

# Zlepšení RIR Bytecode překladače a interpretu

Jan Ječmen

2017

# Přehled

- Představení jazyka R, projektu RIR
- Změny provedené v RIR
- Dosažené výsledky

# Jazyk R

- Velmi dynamický jazyk
- Rostoucí popularita
- Vlastnosti
  - Imperativní styl
  - Funkcionální styl
  - Vektorové operace
  - OOP
  - Lazy vyhodnocování
  - Introspekce
  - ...

# GNU R

- Referenční implementace R
- REPL
- Interpret AST
- Možnost JIT překladu do bytecode
  - Kompilátor (napsaný v R)
  - Interpret (zásobníkový stroj)

# RIR

- Výzkumný projekt prof. Vitka
- Alternativa ke GNU R bytecode
- Cílem je umožnit statickou analýzu a optimalizace
  - Bytecode navržen s ohledem na analýzy
  - Framework pro abstraktní interpretaci RIR bytecode

# Změny v RIR

- Rozšíření bytecode
- Úpravy kompilátoru
- Změny interpretu bytecode

# Změny v RIR bytecode

- GNU R bytecode instrukce vs. RIR volání AST interpretu
- Inlining vestavěných funkcí
- Princip: optimalizovaná cesta pro obvyklý případ
- Funkce: builtin / special / closures

# Změny v RIR kompilátoru

- Optimalizace kontextů cyklů

```
f <- function(x) {  
  while (T) {  
    foo(break);  
  }  
}
```

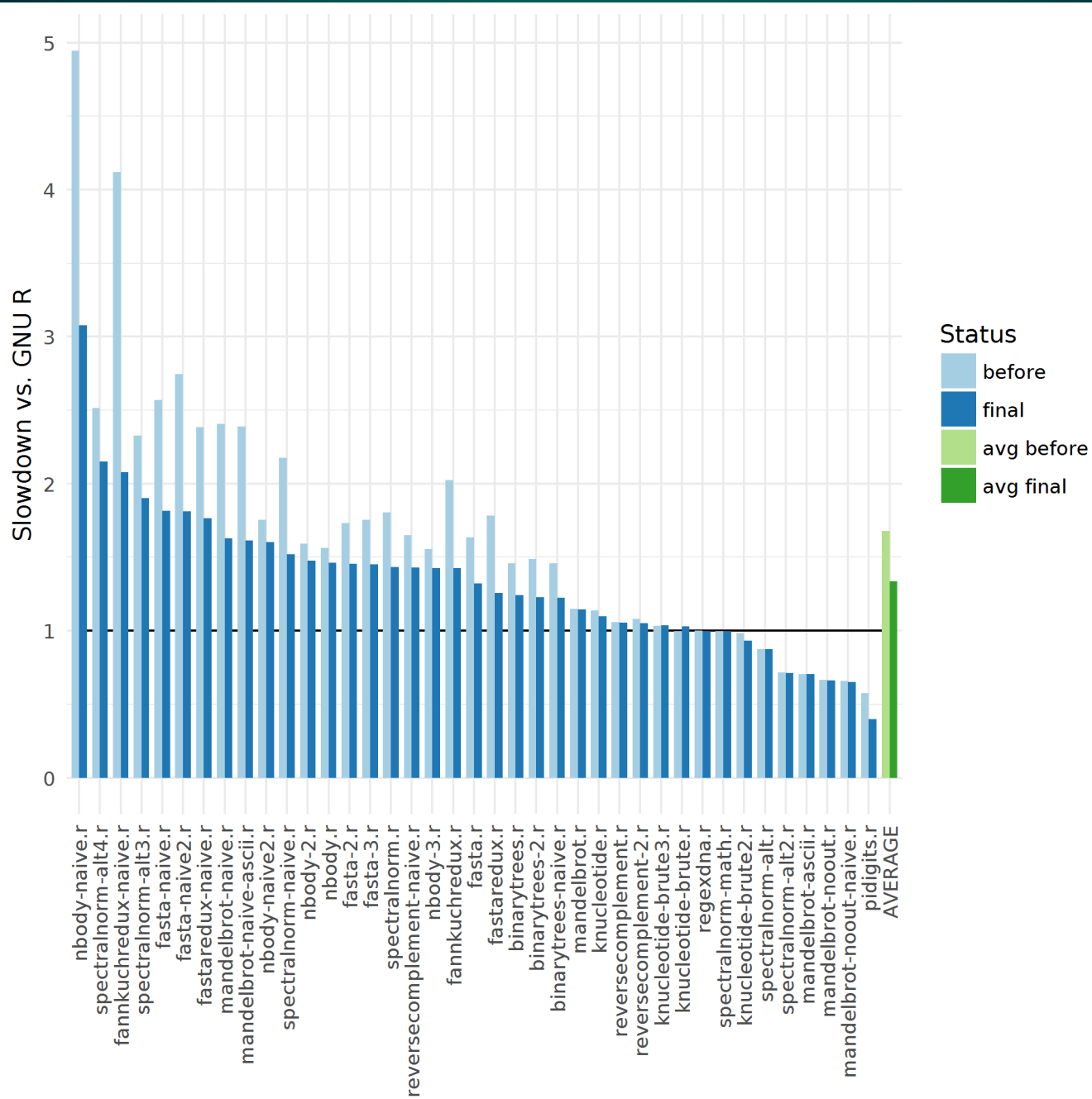


# Změny v RIR interpretu

- Refaktorování hlavní smyčky interpretu
- Ruční inlining bytecode instrukcí
- Threaded kód

# Výsledky

- Shootout benchmarky
- Největší zrychlení
  - Naivní verze
  - Tráví většinu času v bytecode
  - Minimum volání funkcí R
- Beze změny
  - Optimalizované verze
  - Bytecode jen zprostředkovává volání do R



# Závěr

- Implementovány změny v RIR
- Aplikované v současné verzi
- Náskok GNU R snížen o 50 %

Děkuji za pozornost

# Otázky oponenta

- 1) Další benchmarky kromě Shootout
- 2) Jak optimalizovat RIR dál?