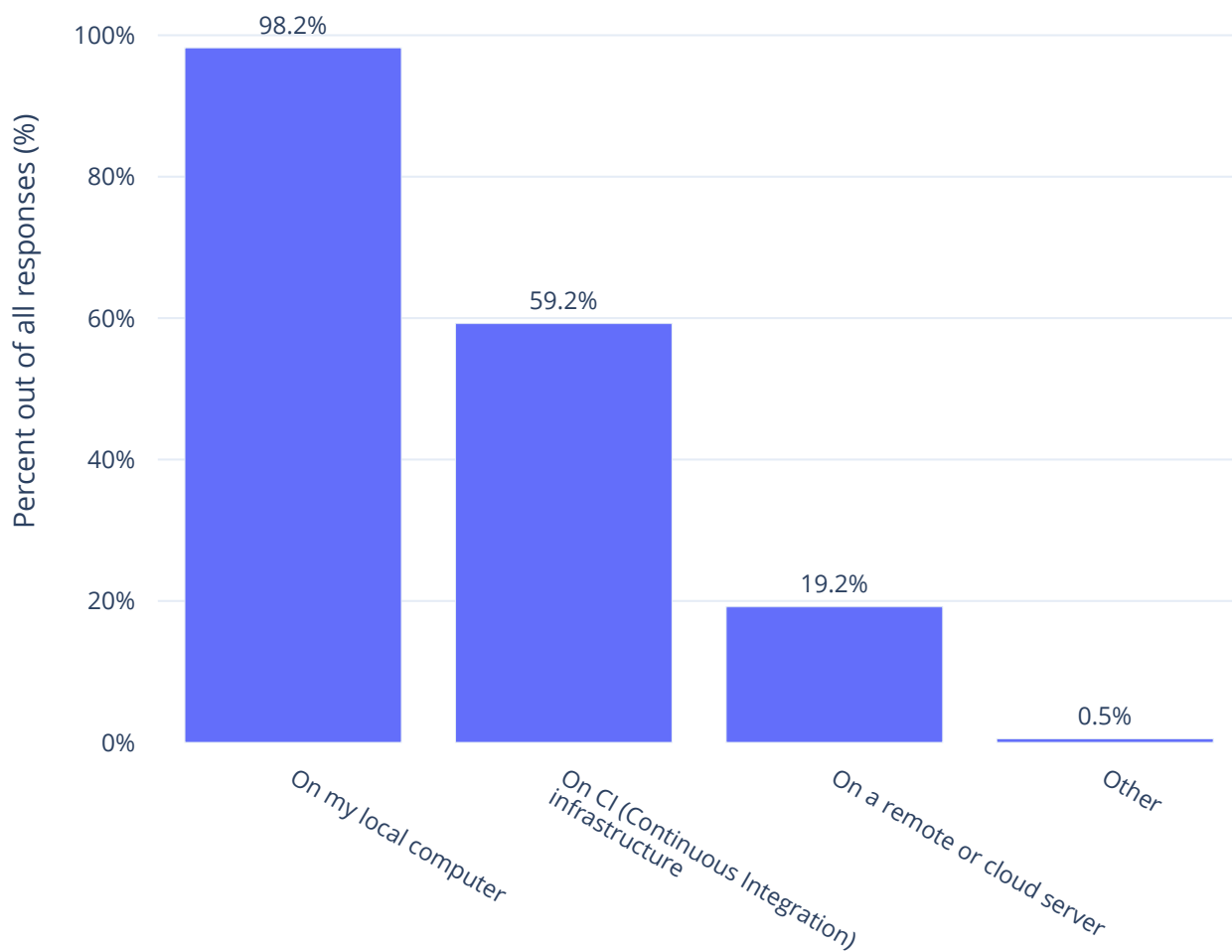
(total responses = 2521, multiple answers)

Wordcloud of open answers for the previous chart:

How do you build your Rust projects?

(total responses = 2524, multiple answers)

Which development workflows limit your productivity?

(total responses = 2490)

Workflow	Big problem for me	Could be improved, but does not limit me	Not an issue for me at all	I don't use this workflow
Unoptimized rebuilds	24.76%	46.43%	25.49%	3.33%
Workspace rebuilds	30.93%	38.29%	19.19%	11.59%
Optimized rebuilds	20.60%	44.41%	27.43%	7.56%
Clean optimized builds	21.66%	40.27%	30.25%	7.82%
Clean unoptimized builds	18.84%	44.65%	28.32%	8.18%
cargo check	14.39%	44.06%	38.55%	3.00%
cargo test	10.99%	39.47%	40.41%	9.13%
CI builds	17.27%	32.48%	24.09%	26.16%
Docker builds	10.95%	19.32%	17.69%	52.04%

Severity

- Big problem for me
- Could be improved, but does not limit me
- Not an issue for me at all
- I don't use this workflow

Do you use any caching to speed up CI builds?

(total responses = 801, multiple answers)
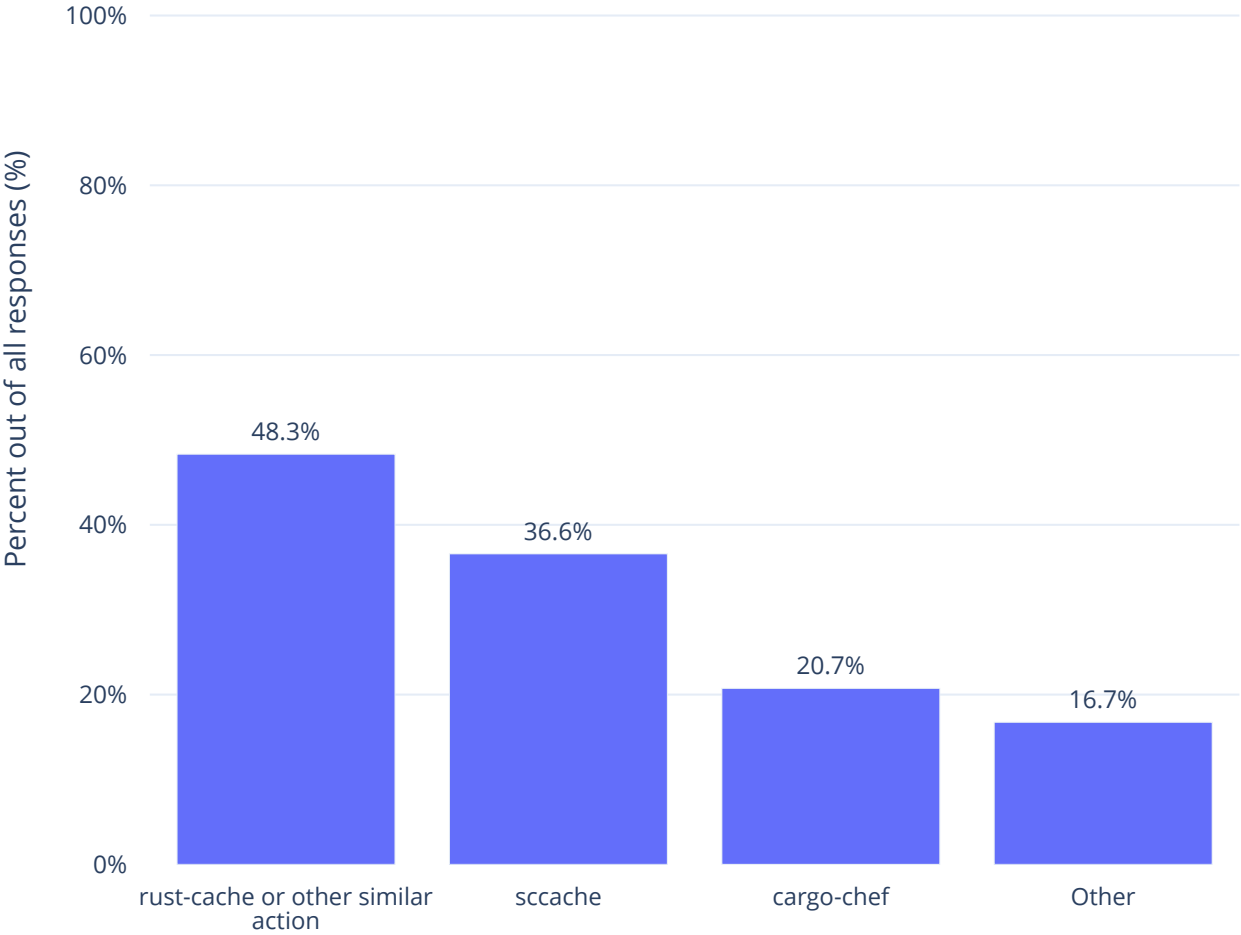
Wordcloud of open answers for the previous chart:

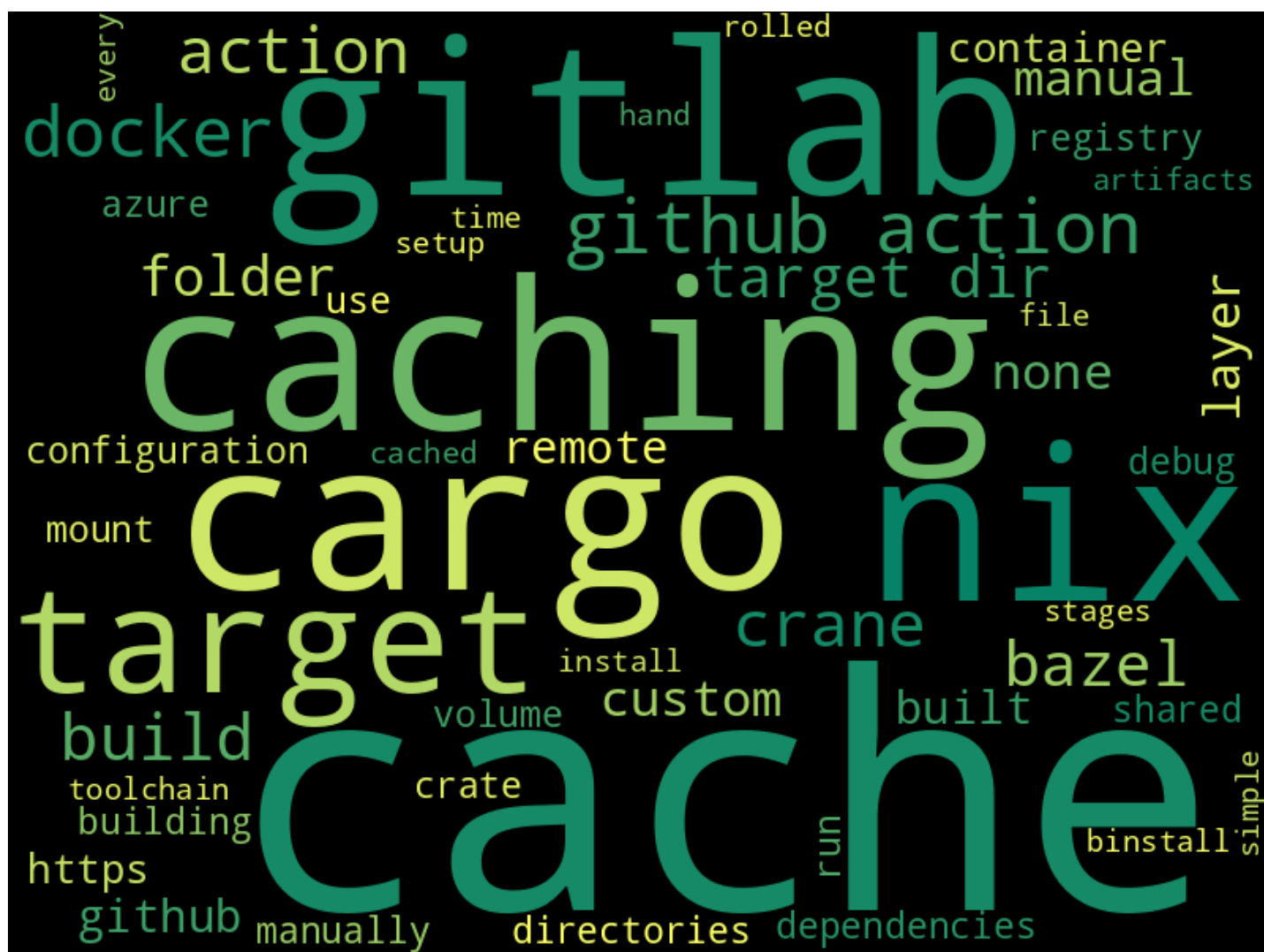## How large is the Rust project that you work on in terms of lines of Rust code?

(total responses = 2469, single answer)
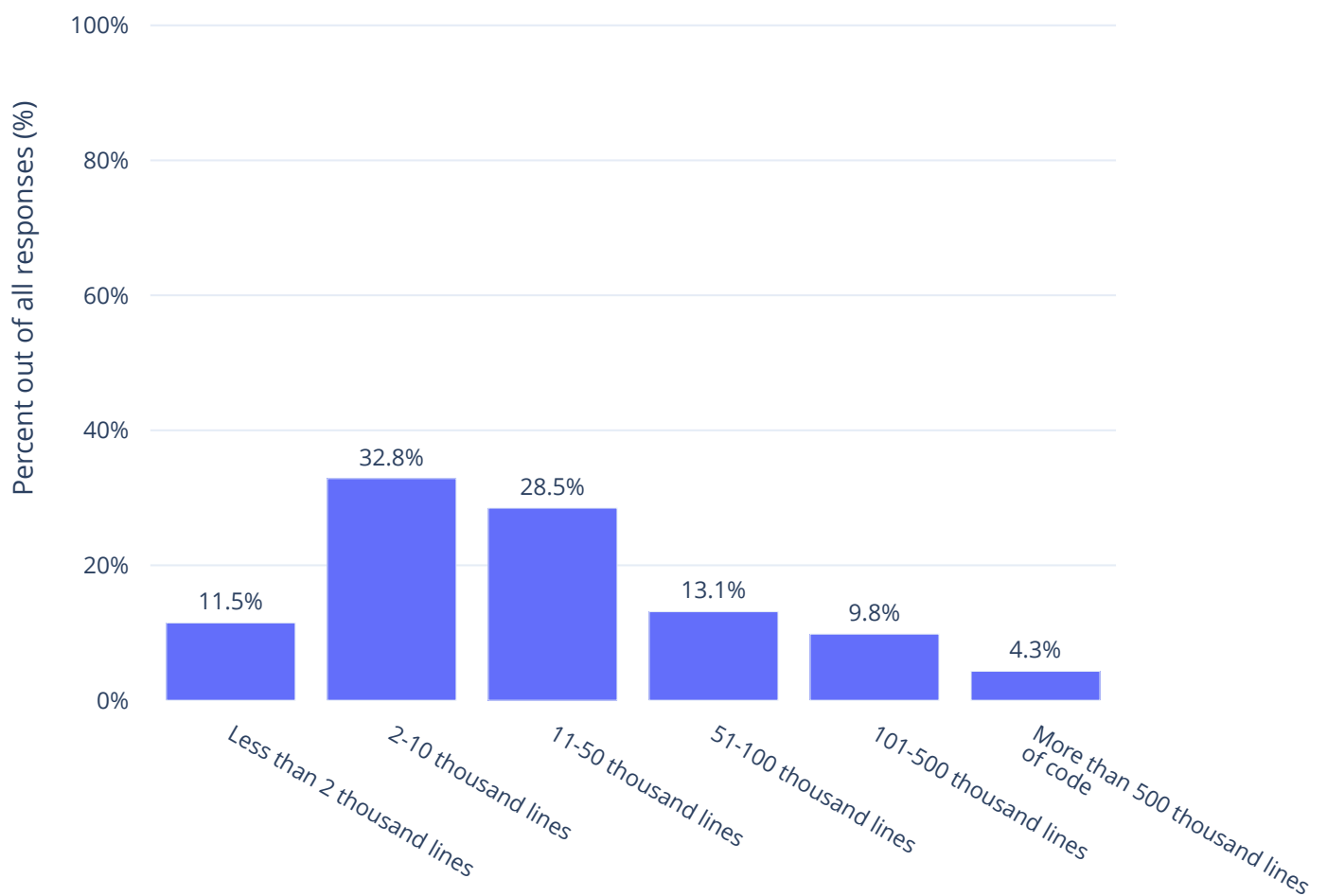
How large is the Rust project that you work on in terms of (Cargo) dependencies?

(total responses = 2473, single answer)

Percent out of all responses (%)

- No dependencies: 0.7%
- 1-10 dependencies: 12.1%
- 11-50 dependencies: 22.9%
- 51-100 dependencies: 12.3%
- 101-200 dependencies: 11.0%
- 201-300 dependencies: 11.0%
- 301-500 dependencies: 13.4%
- More than 500 dependencies: 16.7%
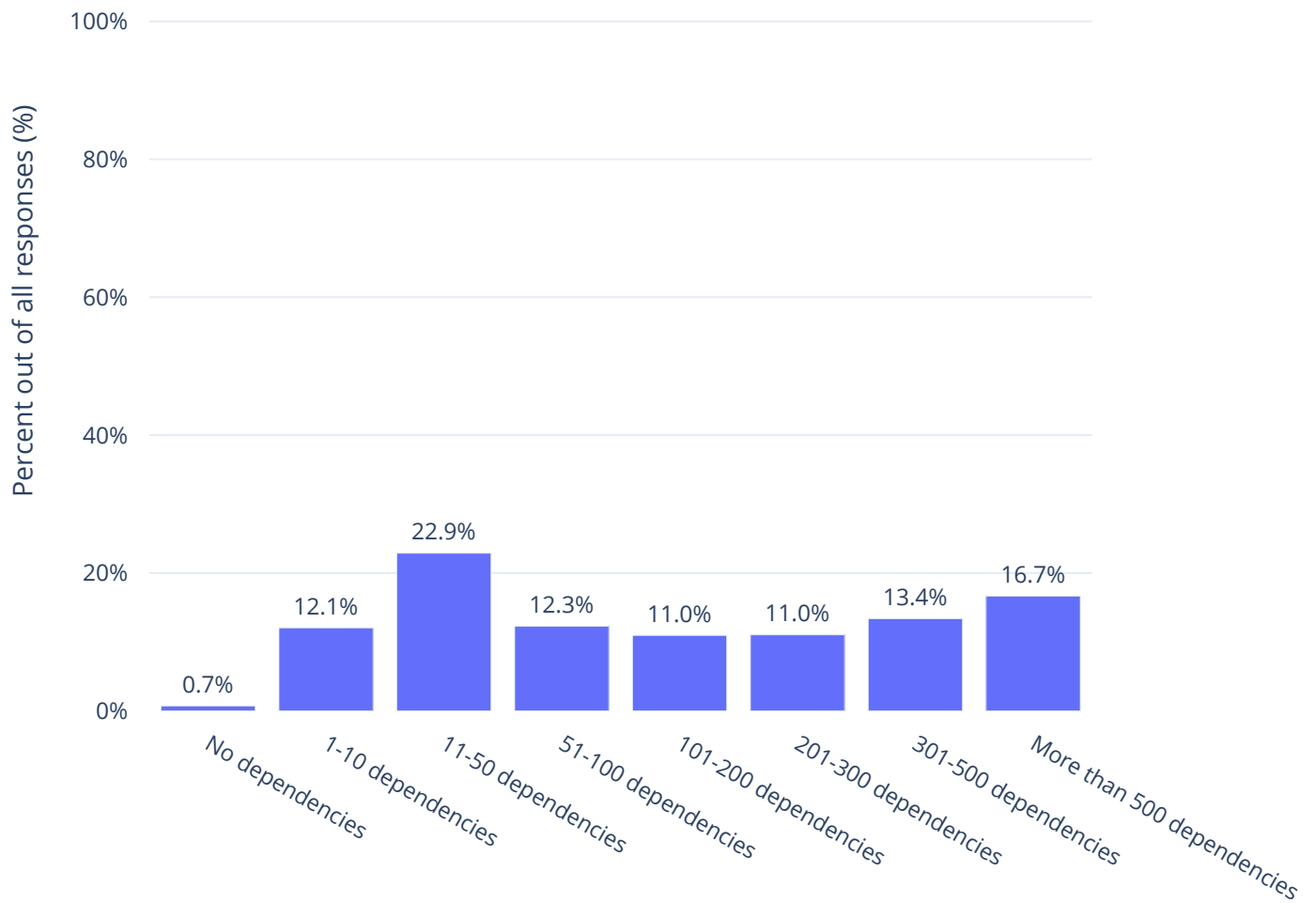
How long do you need to wait for the compiler to rebuild your code after making a change?

(total responses = 2445, single answer)
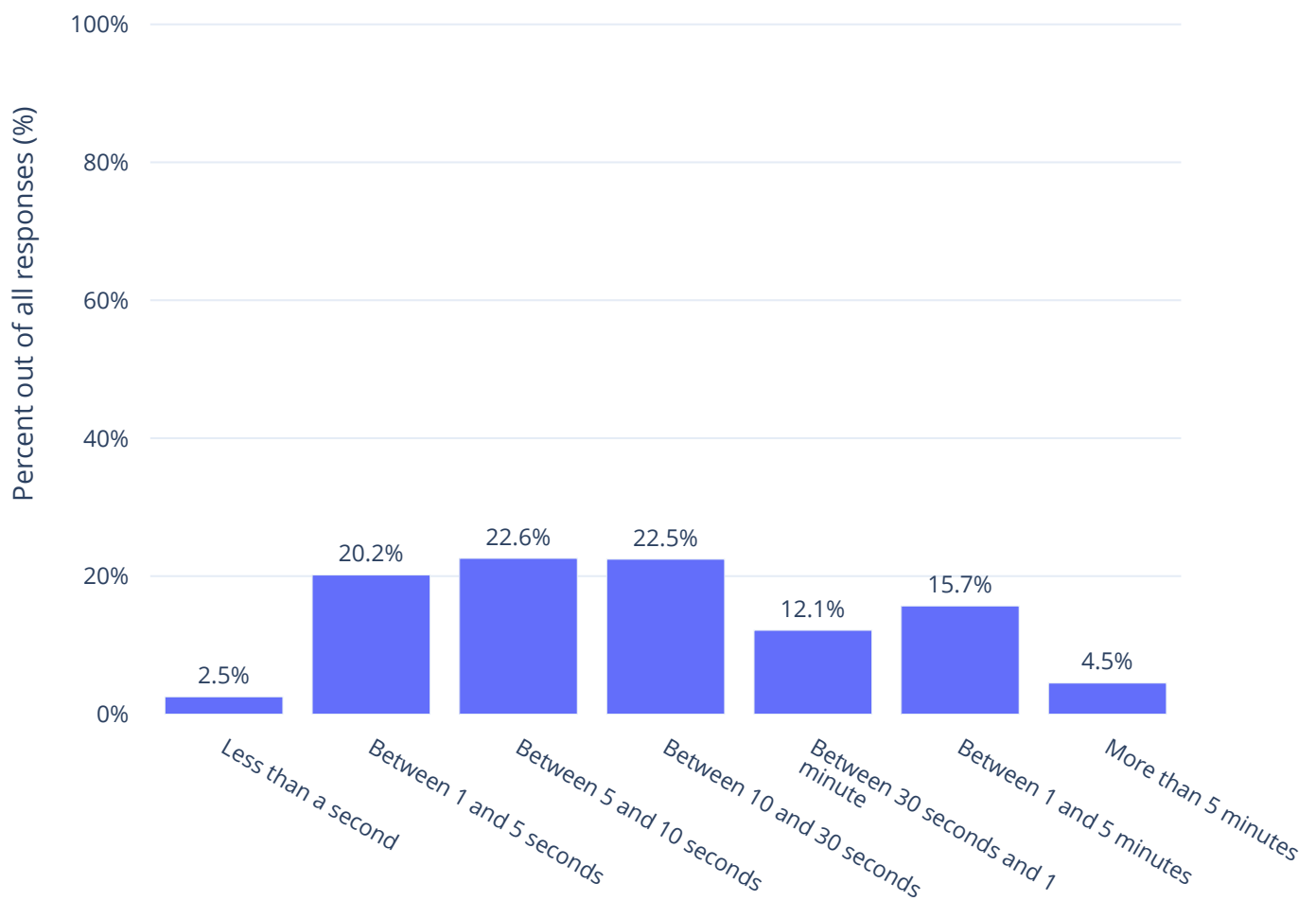
Average rebuild time based on OS (%)

The percentages are relative to each individual group on the X axis
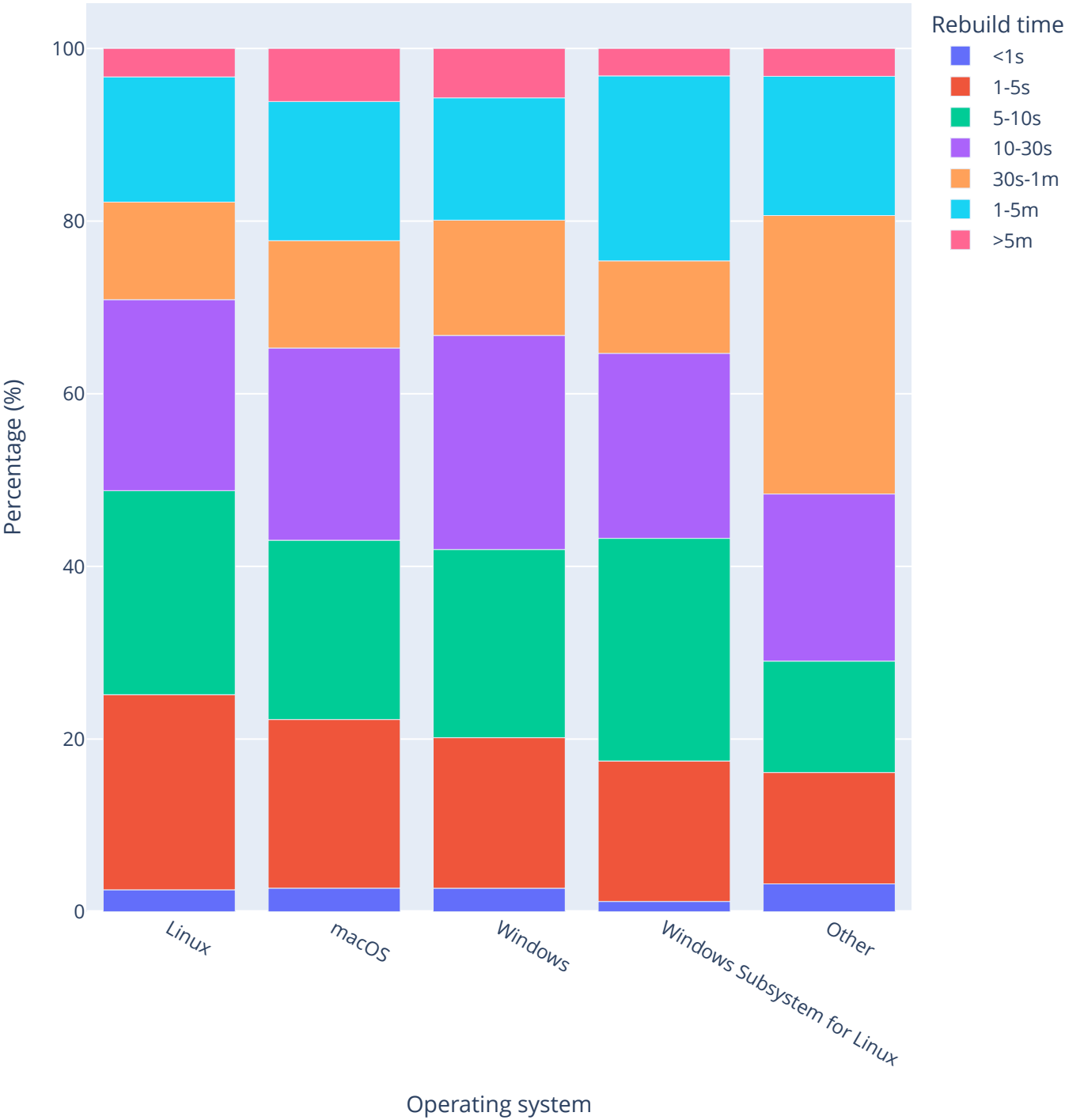
Rebuild time
- <1s
- 1-5s
- 5-10s
- 10-30s
- 30s-1m
- 1-5m
- >5m

Average rebuild time based on project size (%)

The percentages are relative to each individual group on the X axis

Rebuild time
- <1s
- 1-5s
- 5-10s
- 10-30s
- 30s-1m
- 1-5m
- >5m

Percentage (%)

Codebase size

Less than 2 thousand lines
2-10 thousand lines
11-50 thousand lines
51-100 thousand lines
101-500 thousand lines
More than 500 thousand lines of code
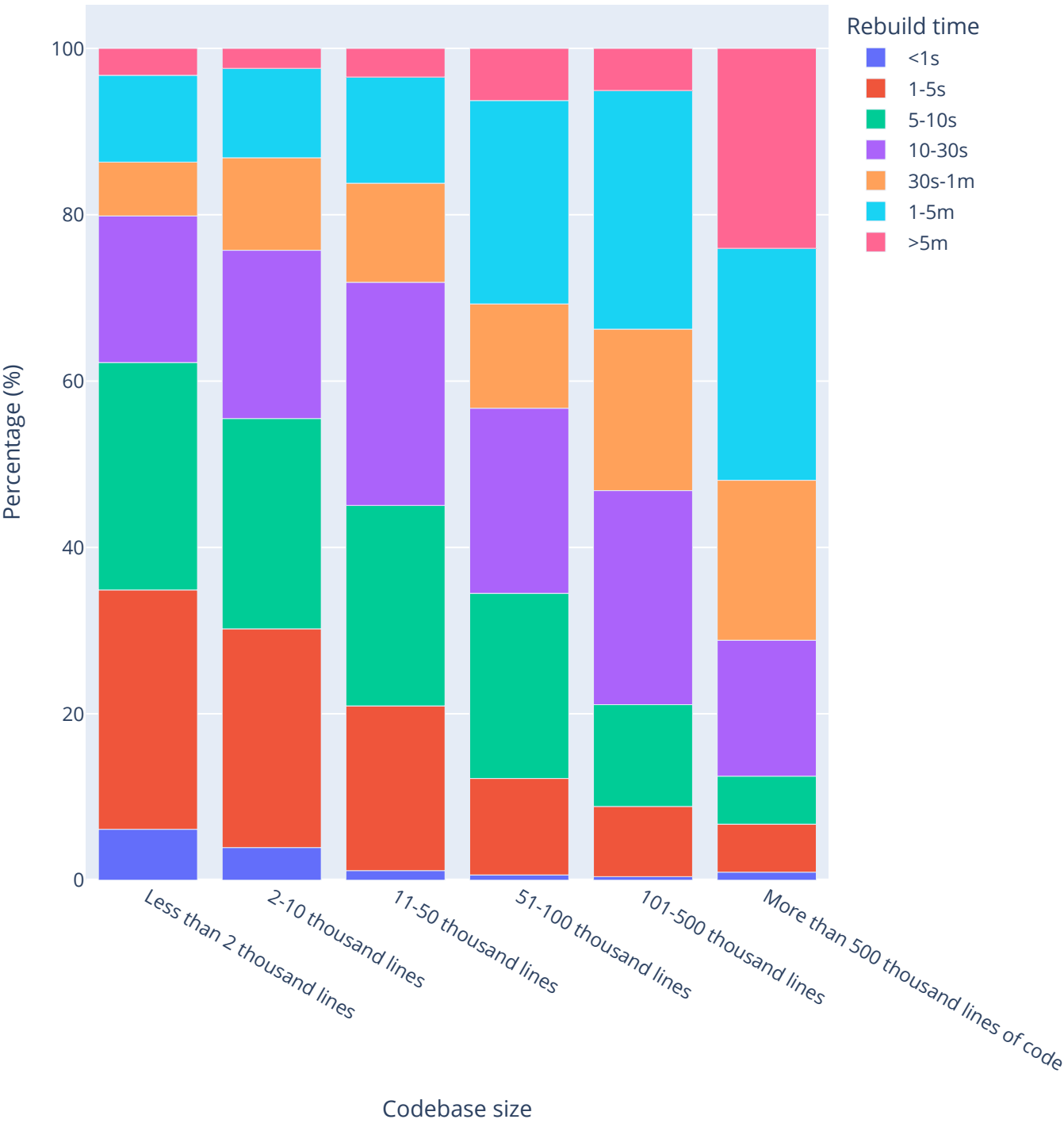
Average rebuild time based on project size and dependency count (%)

The percentages are relative to each individual group on the X axis
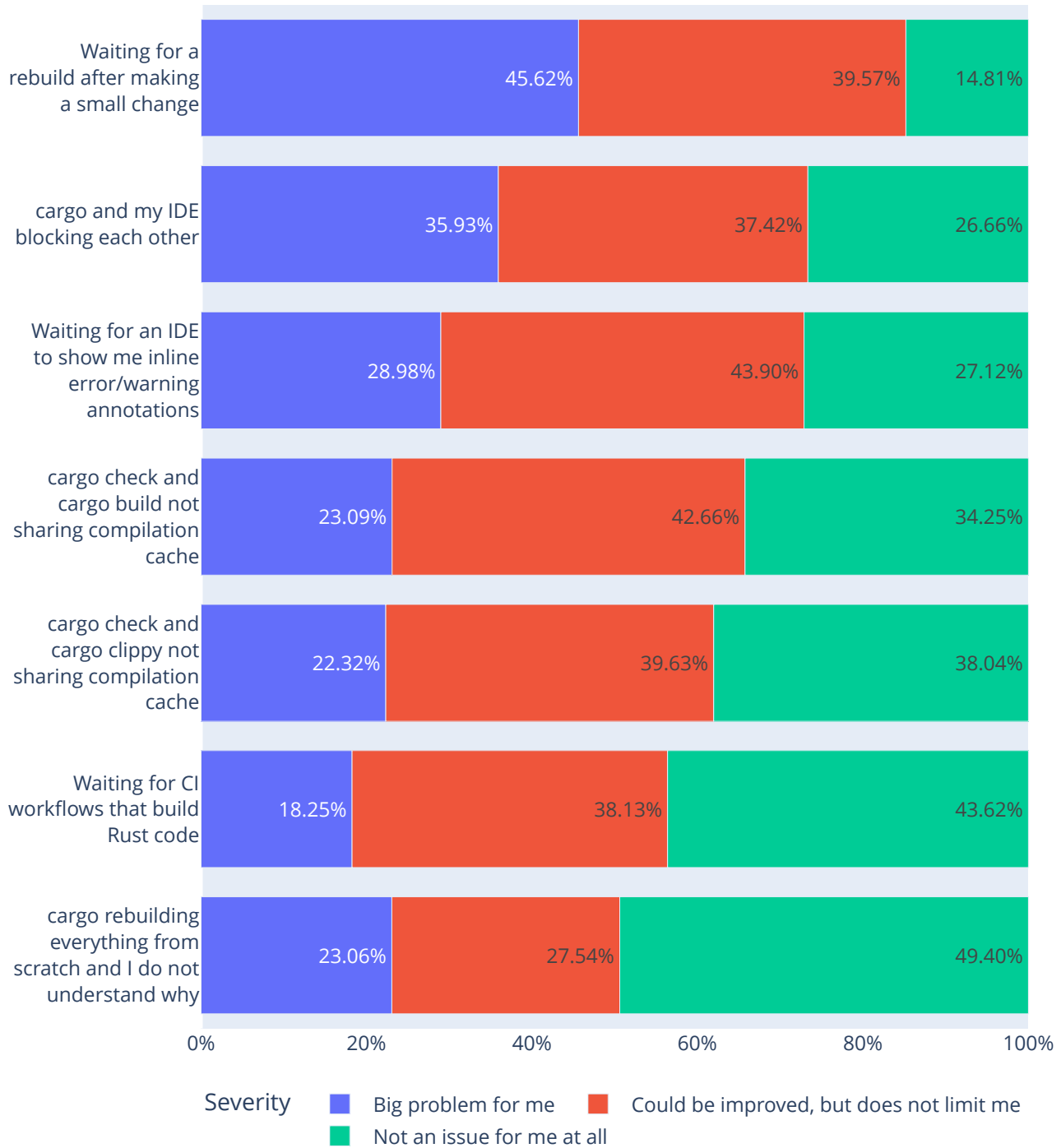

Rebuild time
- <1s
- 1-5s
- 5-10s
- 10-30s
- 30s-1m
- 1-5m
- >5m
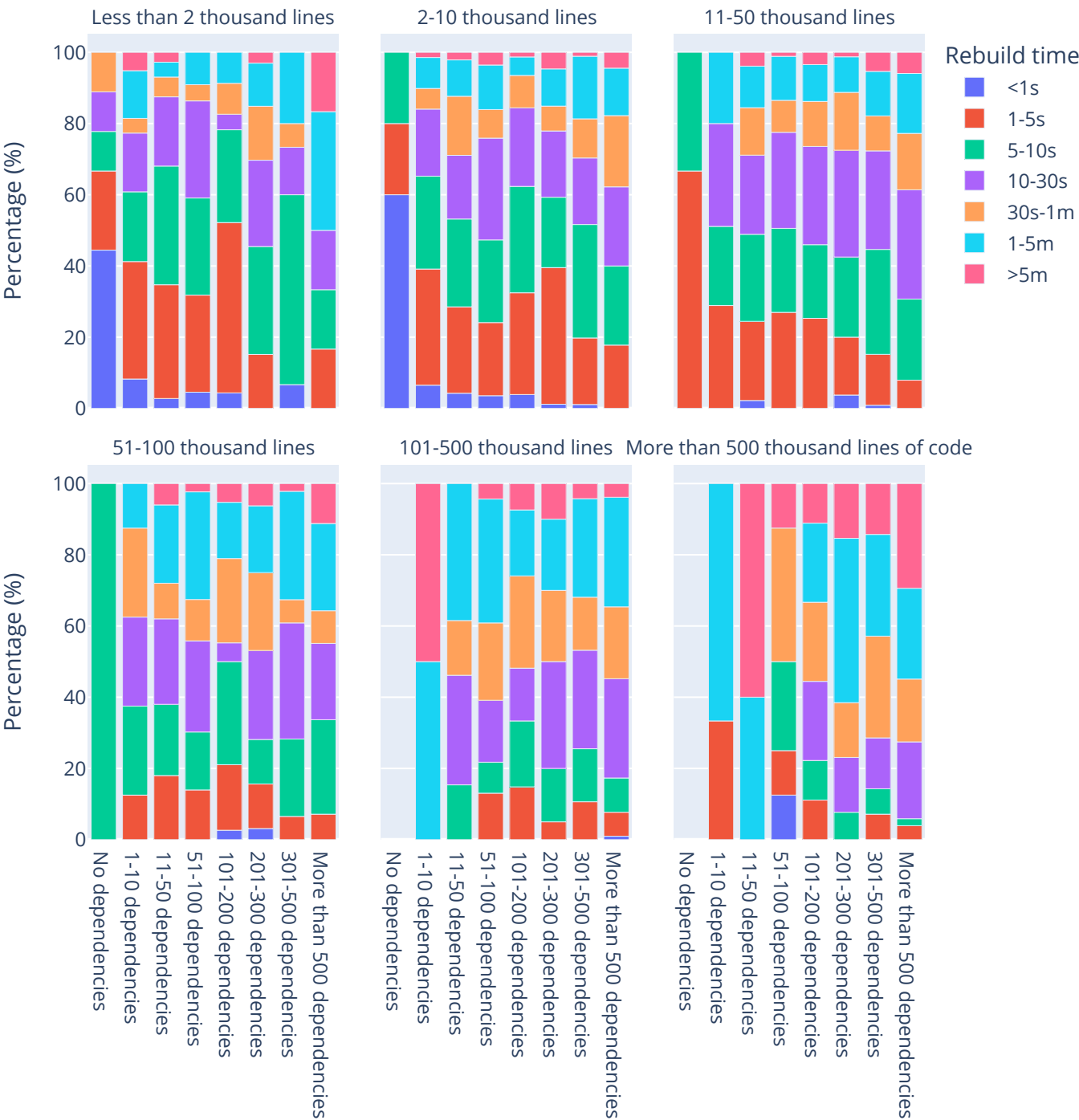
Which of the following problems do you most struggle with?

(total responses = 2463)

Severity			
Waiting for a rebuild after making a small change	45.62%	39.57%	14.81%
cargo and my IDE blocking each other	35.93%	37.42%	26.66%
Waiting for an IDE to show me inline error/warning annotations	28.98%	43.90%	27.12%
cargo check and cargo build not sharing compilation cache	23.09%	42.66%	34.25%
cargo check and cargo clippy not sharing compilation cache	22.32%	39.63%	38.04%
Waiting for CI workflows that build Rust code	18.25%	38.13%	43.62%
cargo rebuilding everything from scratch and I do not understand why	23.06%	27.54%	49.40%

Severity
- Big problem for me
- Could be improved, but does not limit me
- Not an issue for me at all

How do you primarily examine compilation errors in your code?

(total responses = 2447, multiple answers)
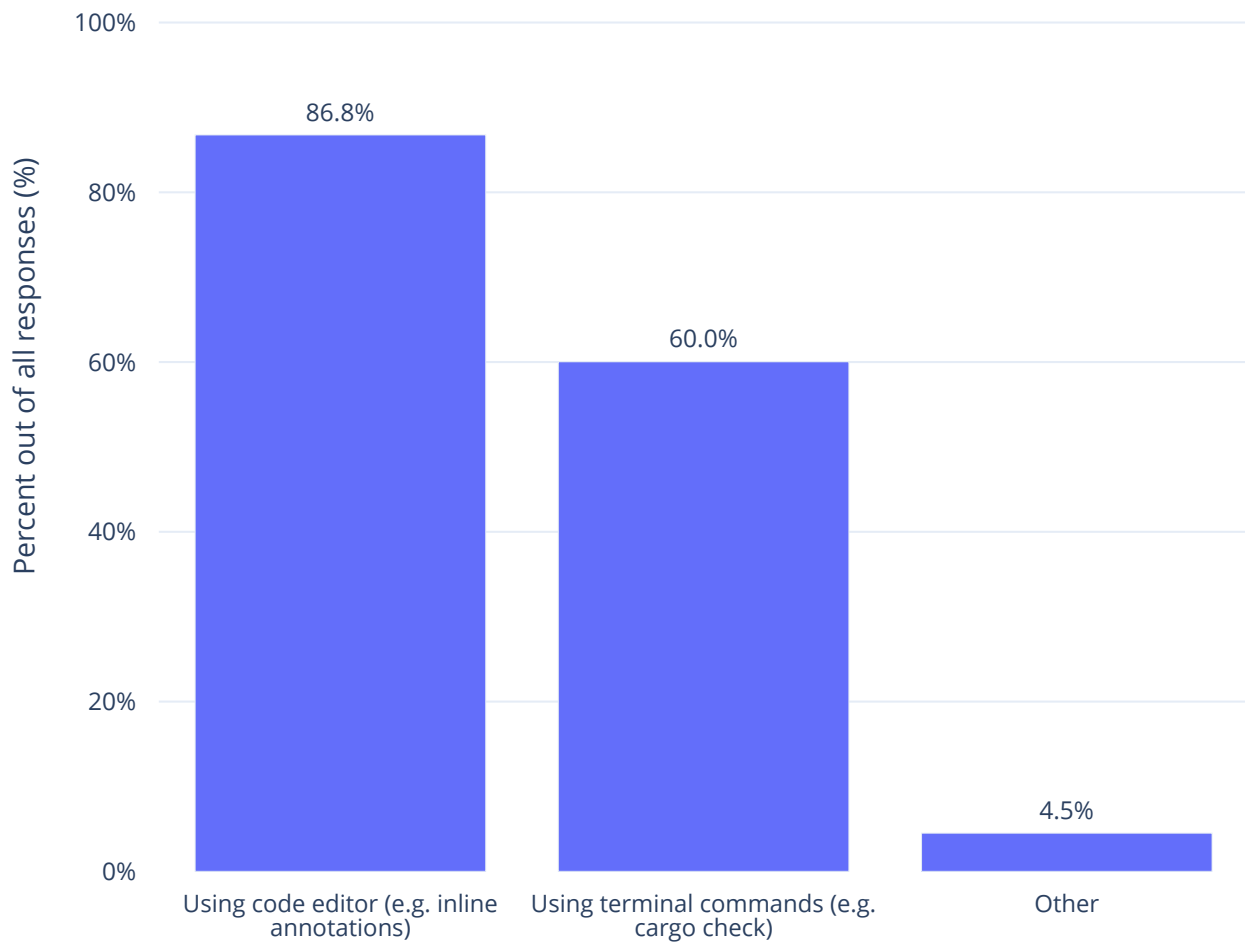
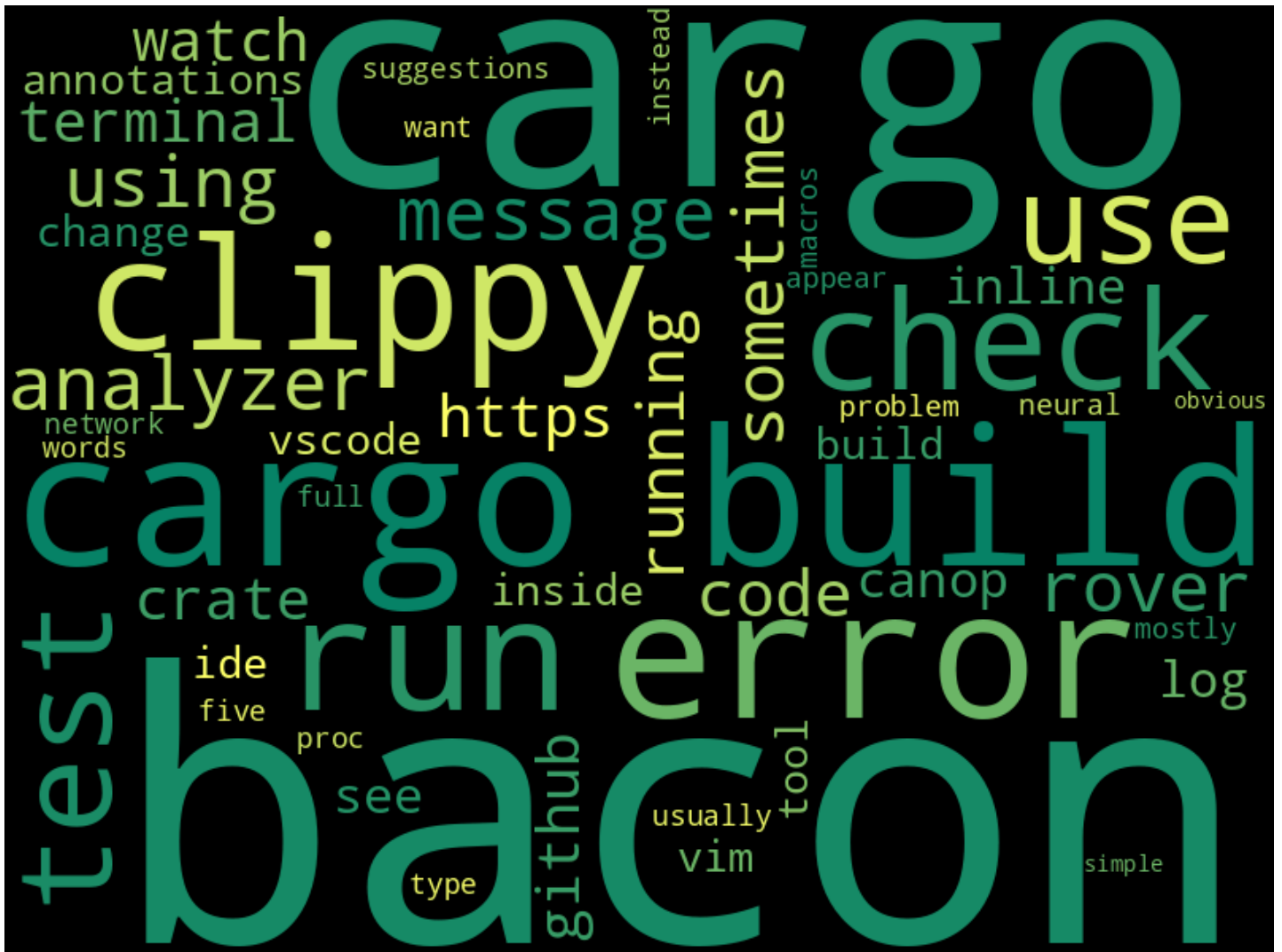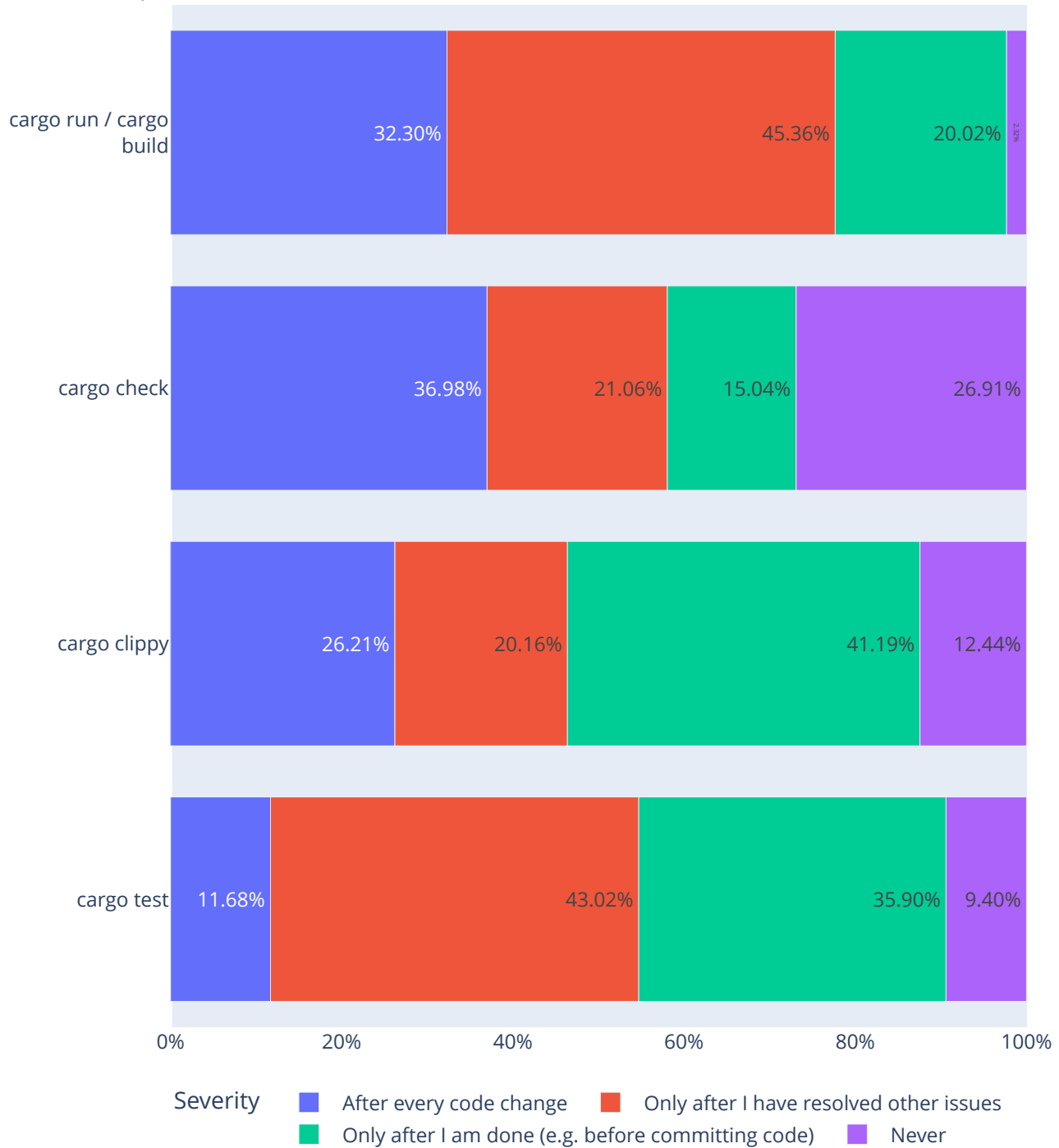Wordcloud of open answers for the previous chart:

If you use Cargo, how often do you use the following commands after each change to Rust code?

(total responses = 2437)

	Severity
■ After every code change	■ Only after I have resolved other issues
■ Only after I am done (e.g. before committing code)	■ Never

cargo run / cargo build: 32.30%, 45.36%, 20.02%, 2.32%

cargo check: 36.98%, 21.06%, 15.04%, 26.91%

cargo clippy: 26.21%, 20.16%, 41.19%, 12.44%

cargo test: 11.68%, 43.02%, 35.90%, 9.40%

Are you aware of the following tools for profiling Rust compilation?

(total responses = 2469)

Tool	I use it	I know of it, but don't use it	Never heard of it
Cargo timings (cargo build --timings)	18.93%	34.28%	46.79%
Compiler self profiler (-Zself-profile)	6.13%	29.55%	64.33%
Compiler pass analysis (-Ztime-passes)	5.40%	25.78%	68.82%
LLVM function instantiation analysis (cargo-llvm-lines)	6.13%	23.01%	70.86%

Severity   ■ I use it   ■ I know of it, but don't use it   ■ Never heard of it
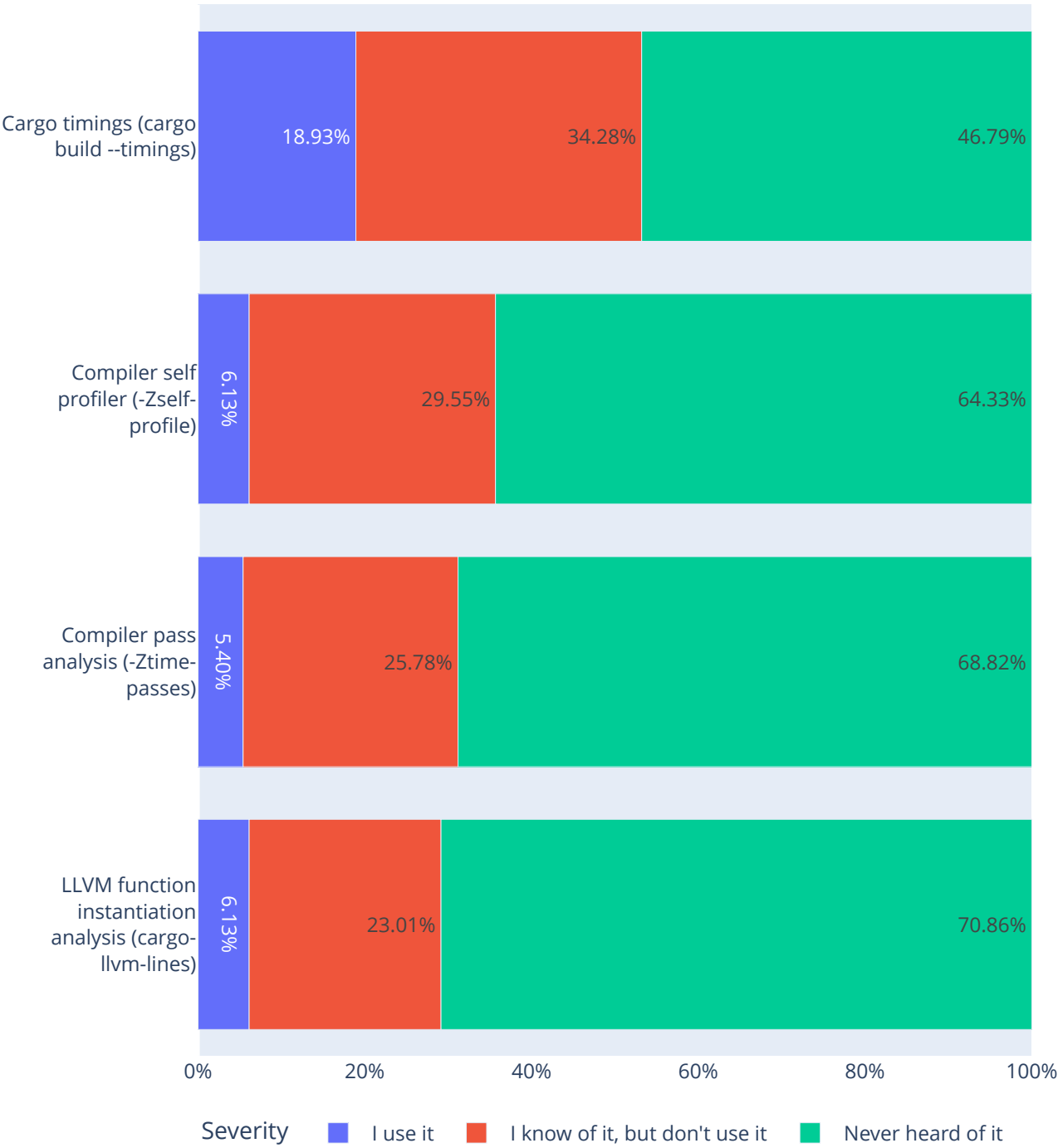
Have you used any of the following mechanisms to improve compilation performance?

(total responses = 2305)

Mechanism	It helps me improve compile times	I tried it, but it didn't help	I know of it, but haven't tried it	Never heard of it
Reduce dependency count	53.06%	12.79%	31.34%	2.81%
Disable default Cargo features of dependencies	45.75%	14.56%	28.54%	11.15%
Split crates	41.45%	14.68%	37.08%	6.78%
Create opt-in Cargo features	39.28%	14.23%	36.04%	10.45%
Alternative linker	35.42%	12.00%	36.93%	15.64%
Reduce usage of procedural macros	26.93%	11.07%	53.22%	8.78%
Disable or reduce debuginfo	21.82%	14.02%	44.41%	19.75%
Reduce usage of generic code	12.35%	10.02%	62.08%	15.55%
Caching compiler wrapper	16.04%	12.62%	39.94%	31.40%
Parallel compiler frontend	13.70%	10.52%	37.75%	38.02%
Cranelift backend	8.19%	5.40%	52.81%	33.60%
Share target directory	9.58%	7.82%	36.78%	45.82%
Merge integration tests into a single binary	11.38%	3.98%	36.12%	48.52%

Severity
- It helps me improve compile times
- I tried it, but it didn't help
- I know of it, but haven't tried it
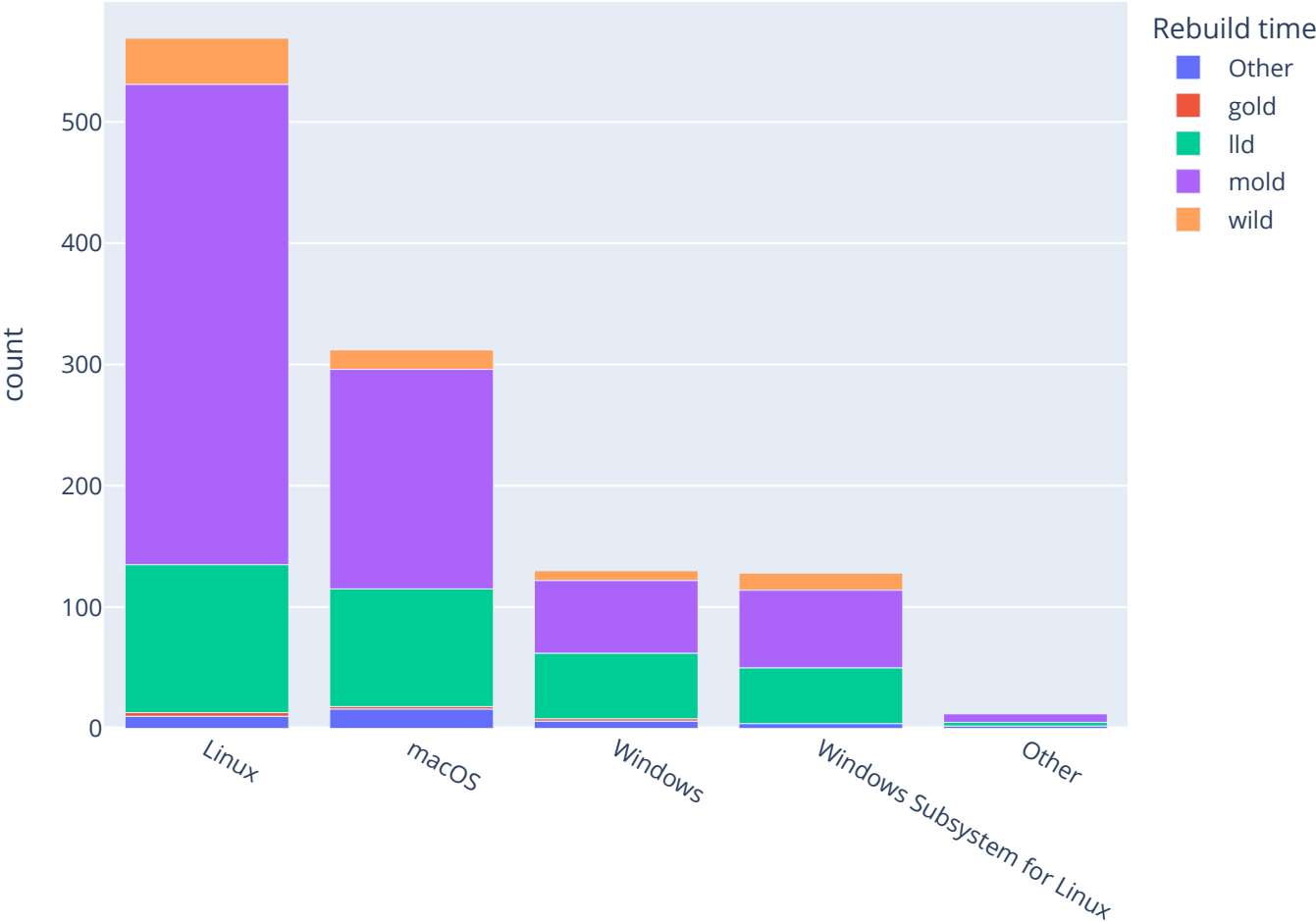- Never heard of it

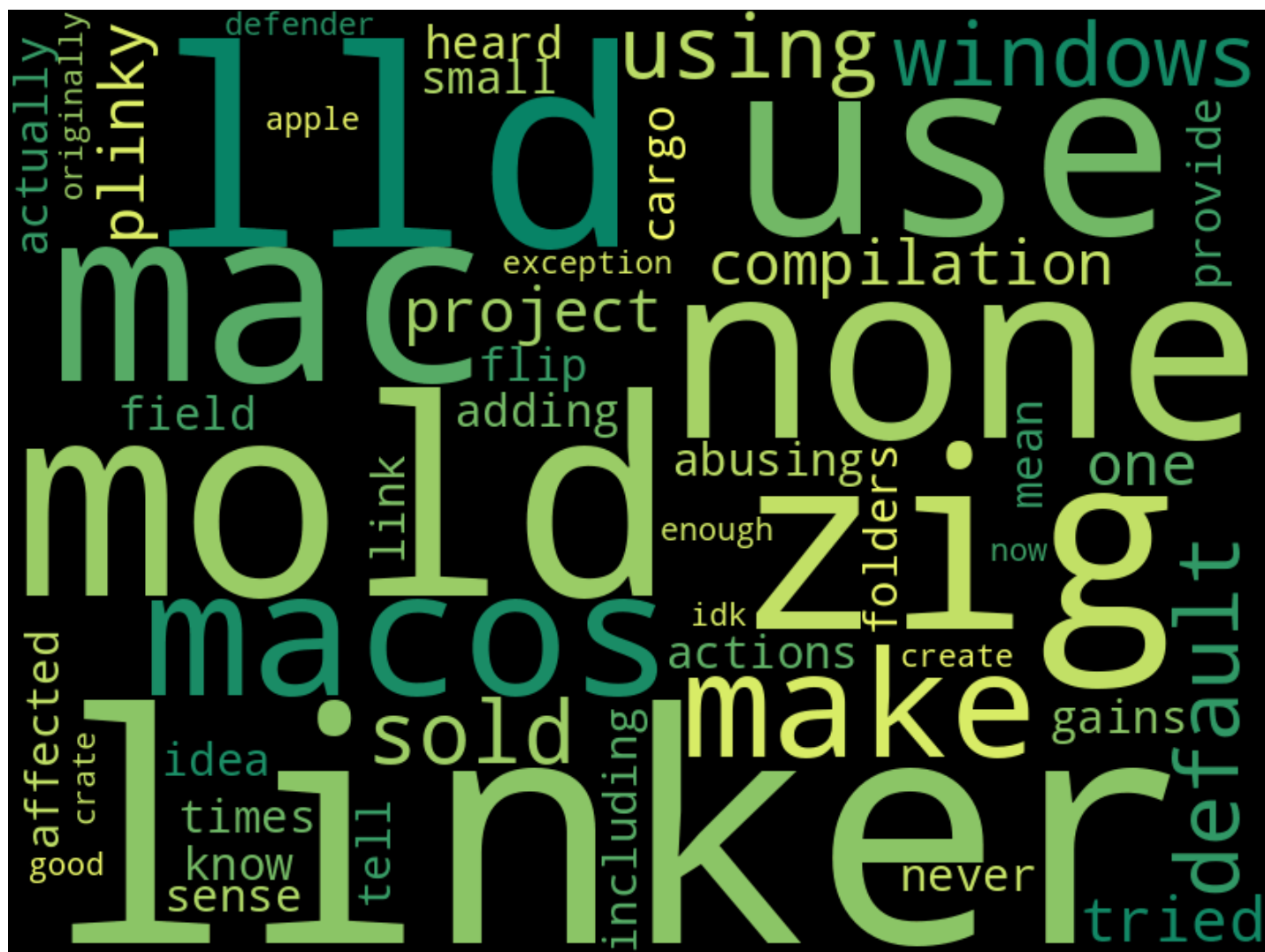# If you use an alternative linker, which one do you use?

(total responses = 1154, multiple answers)
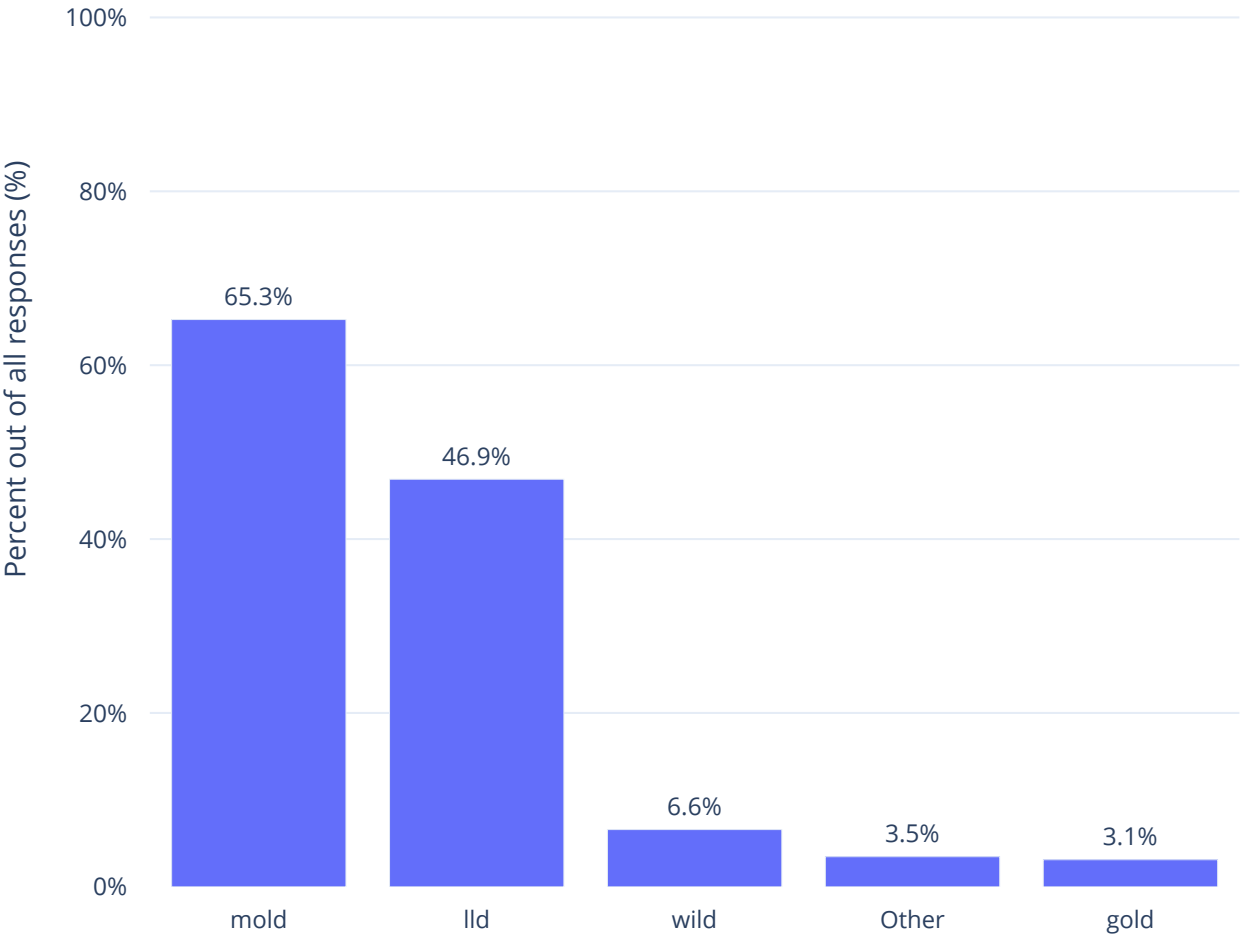
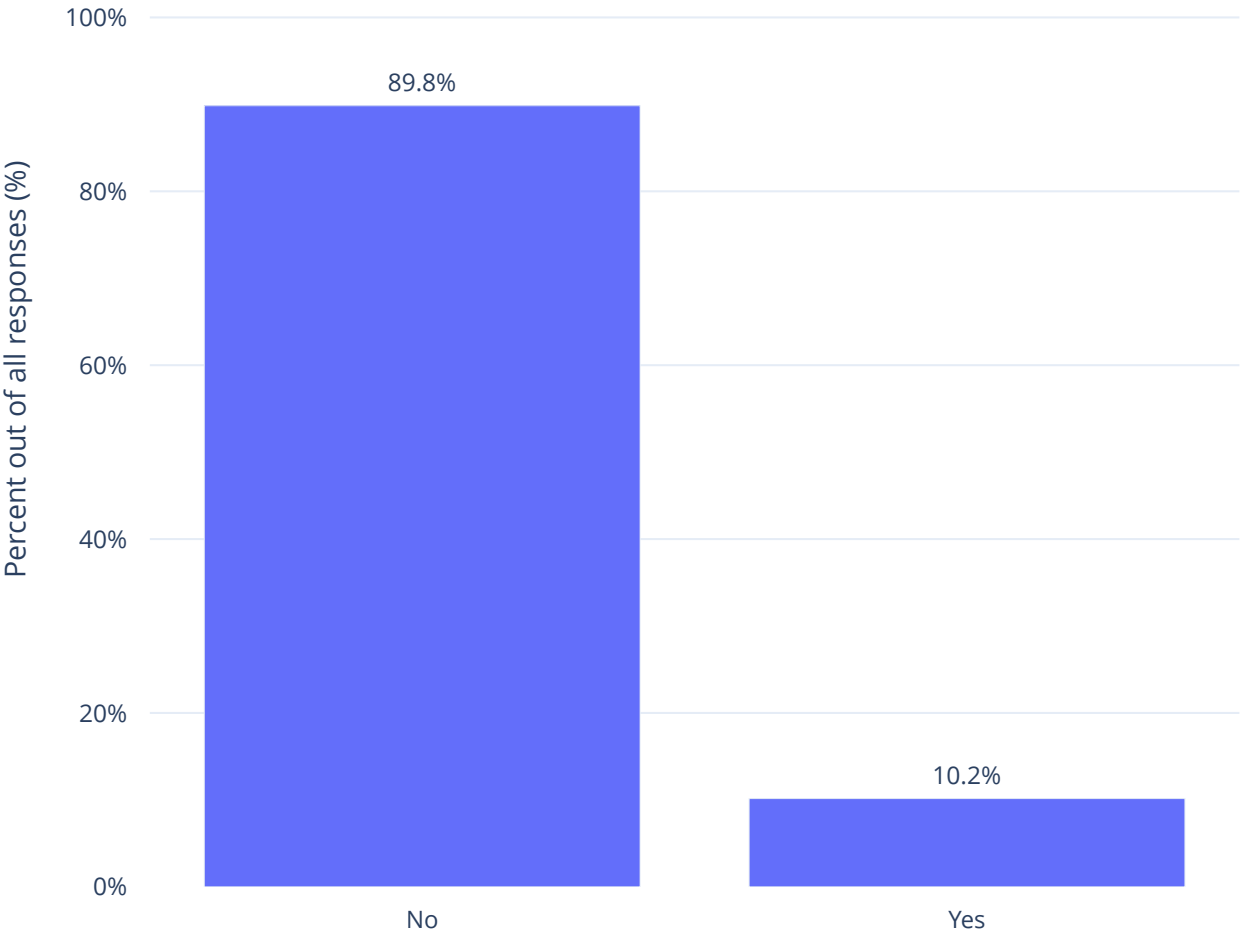# Alternative linker used based on OS (absolute counts)

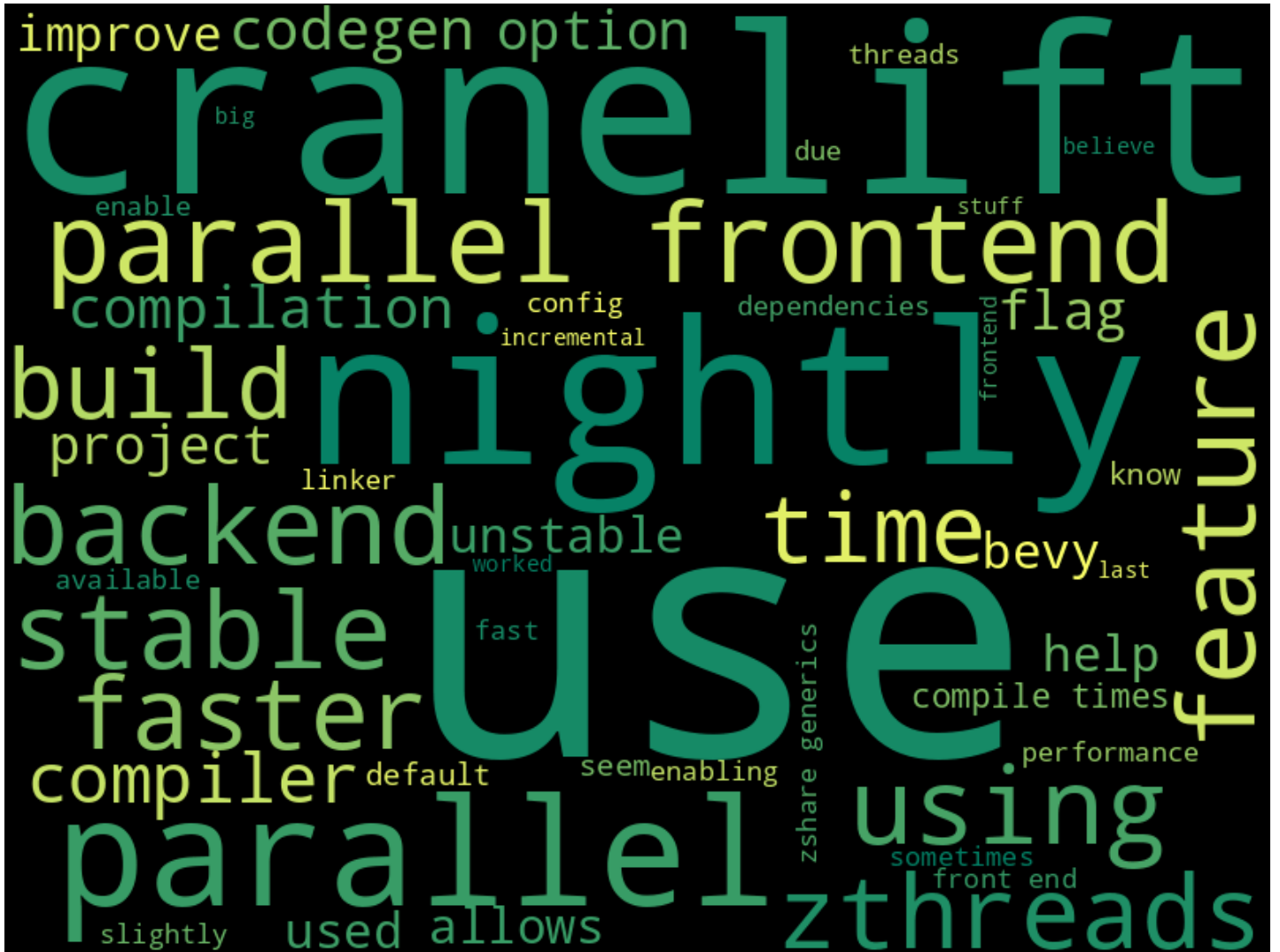Wordcloud of open answers for the previous chart:

Do you use a nightly compiler to achieve better compilation performance?
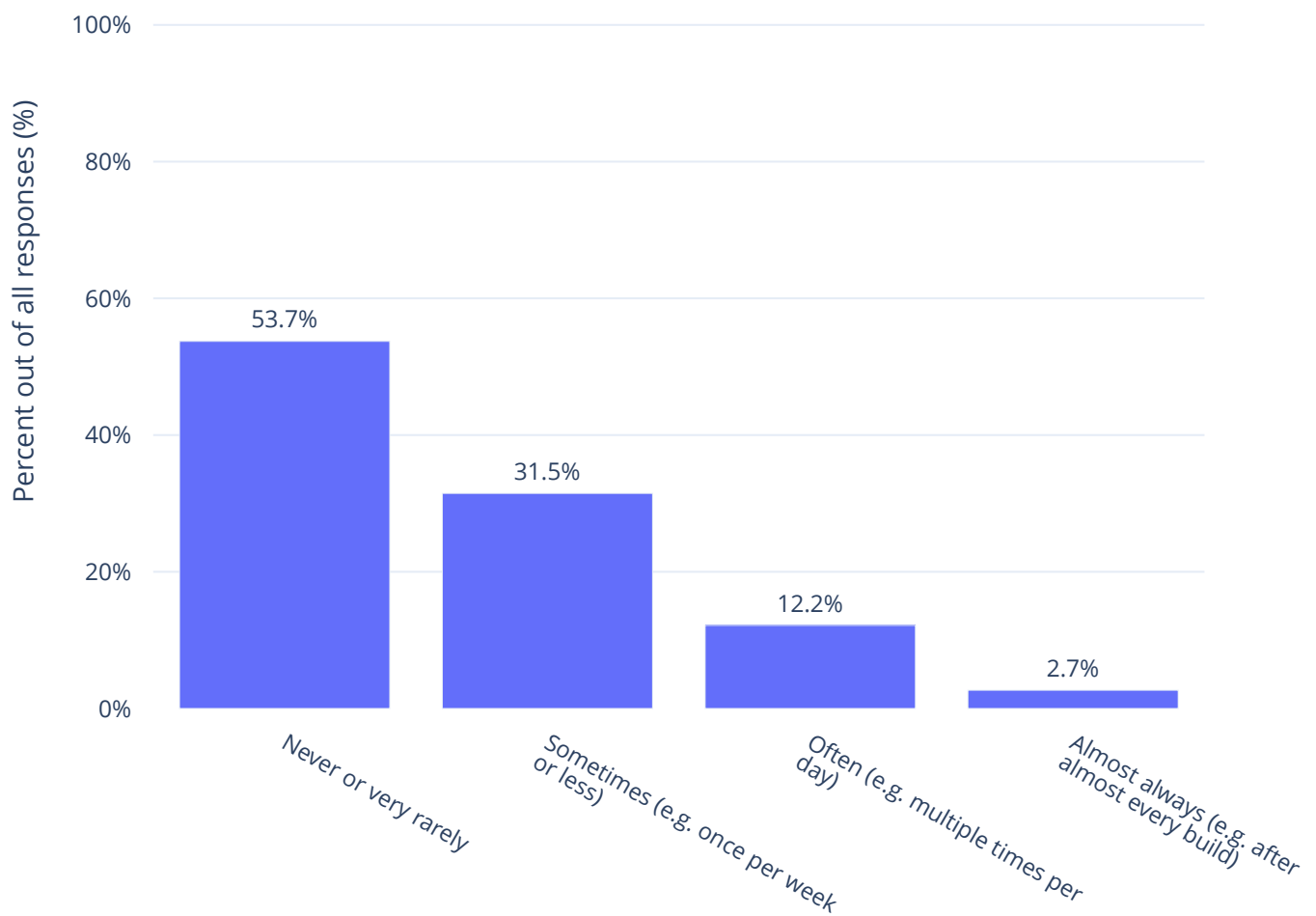
(total responses = 2265, single answer)

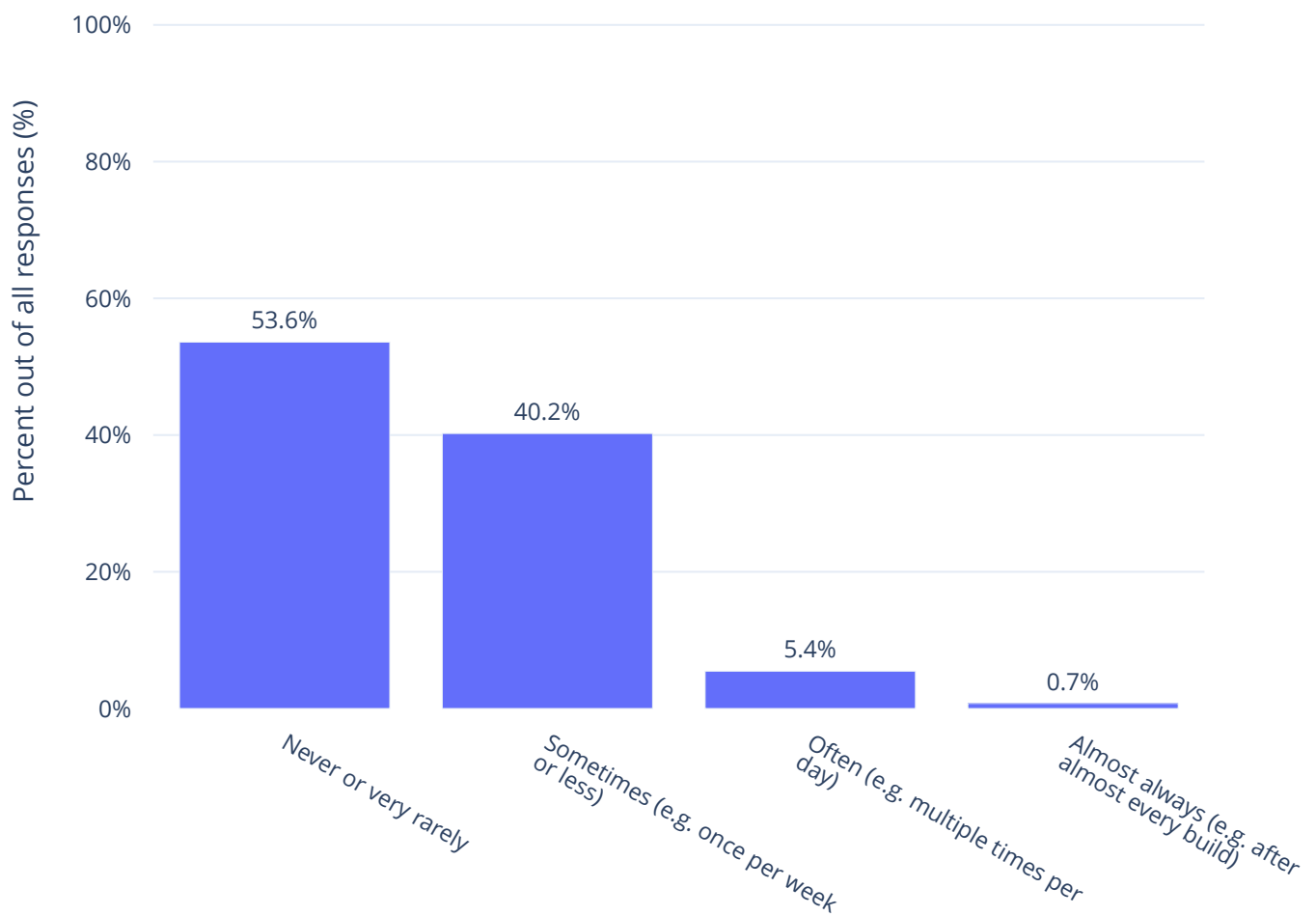Wordcloud of open answers for the previous chart:

How often do you use a debugger to debug your Rust code?
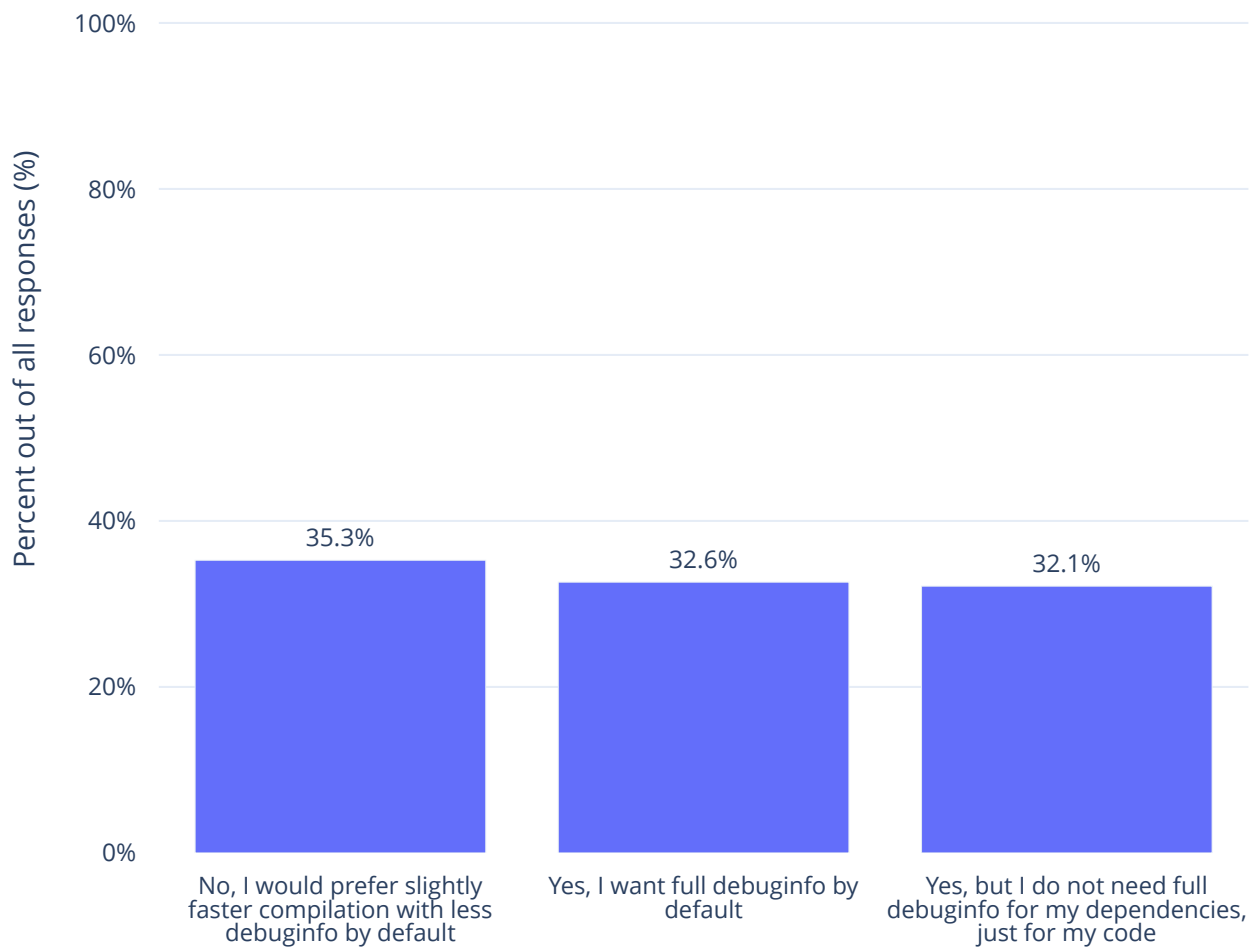
(total responses = 2320, single answer)

How often do you use a profiler to profile your Rust code?
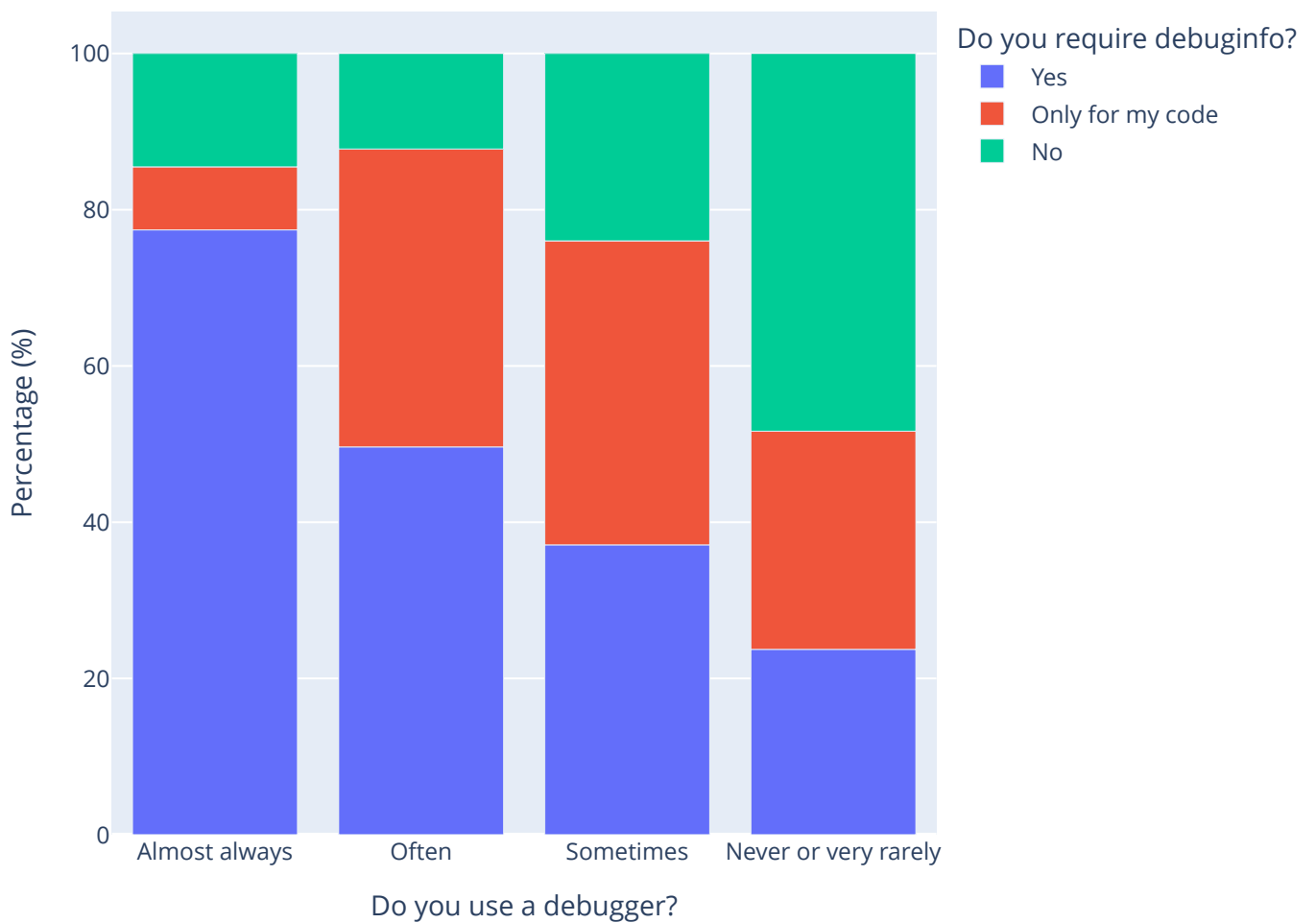
(total responses = 2312, single answer)

Chart axis:
- Y-axis: Percent out of all responses (%), 0% to 100%
- Bars:
  - Never or very rarely: 53.6%
  - Sometimes (e.g. once per week or less): 40.2%
  - Often (e.g. multiple times per day): 5.4%
  - Almost always (e.g. after almost every build): 0.7%

Do you require unoptimized builds to have debuginfo by default?

(total responses = 2275, single answer)

Percent out of all responses (%)

35.3%	32.6%	32.1%

No, I would prefer slightly faster compilation with less debuginfo by default

Yes, I want full debuginfo by default

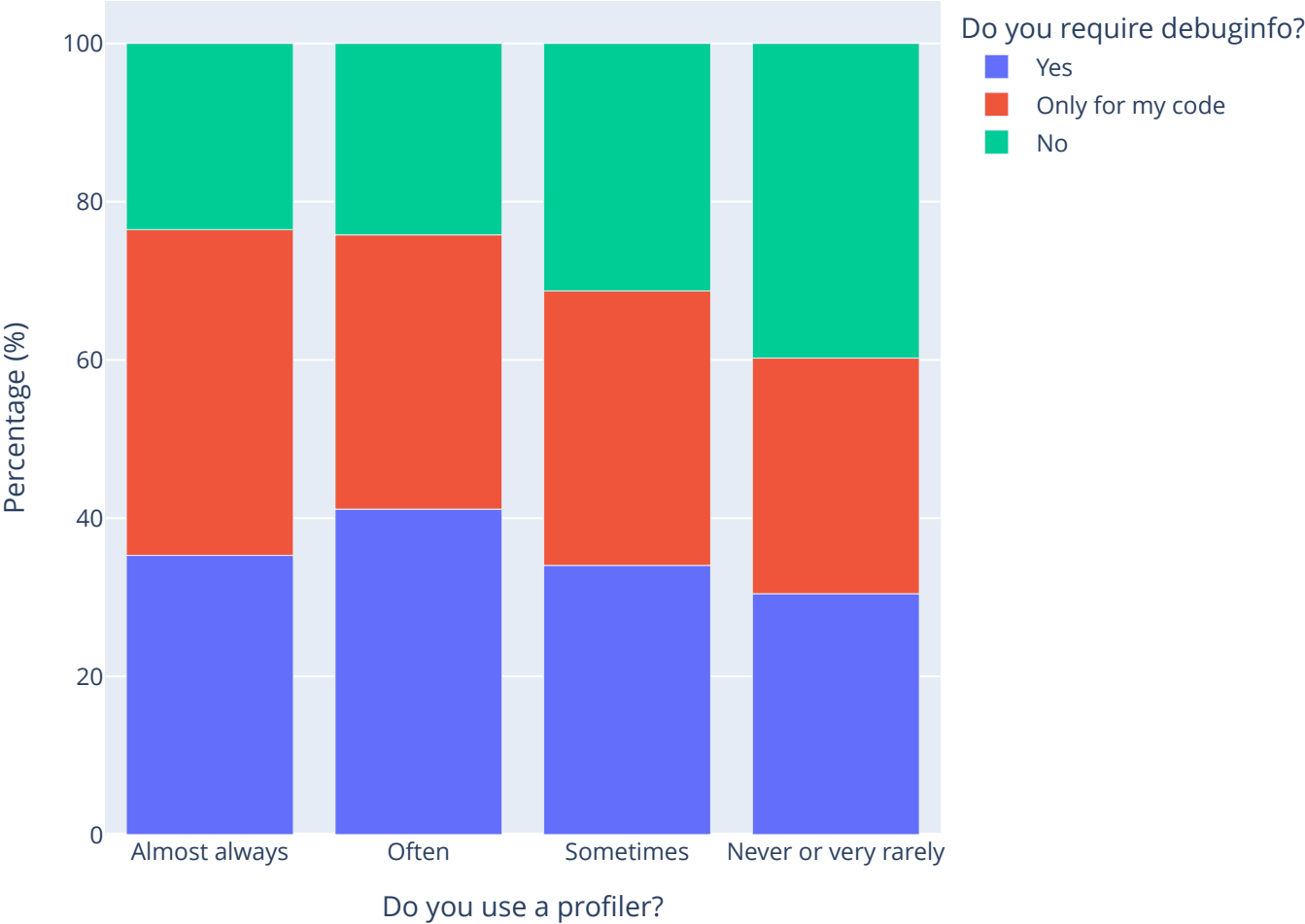Yes, but I do not need full debuginfo for my dependencies, just for my code

Debuginfo requirement based on the usage of a debugger (%)

The percentages are relative to each individual group on the X axis

Do you require debuginfo?
- Yes
- Only for my code
- No

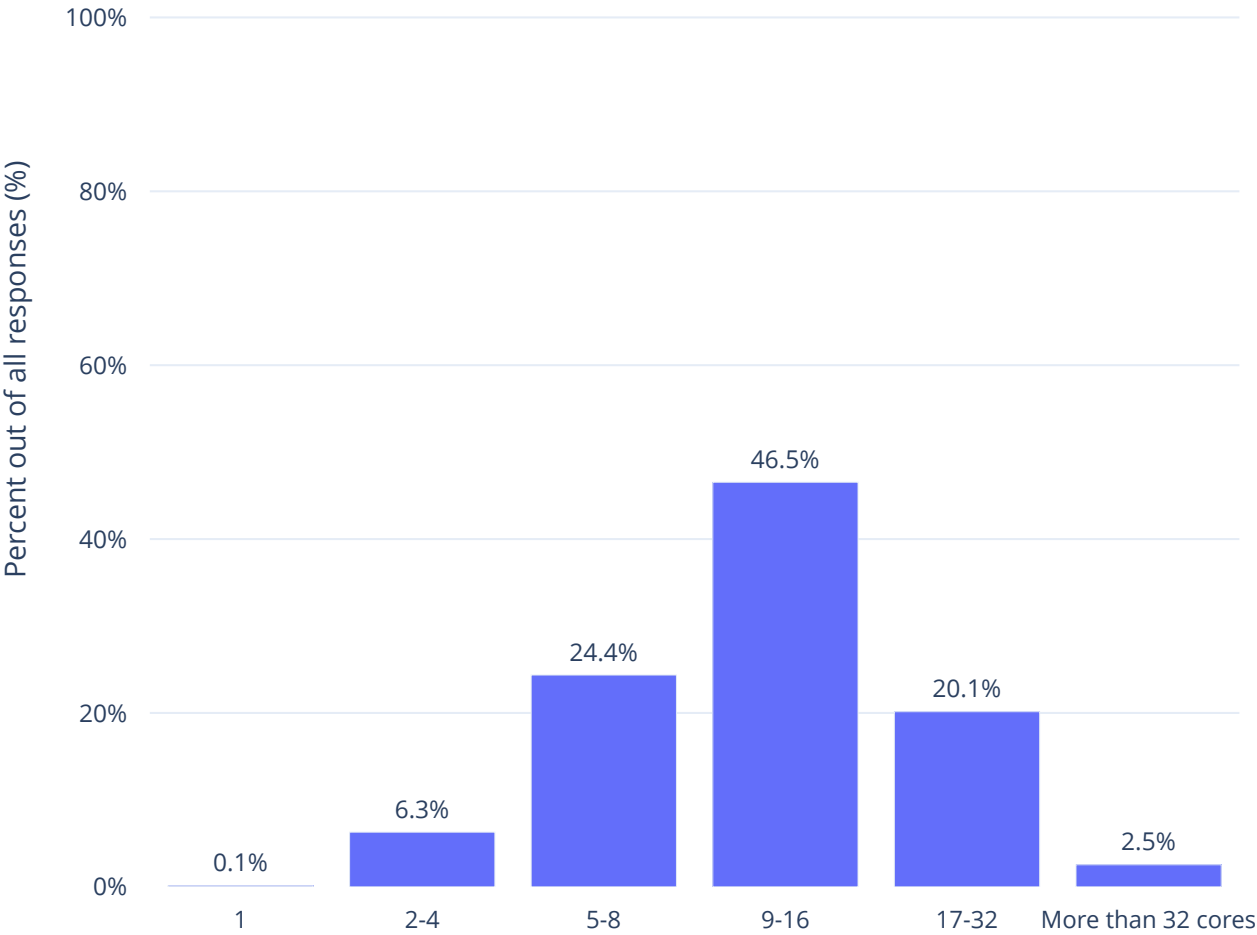# Debuginfo requirement based on the usage of a profiler (%)

The percentages are relative to each individual group on the X axis

Do you require debuginfo?
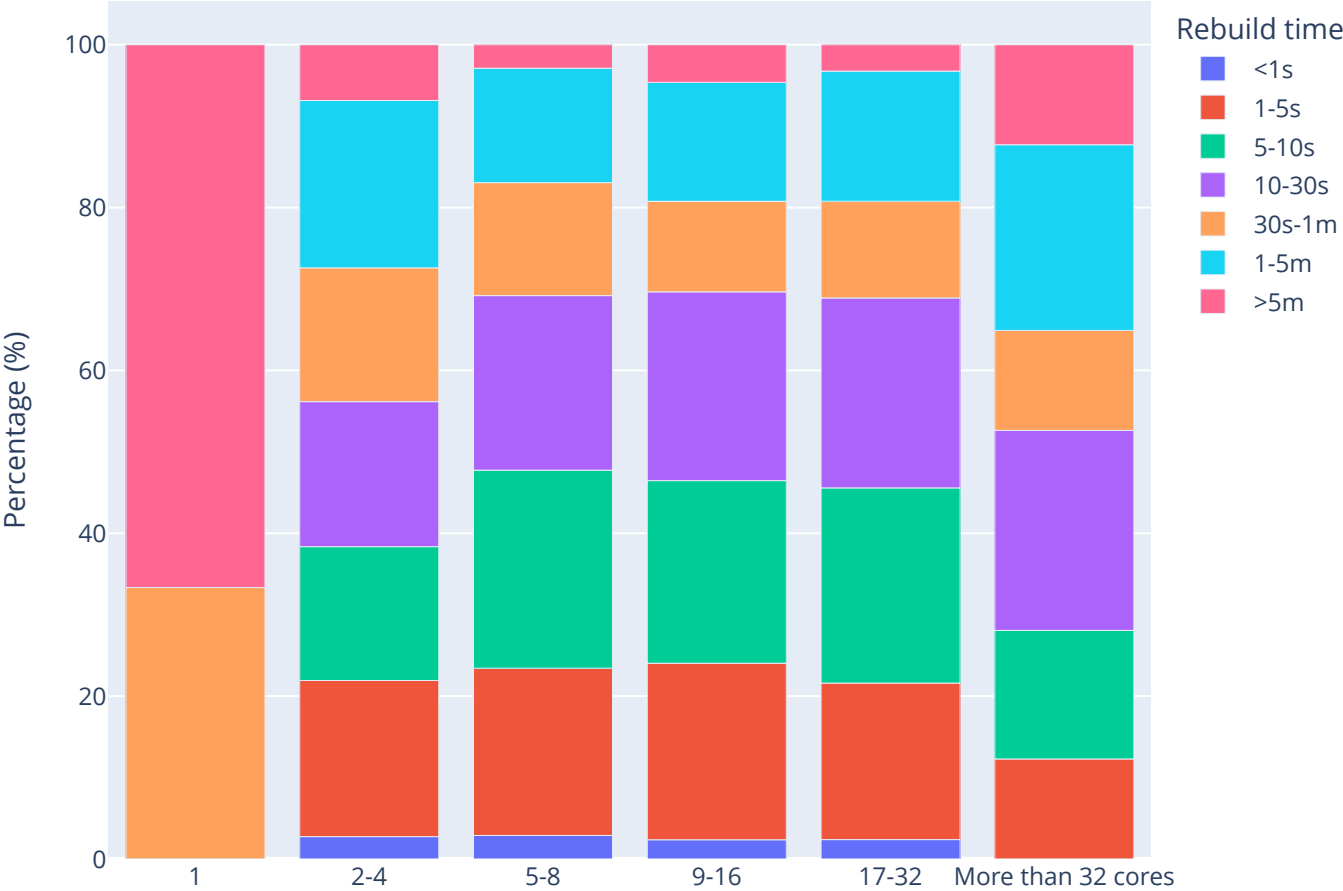- Yes
- Only for my code
- No

How many cores does your computer have?

(total responses = 2323, single answer)

Percent out of all responses (%)

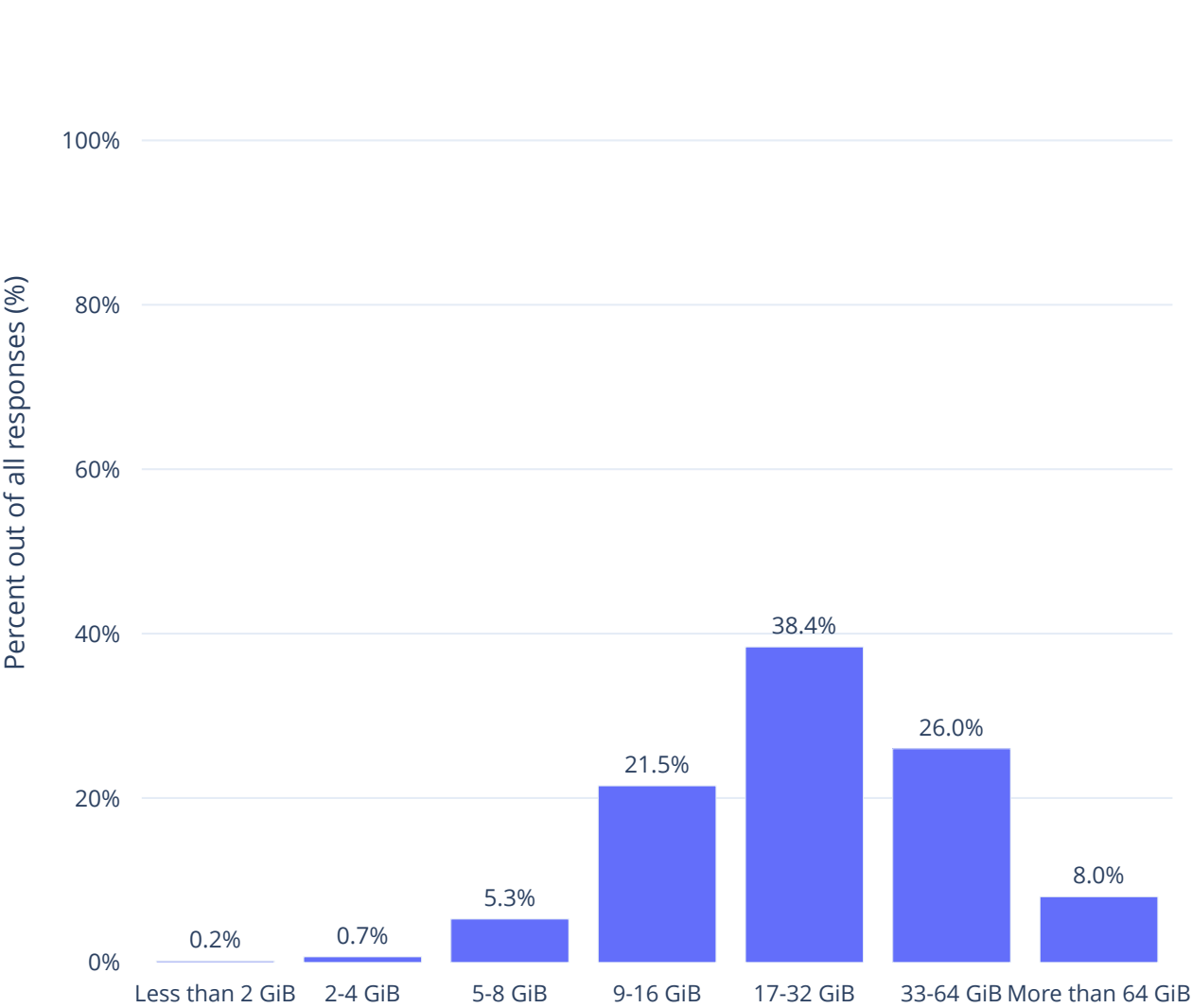| Label	Value
1	0.1%
2-4	6.3%
5-8	24.4%
9-16	46.5%
17-32	20.1%
More than 32 cores	2.5%

Rebuild time based on core count (%)

The percentages are relative to each individual group on the X axis

How much RAM memory does your computer have?

(total responses = 2327, single answer)

Bar chart — Percent out of all responses (%):

- Less than 2 GiB: 0.2%
- 2-4 GiB: 0.7%
- 5-8 GiB: 5.3%
- 9-16 GiB: 21.5%
- 17-32 GiB: 38.4%
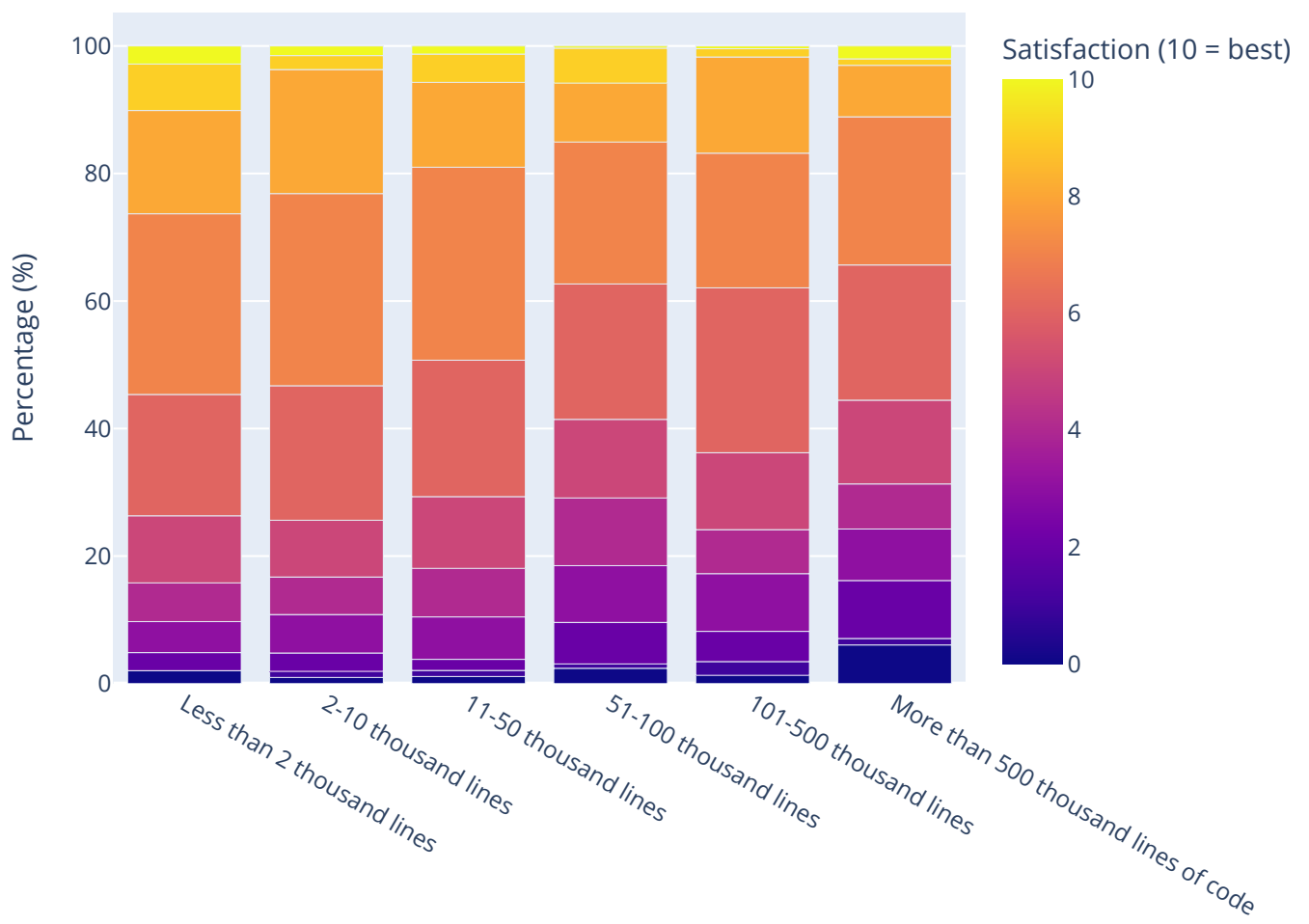- 33-64 GiB: 26.0%
- More than 64 GiB: 8.0%

Overall, how satisfied are you with Rust compilation performance? (0 = worst, 10 = best)

(total responses = 2257, single answer)
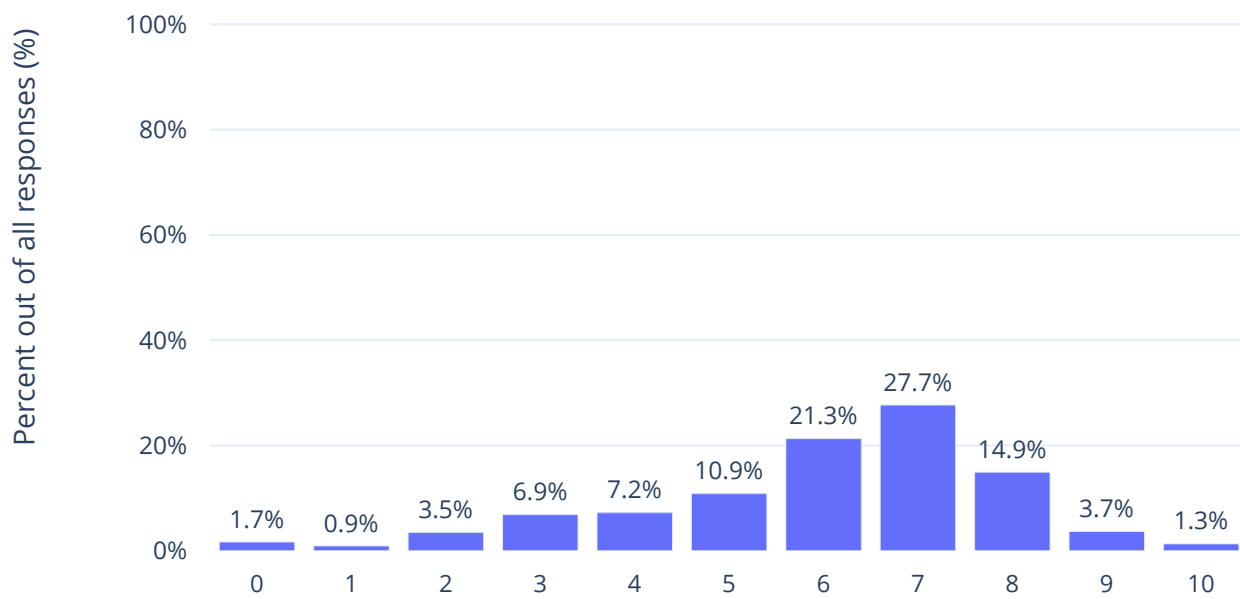
Overall satisfaction based on project size (%)

The percentages are relative to each individual group on the X axis

Overall satisfaction based on operating system (%)

The percentages are relative to each individual group on the X axis