IBM PL DAY 2019

Revisiting Type-Awareness for Automatic Parallelization

Ziyang Xu, Sotiris Apostolakis, Greg Chan, Susan Tan,
*Simone Campanoni, David I. August

Princeton University, *Northwestern University

Liberty Research Group
Princeton University

The State-of-the-Art* Workflow of Automatic Parallelization

- Start from Intermediate Representation (LLVM IR)
- Deal with dependences (disprove, break, tolerate)
- Make parallelization plan (choose loops…)
- Generate parallel code/IR (MTCG)

* **Apostolakis et al., ASPLOS ’20; Johnson et al., PLDI ’12; Kim et al, CGO ’12; Tian et al., PLDI ’10; Mehrara et al., PLDI ’09**

An Eternal War Against Dependences

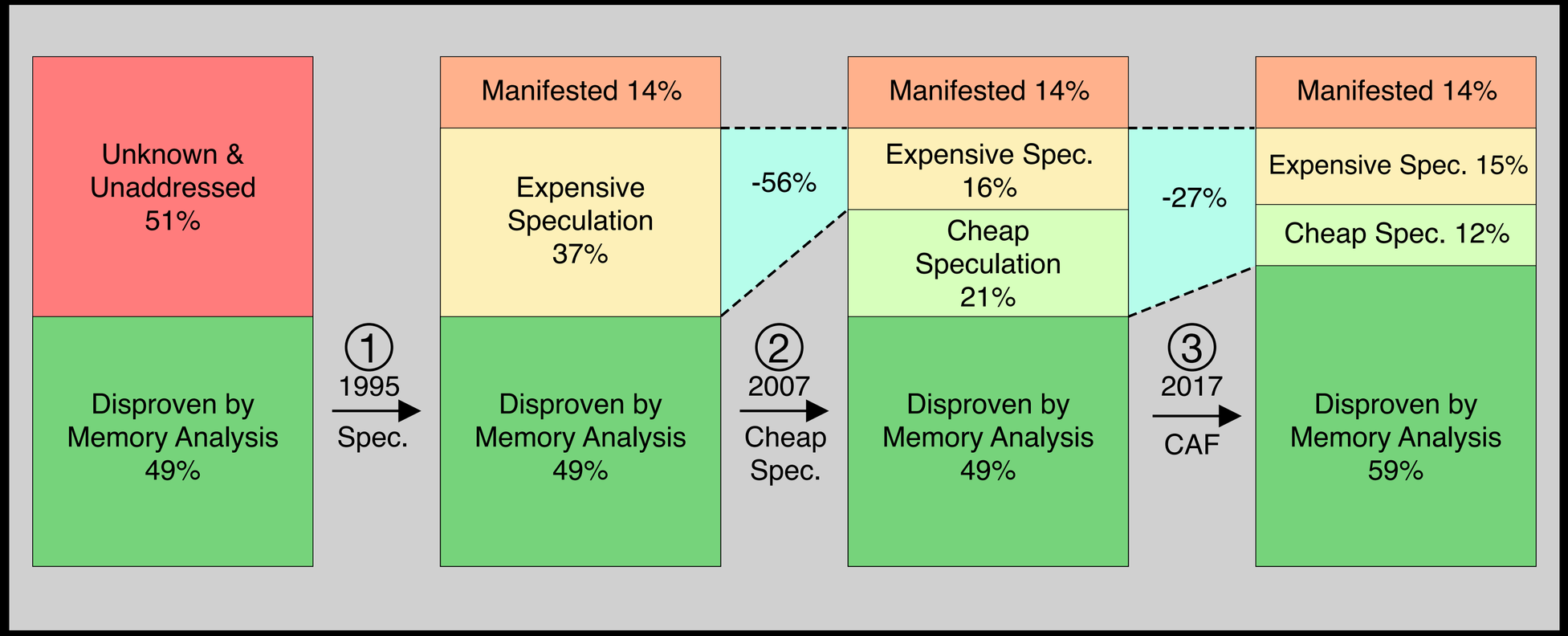
- To **Disprove** —> Analyses
- To **Break** —> Enabling Transformations
- To **Tolerate** —> Parallelization Techniques

An Eternal War Against Dependences

- To **Disprove** —> Analyses
- To **Break** —> Enabling Transformations

- To **Tolerate** —> Parallelization Techniques

History of Memory Dependence Handling (~2017)



History of memory dependence handling for automatic parallelization.
Percentages are relative to the total number of data flow dependences in hot loops of 14 C/C++ SPEC benchmarks.

How to Improve?

- Make better use of existing information
  - Make sensible decisions to reduce overheads (ASPLOS ’20*)
  - Combine profiling information with analyses’ information (Current Work)
- Discover more information from source code (This Work)

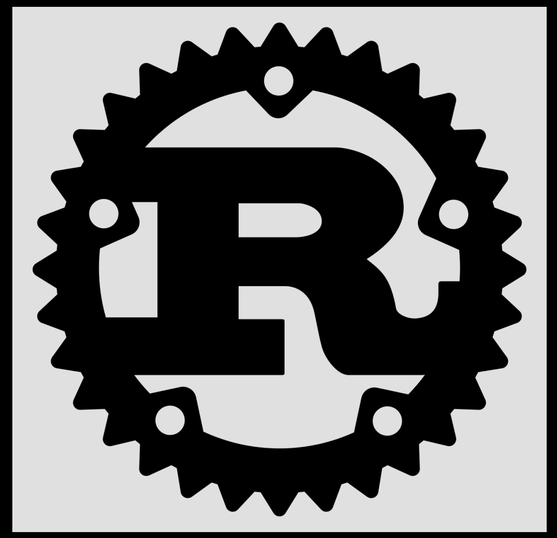
* S. Apostolakis, Z. Xu, G. Chan, S. Campanoni, and D. I. August, “Perspective: A Sensible Approach to Speculative Automatic Parallelization,” to appear in Proceedings of the 25th ASPLOS, 2020.

Motivation

- “Today, most high-performance compilers rely heavily on information gathered by the typechecker luring optimization and code-generation phases.”

- “Even compilers for languages without type system per se work hard to recover approximations to this typing information.”

- Types and Programming Languages
Benjamin C. Pierce



IBM Programming Languages Day 2018

Keynote:

Nicholas Matsakis (Mozilla Research), **Designing Rust**

Rust Type System & Alias Analysis

- Type-Safe
- Strong & Static
- Ownership and Borrowing

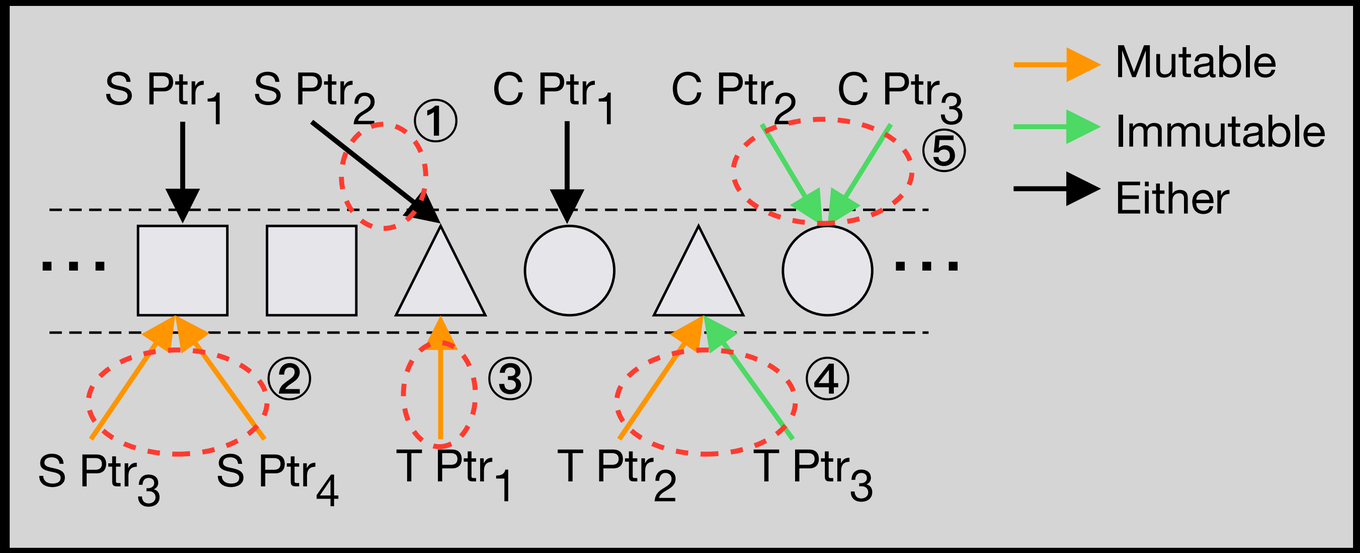
Strong + Type-Safe

=> ①: NO!

Ownership and Borrowing

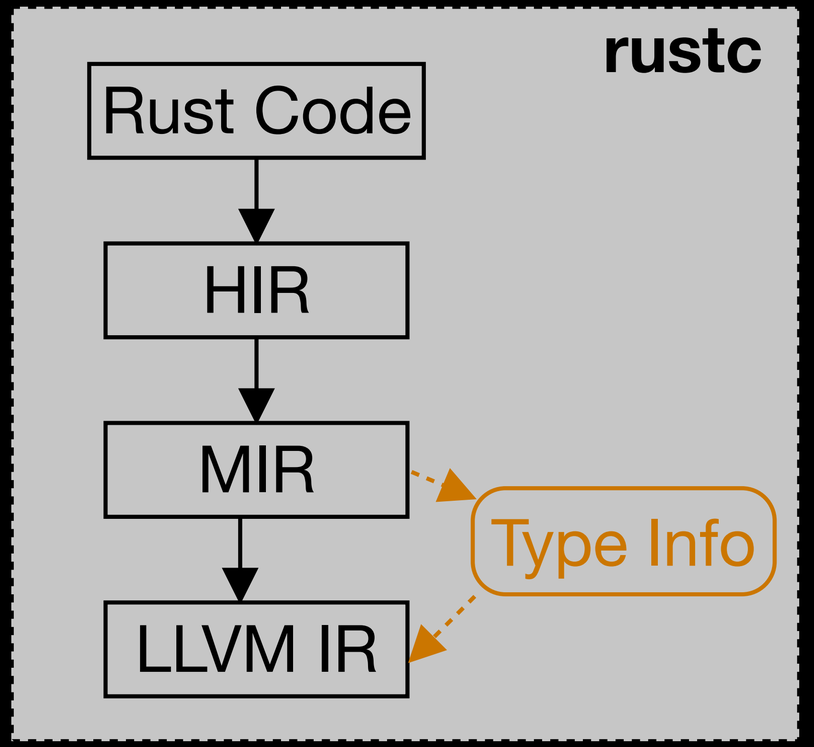
=> ② ④: NO!

③ ⑤: YES!



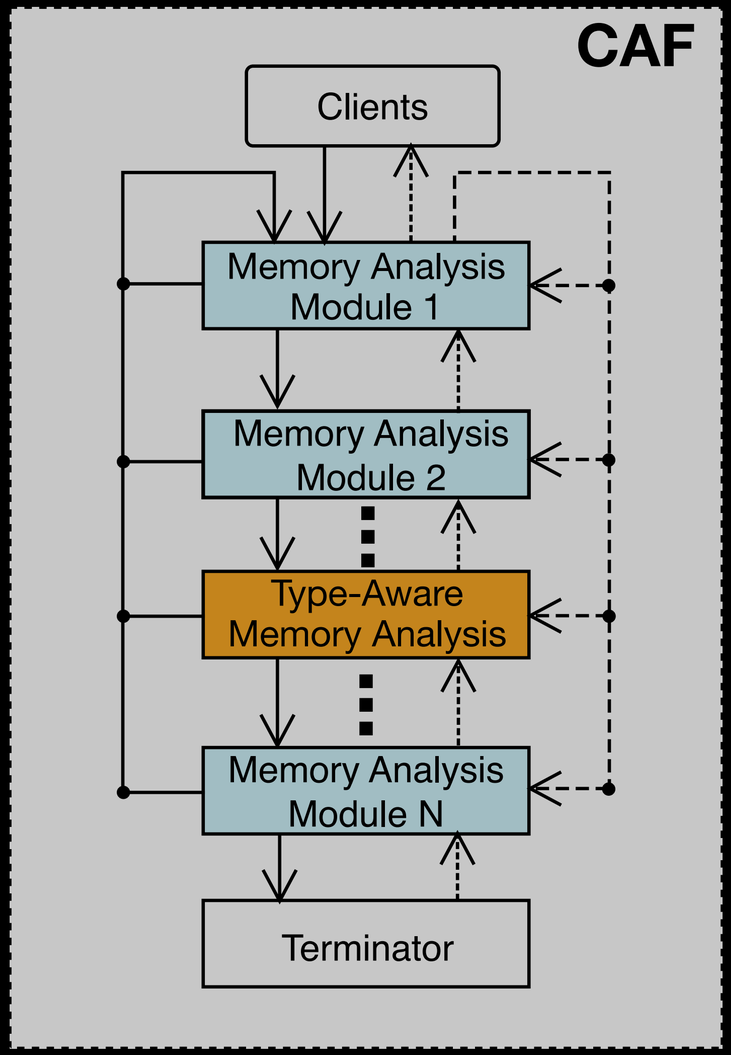
TAMA: Type-Aware Memory Analysis

- Emit type information in rustc
- Implement TAMA module in CAF
- Incorporate more Rust features
- Evaluate TAMA against Rust benchmarks



TAMA: Type-Aware Memory Analysis

- Emit type information in rustc
- Implement TAMA module in CAF*
- Incorporate more Rust features
- Evaluate TAMA against Rust benchmarks



* N. P. Johnson, J. Fix, S. R. Beard, T. Oh, T. B. Jablin, and D. I. August, "A collaborative dependence analysis framework," in CGO'17.

TAMA: Type-Aware Memory Analysis

- Emit type information in rustc
- Implement TAMA module in CAF
- Incorporate more Rust features
- Evaluate TAMA against Rust benchmarks

Peeking into the Future:
The War Against Dependences

- To **Disprove** —> Collaborative Analysis Framework Including TAMA ⬆️🔥
- To **Break** —> Enabling Transformations Collaborating with TAMA ⬆️🔥
- To **Tolerate** —> New Parallelization Techniques Inspired by TAMA ⬆️🔥

Wrap Everything in RAPPORT
A Research Automatic Parallelization Compiler

Thank you!
Questions?

Liberty Research Group
Princeton University