Demystifying metaprogramming and package precompilation

by Cédric Belmant

Julia Optimization Days 2024 — Toulouse — ENSEEIHT

@serenity4

About myself

Engineer by training, majoring in Applied Mathematics.

Involved in Julia projects related to GPU rendering (Vulkan) for 4+ years.

Working as a software consultant for 3+ years, exclusively in Julia.

Metaprogramming

Metaprogramming

Metaprogramming is a computer programming technique in which computer programs have the ability to treat other programs as their data.

Metaprogramming is the ability to programmatically generate code.

```
function generate_code(args...)
  return Expr(...)
end
```

Metaprogramming

- Expresses complex operations with a simple syntax, given well-defined semantics.
- May define domain-specific languages (DSL).
- Generally relies on macros, expanding to code programmatically generated.

```
@constraint model residuals == A * x - b

@constraint(model, residuals == A * x - b)
```

Metaprogramming

When should I use a macro over a function?

- Macros are oftentimes unnecessary.
- If you can use a function, use a function.
- But, macros can be very convenient if you visualize a handy mapping from syntax to code.

Metaprogramming

Demo

https://github.com/serenity4/JuliaOptimizationDays2024

Metaprogramming

Leverage Julia syntax with macro-specific semantics, or by writing source files.

Macros are special functions that return code, executed at parsing time.

Scoping rules and macro hygiene are probably the most difficult part of macros.

Package precompilation

Package precompilation

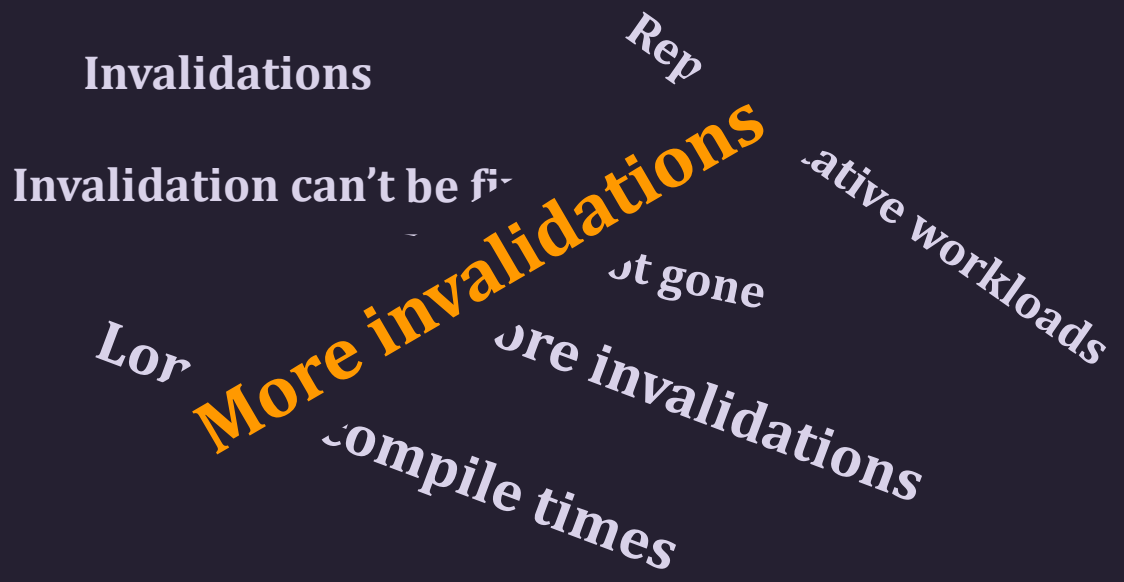
Issue: *package users notice significant latency on first execution.*

Reason: *methods need to be compiled before being executed.*

Solution: *compile methods ahead of time!*

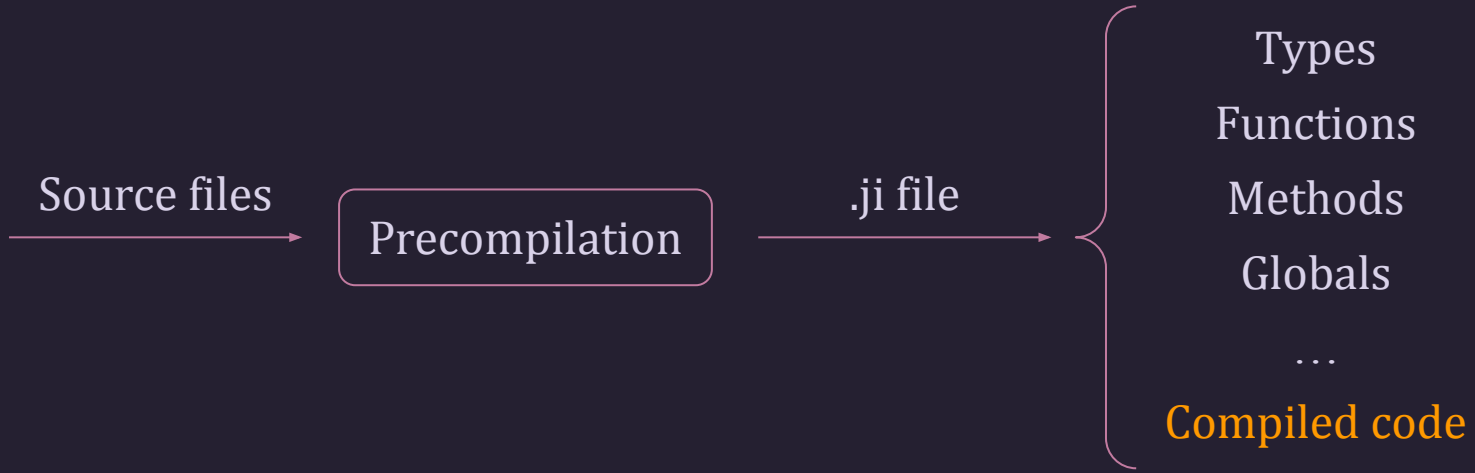
… simple, right?

Package precompilation



… simple, right?

Package precompilation



Package precompilation

Challenges

- Find workloads that fit library usage.
- Avoid excessive compilation.
- Prevent compiled code from being discarded.

Package precompilation

Demo

https://github.com/serenity4/JuliaOptimizationDays2024

Summary

Metaprogramming

Leverage Julia syntax with macro-specific semantics.

Macros are special functions that return code, expanded when parsed.

Scoping rules and hygiene are probably the most difficult part of macros.

Package precompilation

Compiled code gets saved when the package gets processed on first use.

The challenge lies in identifying code paths to explicitly compile during this stage.

Invalidations may occur when inserting new methods; they discard previously compiled code, beware!

Resources

Code available here: https://github.com/serenity4/JuliaOptimizationDays2024

Metaprogramming documentation: https://docs.julialang.org/en/v1/manual/metaprogramming

PrecompileTools.jl official documentation: https://julialang.github.io/PrecompileTools.jl

SnoopCompile.jl tutorial on invalidations:
https://timholy.github.io/SnoopCompile.jl/stable/tutorials/invalidations

GitHub/Discourse: **@serenity4**
Zulip/Slack: **Cédric Belmant**
Email: **cedric.bel@hotmail.fr**

Thank you!