Shipping Software as LLVM IR

Birds of a Feather
LLVM Developer's Meeting 2016

Will Dietz, Vikram Adve

November 4, 2016

University of Illinois at Urbana-Champaign

Scope of Discussion

"Shipping Software as LLVM IR" broadly includes:

Any use case that distributes software in bitcode form

Usually this means delaying translation to native code.

Topics to avoid (mostly for lack of time):

Portability

Parallel extensions

Discussion Points

Overview of prepared discussion points:

- Use cases
- Building Commodity Software as IR
- Representations: Encoding and Distribution
- IR Semantics
- Debugging and Privacy

Additional Topics Encouraged!

`http://slido.com` using code #ShipIR

Prepared Discussion Topics

Use Cases

Do you ship software as IR today?

Have you in the past?

Would you like to do so in the future?

Please share a quick summary!

Note: Challenges are separate discussion topics

Please focus on goals and use-cases, technical challenges should have their own topic!

Building Commodity Software as IR

A number of related solutions exist publicly:

- -fembed-bitcode
- LTO, ThinLTO
- WLLVM

Discussion

- Pros/Cons
- Fat binaries?
- Future: A shared approach?

Representations: Encoding and Distribution

- Serialization, encoding: design space and trade-offs
  - Alternative storage formats (flat buffers)?
- Beyond the `llvm::Module`?
- How to best manage bitcode across versions?
- Distribution: techniques, challenges

IR Semantics

What are the semantics of the software?
How can they be preserved?

- Ensure execution matches behavior tested by developer
- As toolchain continuously evolves?
- Compatibility: Reading, Writing, Verifier
- Pipeline non-determinism?

Debugging and Privacy

Debugging

- How is debugging information managed?
- Mapping Crash Reports $\rightarrow$ Source

Privacy

- How can sensitive information be protected?
  - Trade-offs vs bitcode utility
  - Common infrastructure?

Discussion Time!

Open Discussion

http://slido.com using code #ShipIR

Thanks, folks!