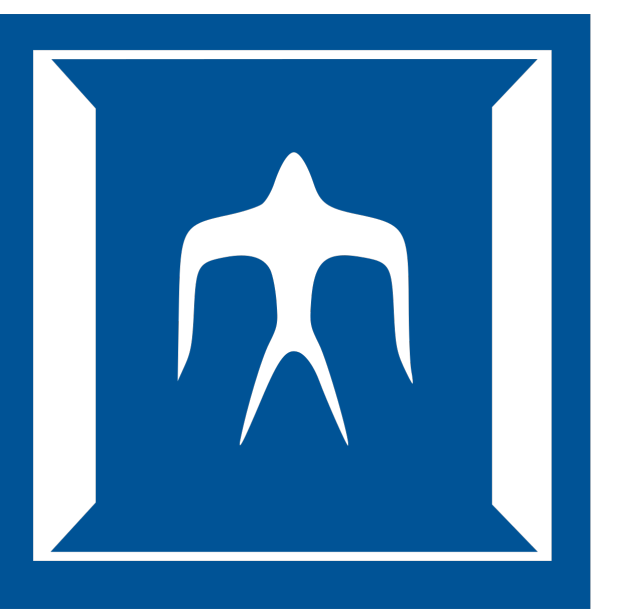




# Making Different JIT Compilations Dancing to the Same Tune, Acting in the Meta-level



Yusuke Izawa and Hidehiko Masuhara

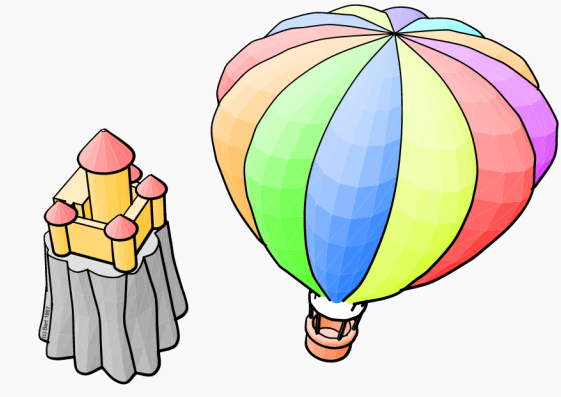
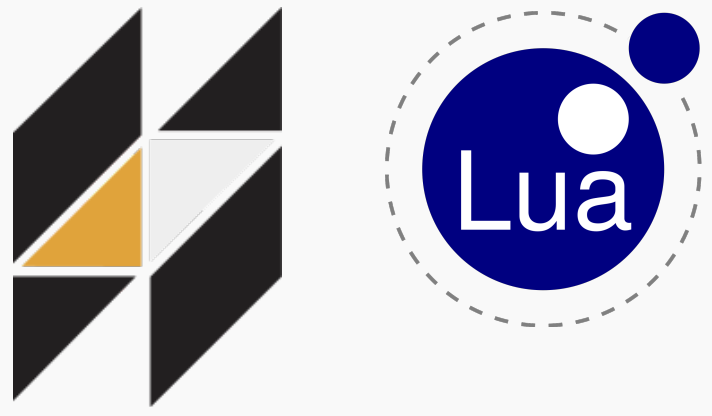
izawa@prg.is.titech.ac.jp, masuhara@is.titech.ac.jp, Tokyo Institute of Technology

## Context JIT Compilation

Two major JIT strategies:

Trace-based compilation

Method-based compilation

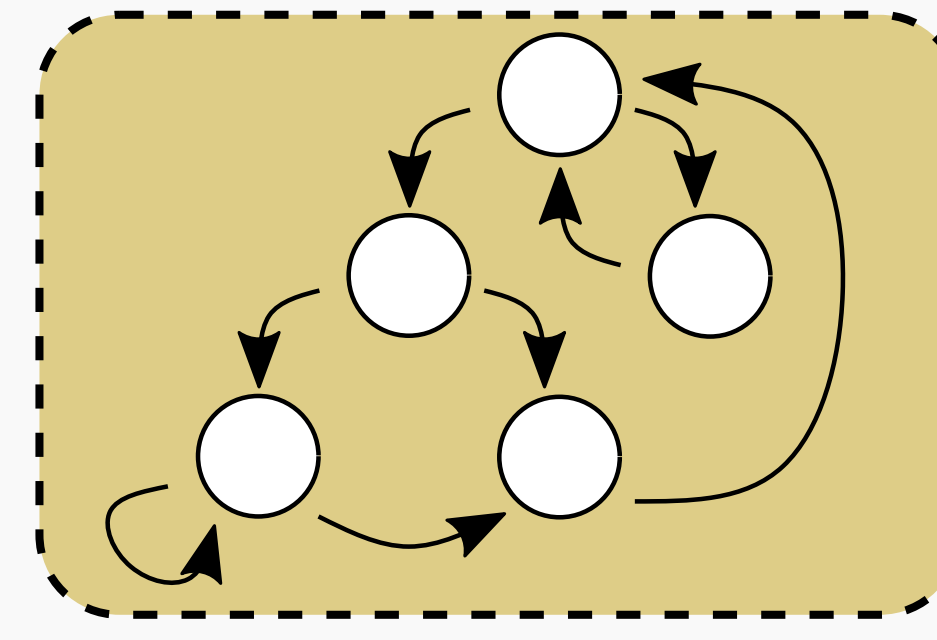
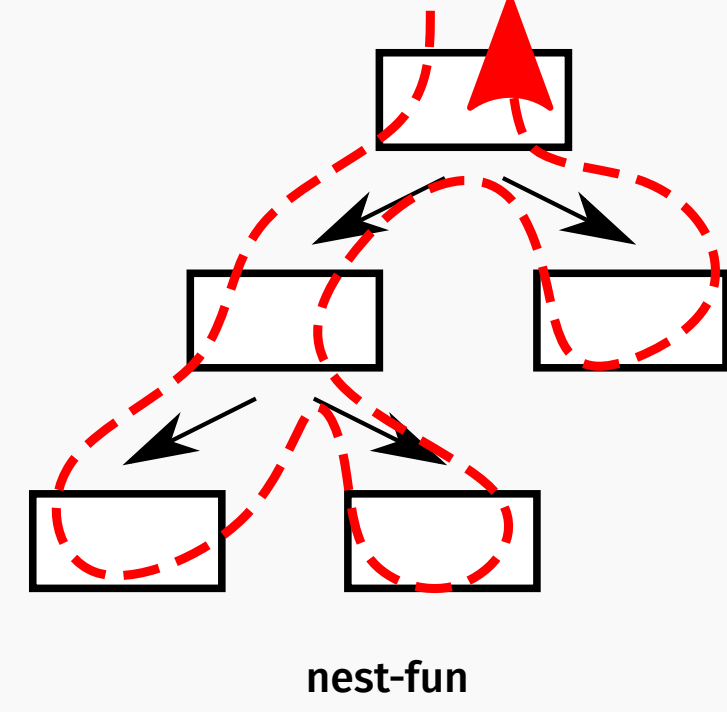


## Problem Trade-offs between the two

trace-based  
- inline automatically

method-based  
- able to apply any programs

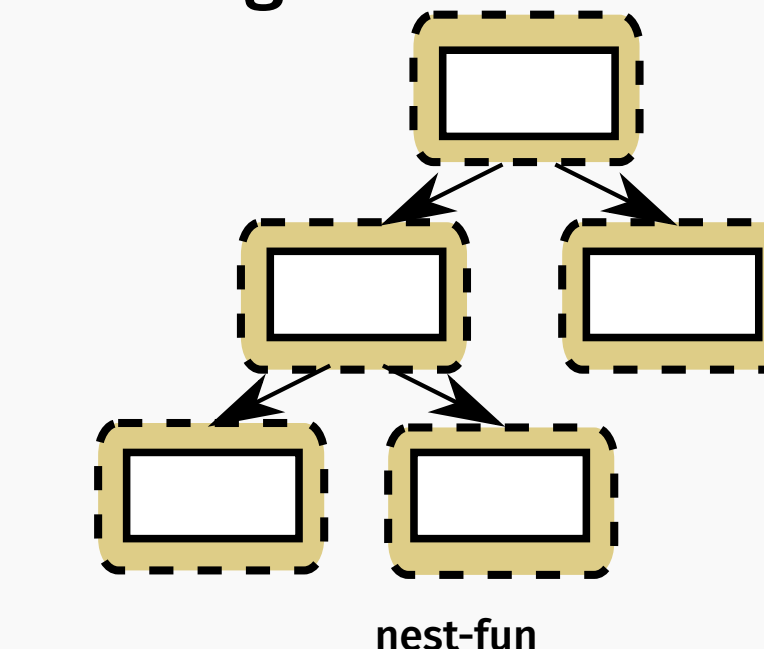
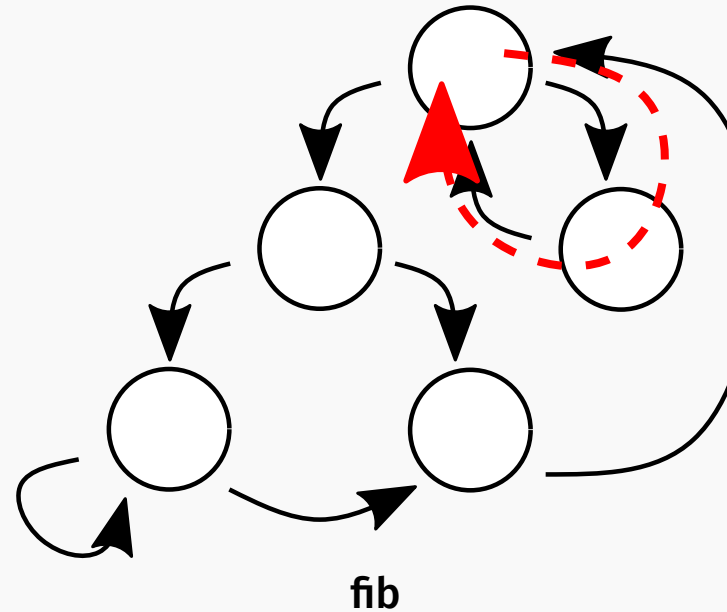
Good



- path-divergence problem[1]

- need carefully management for inlining

Bad



## Our Approach

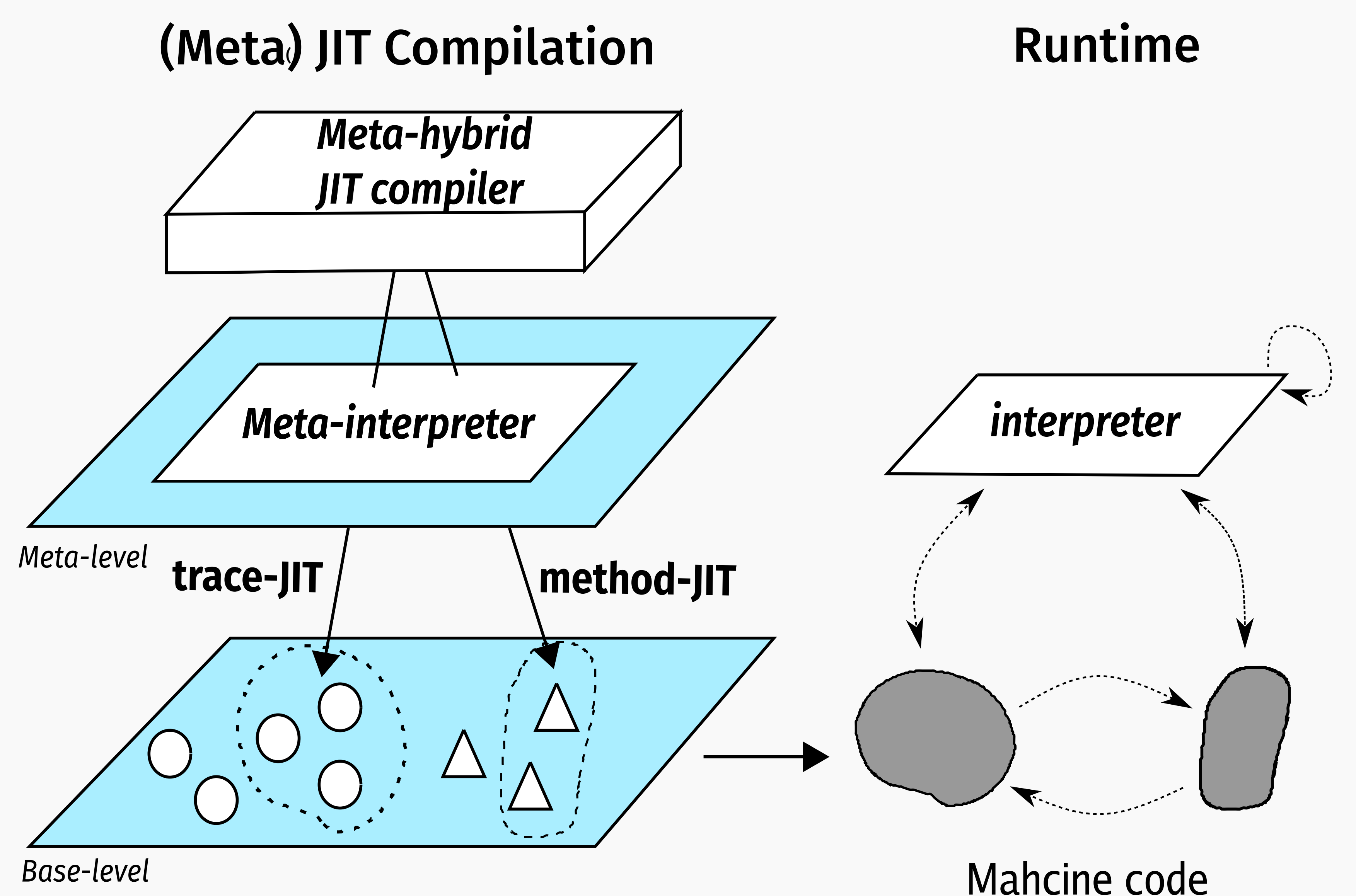
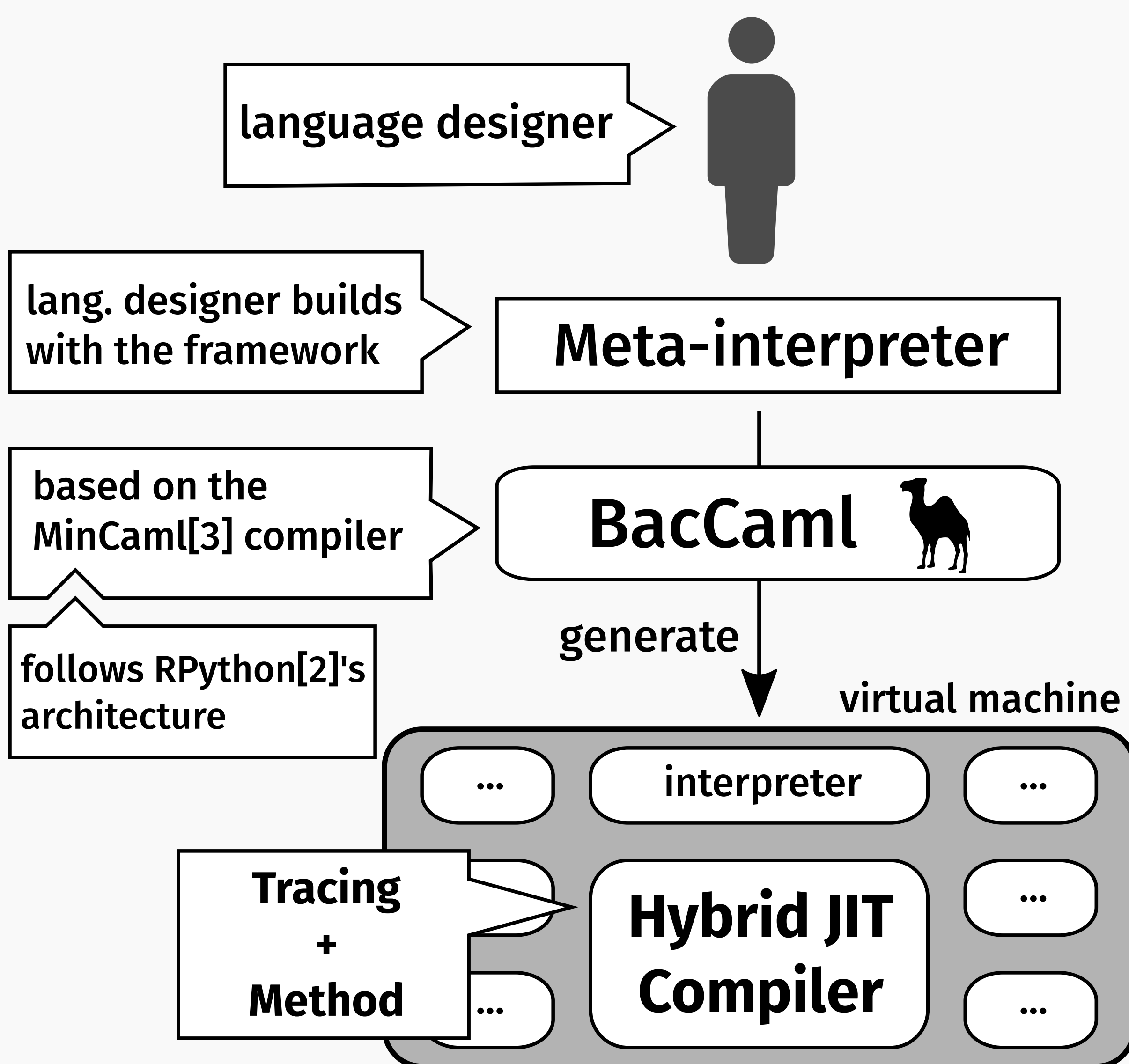
(1) Apply different strategies to different area

(2) Realize as a "meta-JIT compiler framework"

## Proposal Meta-hybrid JIT Compiler Framework

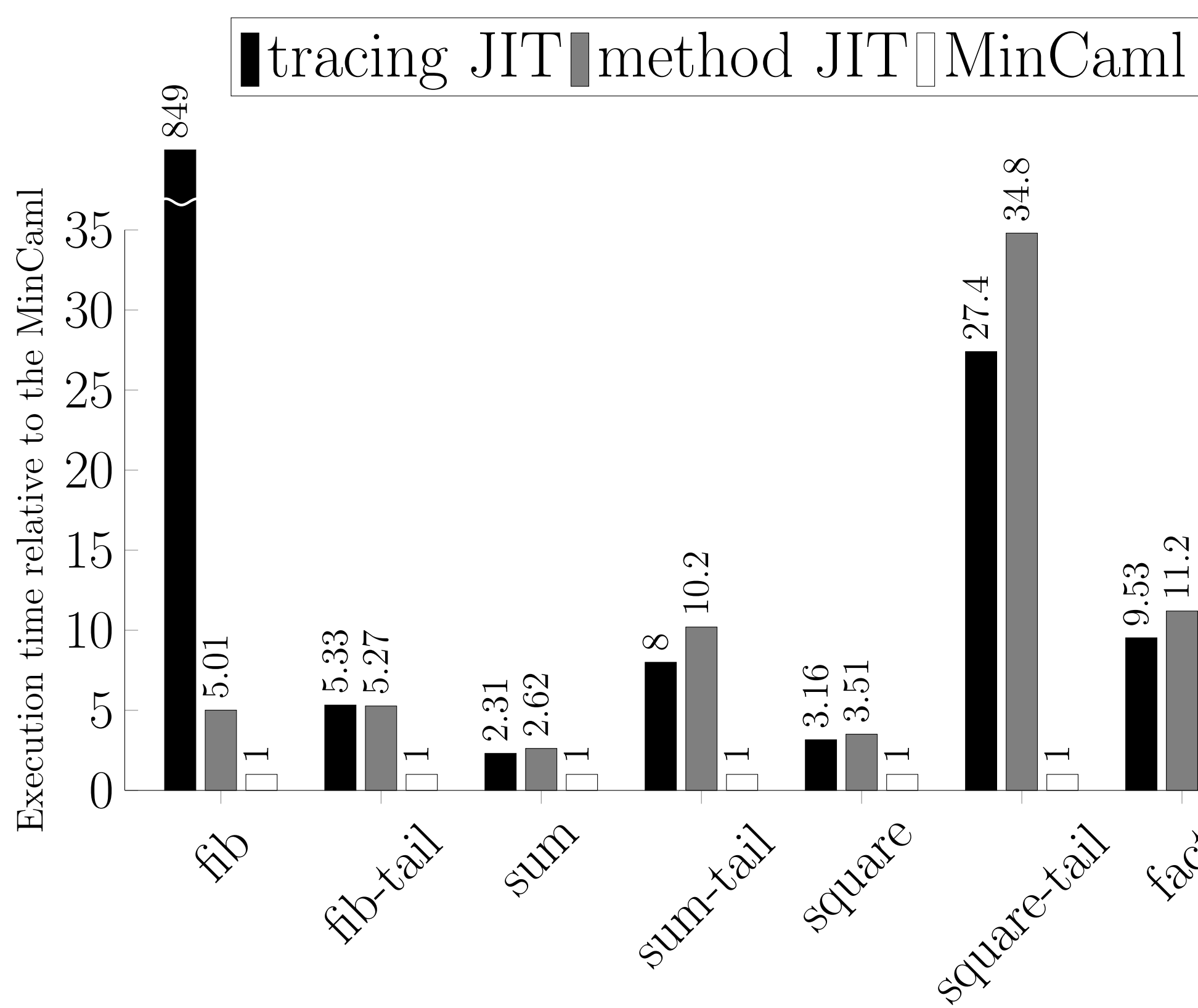
Virtual Machine Generation

Execution

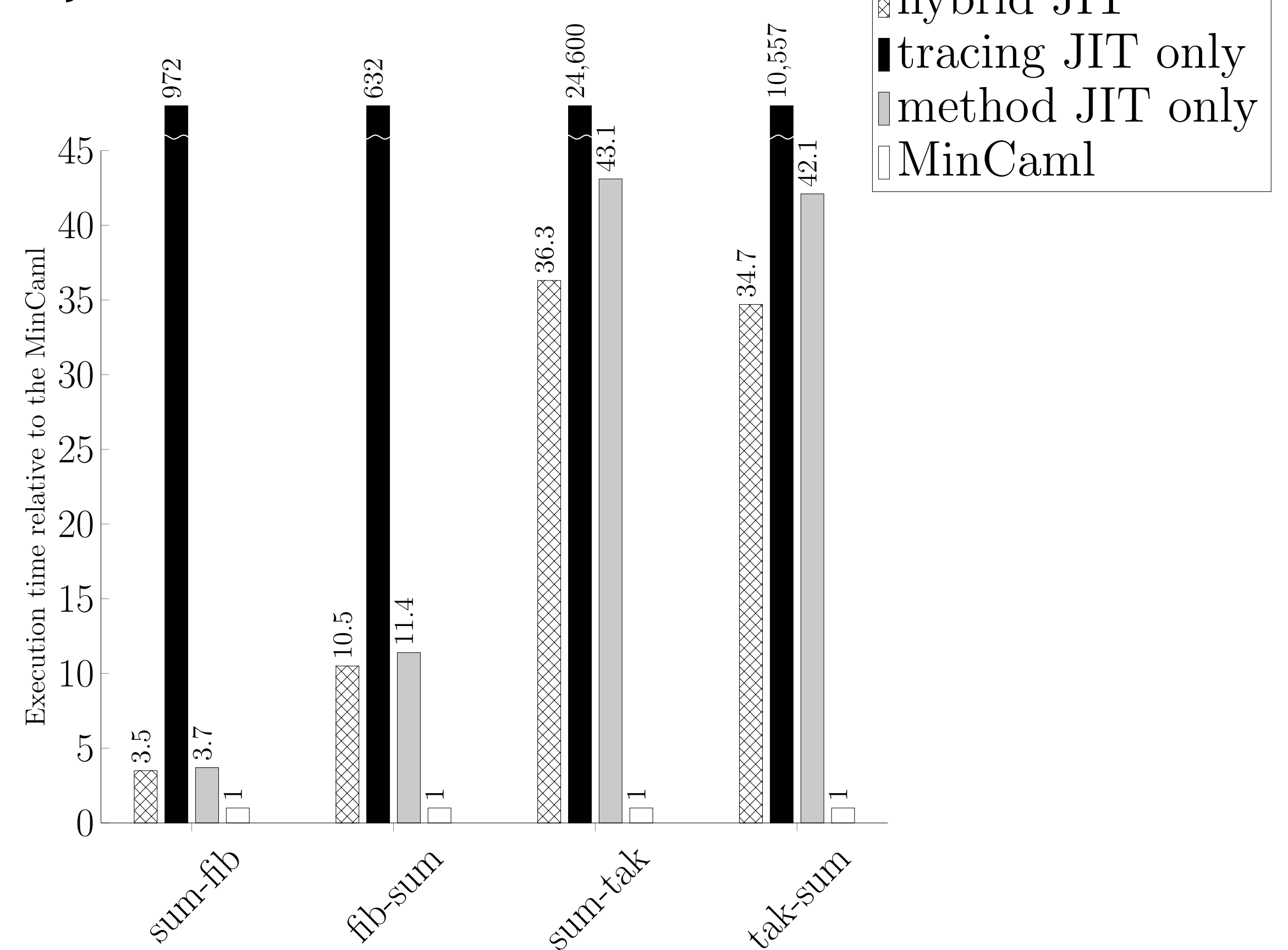


## Result

Standalone JIT Microbenchmark:



Hybrid JIT Microbenchmark:



## Future work

- Add optimizations for resulting traces
- Investigate profiling scheme for automatically selecting a suitable JIT strategy
- Apply our approach for RPython

## References

- [1] Huang Ruochen, Hidehiko Masuhara, and Tomoyuki Aotani. 2016. "Improving Sequential Performance of Erlang Based on a Meta-Tracing Just-In-Time Compiler." In International Symposium on Trends in Functional Programming, 44–58.
- [2] Bolz Carl Friedrich, Antonio Cuni, Maciej Fijałkowski, and Armin Rigo. 2009. "Tracing the Meta-Level: PyPy's Tracing JIT Compiler." In Proceedings of the 4th Workshop on the Implementation, Compilation, Optimization of Object-Oriented Languages and Programming Systems, 18–25.
- [3] Sumii Eijiro. 2005. "MinCaml: A Simple and Efficient Compiler for a Minimal Functional Language." In Proceedings of the 2005 Workshop on Functional and Declarative Programming in Education, 27–38.