Zipr++: Exceptional Binary Rewriting

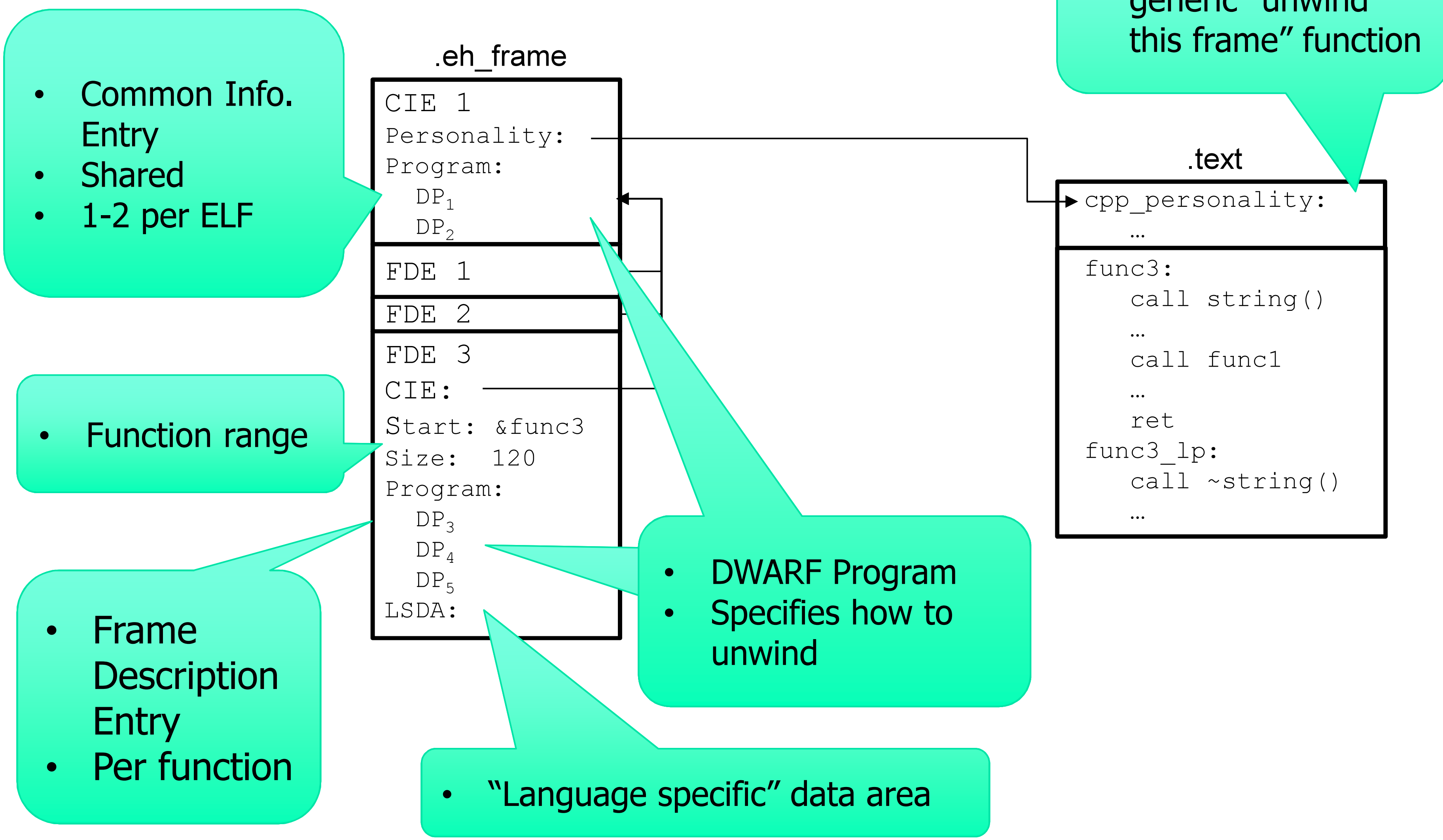
Jason Hiser, Anh Nguyen-Tuong,
William Hawkins, Matthew McGill,
Michele Co, Jack Davidson
University of Virginia

Motivation

Why do binary rewriters care about EH?

- Required by the x86-64 ABI
  - C++ supports exceptions, many programs use them
  - Rust, Ada exceptions essentially required
- More than just exception handling (EH)
  - Stack unwinding
  - `pthread_exit`
  - Profiling, debugging
- A “pin and win” strategy only goes so far
  - Limits efficiency (pinned calls)
  - Limits possible transforms (*e.g.*, stack frame size)
  - Limits randomization (pinned return addresses)

Unwinding with ELF



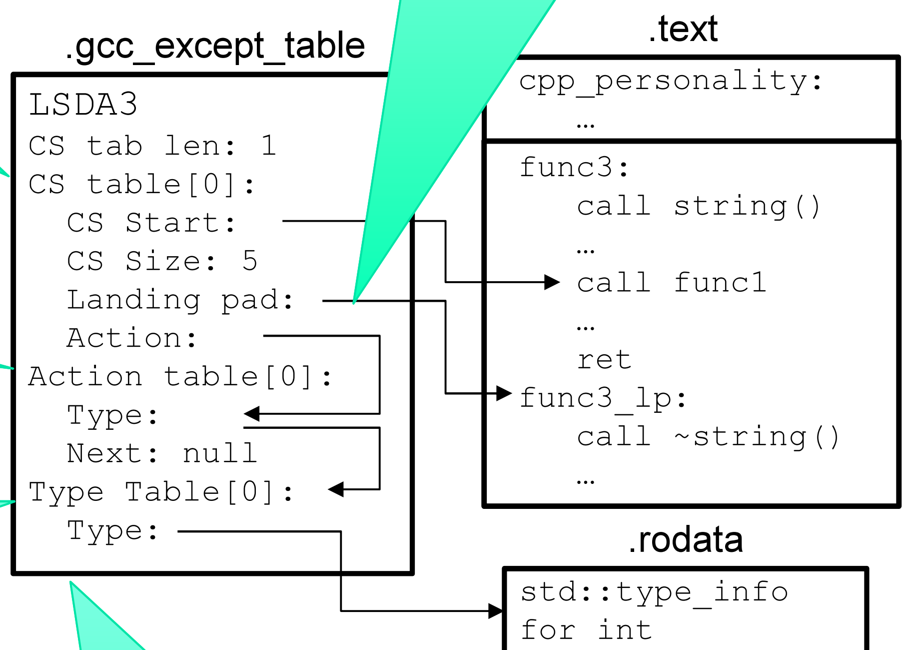
Catching Exceptions

- Landing Pad
- Per function cleanup (call destructors) and rethrow if not caught

- Call Site Table
- One entry per call

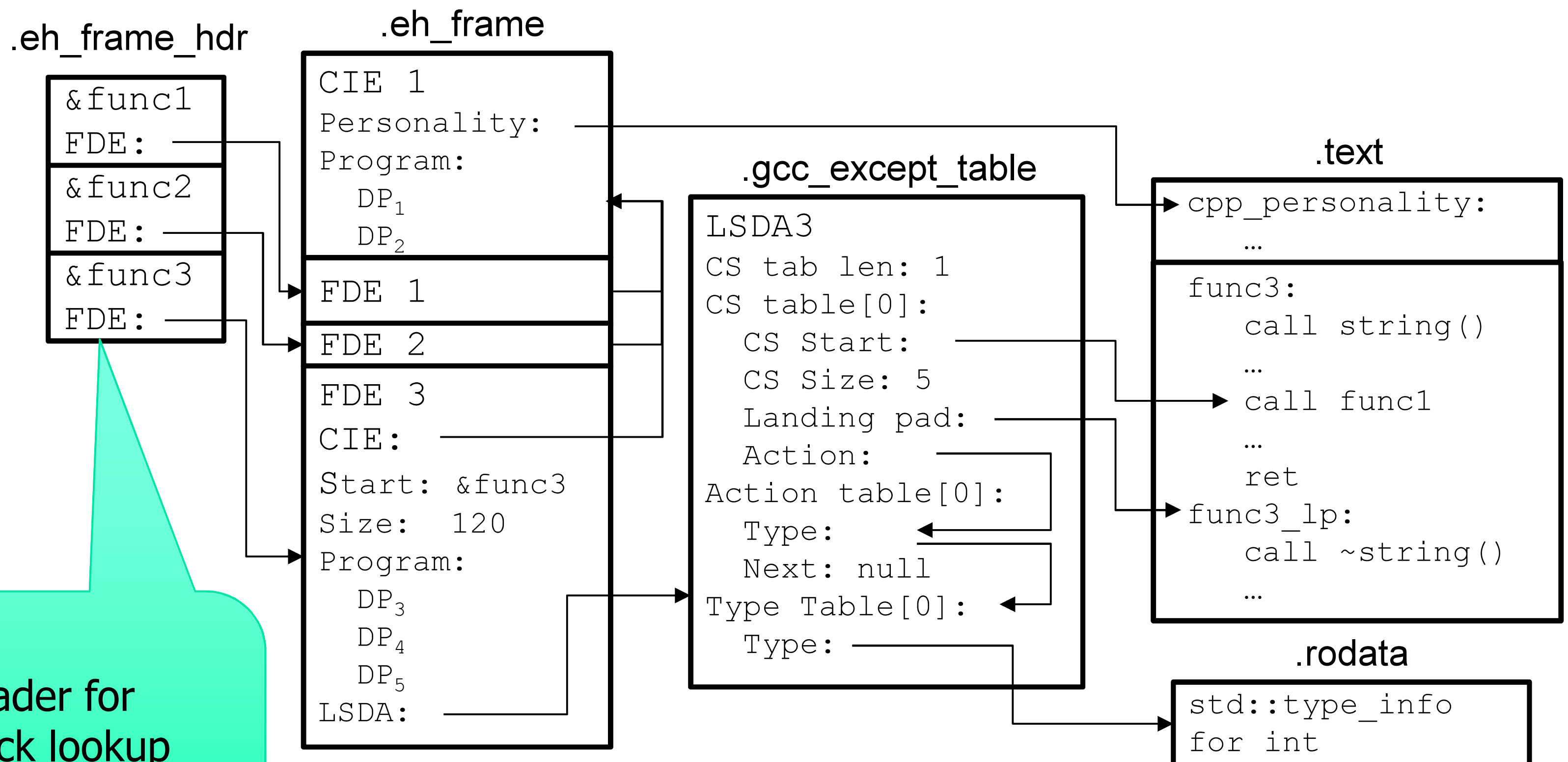
- Action Table
- One entry per "catch"

- Type Table
- Pointers to types "caught"



- "Language specific" data area

EH in ELF



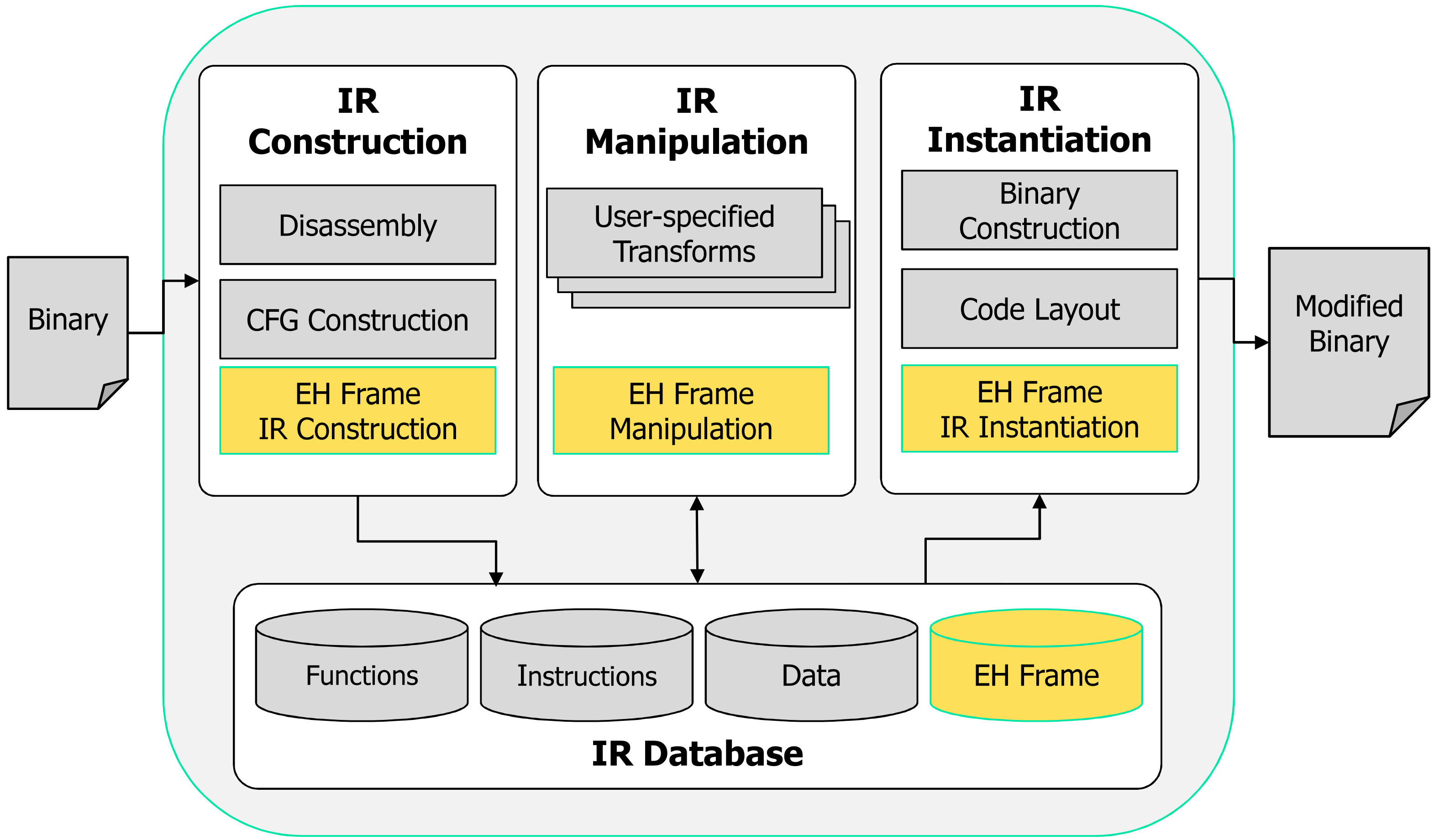
- Header for quick lookup via binary search

Modifying the EH info

- Modificati… …ace (*patching*)
  - Fi…s are …bly length-encoded and PC-r…tive ➔ …nall change requ… full table rewrite
  - …ange-based ➔ hard for p…struction edits
  - Plac… lots of burden on transforma…

Build IR instead!

Zipr++ Architecture



Modifying the EH info

- Step 1: Deconstruct to IR
  - Parse the EH info
  - Throw away all the encodings, ranges, etc.
  - Record essentials in the IR
    - The dwarf program with each instruction
    - Catch information for each call site

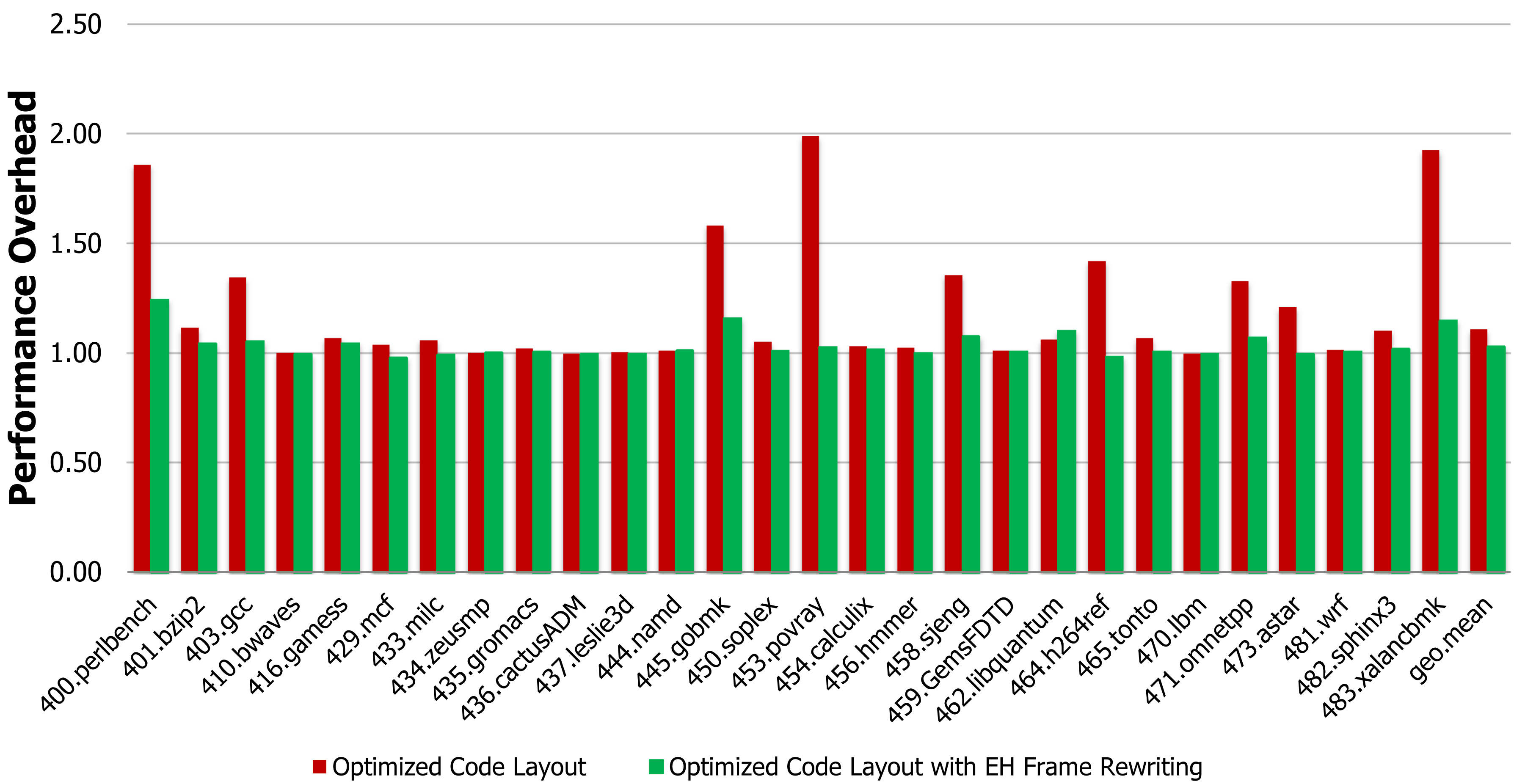
Modifying the EH info

- Step 2: Manipulate during transformation
  - C++ API to access EH IR
  - Most transforms can ignore completely
- Examples
  - Stack Layout Transform
    - Updates dwarf program to update location of return address relative to stack pointer
    - 1 C++ class, ~285 LOC
  - Stamp (`xor`) return addresses
    - Updates how to read the return address from the stack
    - 1 function, 72 LOC

Modifying the EH info

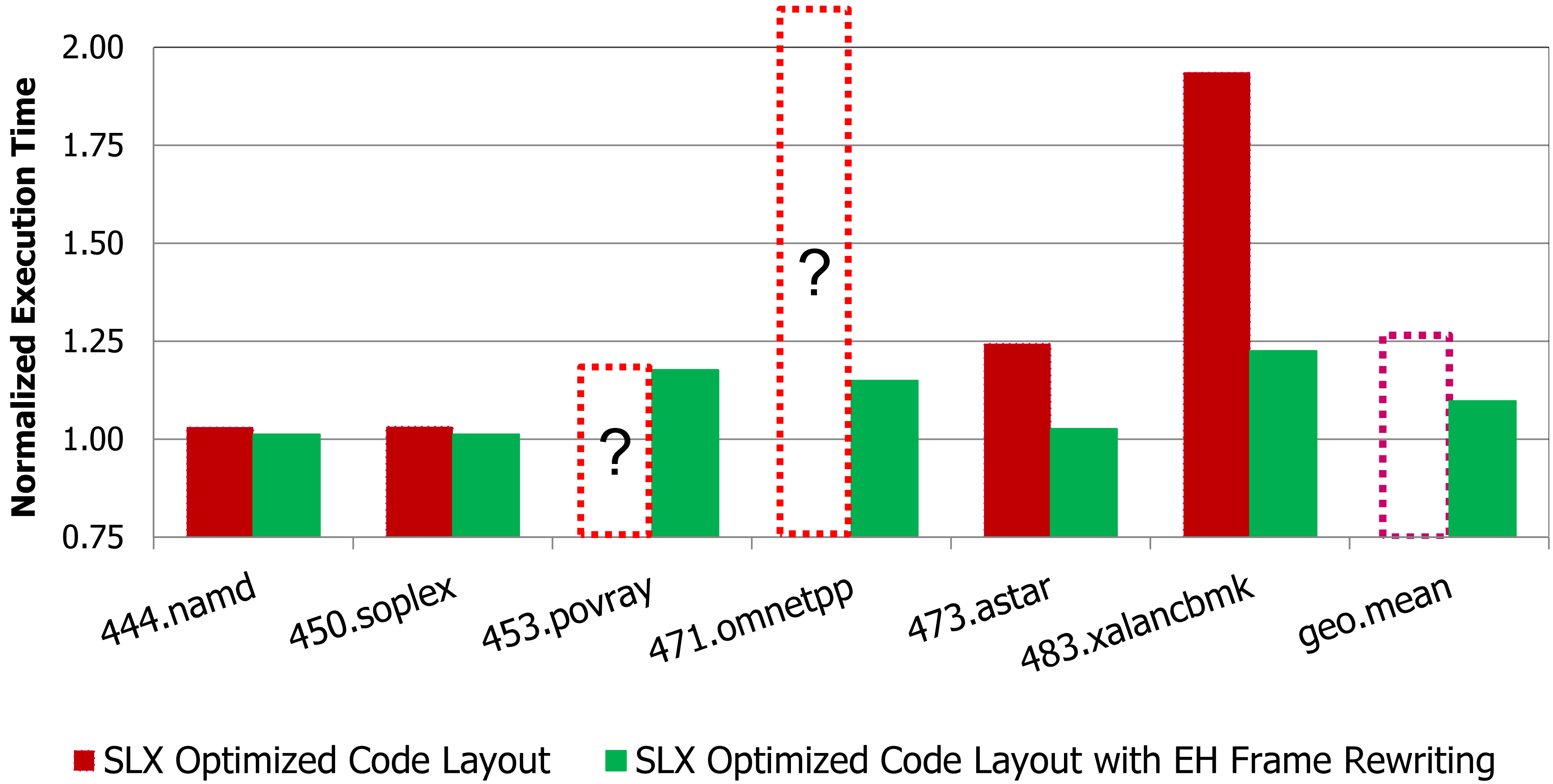
- Step 3: Reconstruct
  - Layout code (already done by Zipr)
  - Simple greedy scans code top to bottom
    - Extend current FDE or create new FDE as necessary
    - Re-use existing or create new CIE as necessary
    - Create an LSDA for each FDE if required
  - Generate/compress tables
    - Emit/generate assembly

Evaluation (Null Transform)



11% → 3%

Stack Padding



Related Work

- Static rewriters that require compiler support
  - ATOM, Diablo, etc.
- Other static rewriters without support EH
  - SecondWrite, UROBOROS, Ramblr, etc.
- Dynamic Rewriters
  - Strata, Pin, DynamoRIO

Summary

- Handling EH info is important
  - C++, Rust, Ada
  - Pthreads, debugging, profiling, performance
- Much EH information is about encoding or compression
  - Lesson Learned: Discard → simpler!
- Performance gains are real!