Portable Executables

Executables that can be ported between ISAs and accelerator configurations

Linus Hagemann and Tom Wollnik

May 14th 2020

Important Terms

Executable

File with instructions a computer can execute

ISA

Instruction Set Architecture. Operations the CPU supports, e.g. ARM vs x86 vs PowerPC

Accelerator

GPU, Vector Calculation Unit, FPGA, ...

Context and Goals

Context

- Energy-aware computing
- Run a program on different ISAs or with different accelerators depending on the energy supply
- Program should be optimized for the ISA or accelerator configuration it runs on

Project Goals

- Build only one portable executable for a given program
- The portable executable can run efficiently on different ISAs or with different accelerators

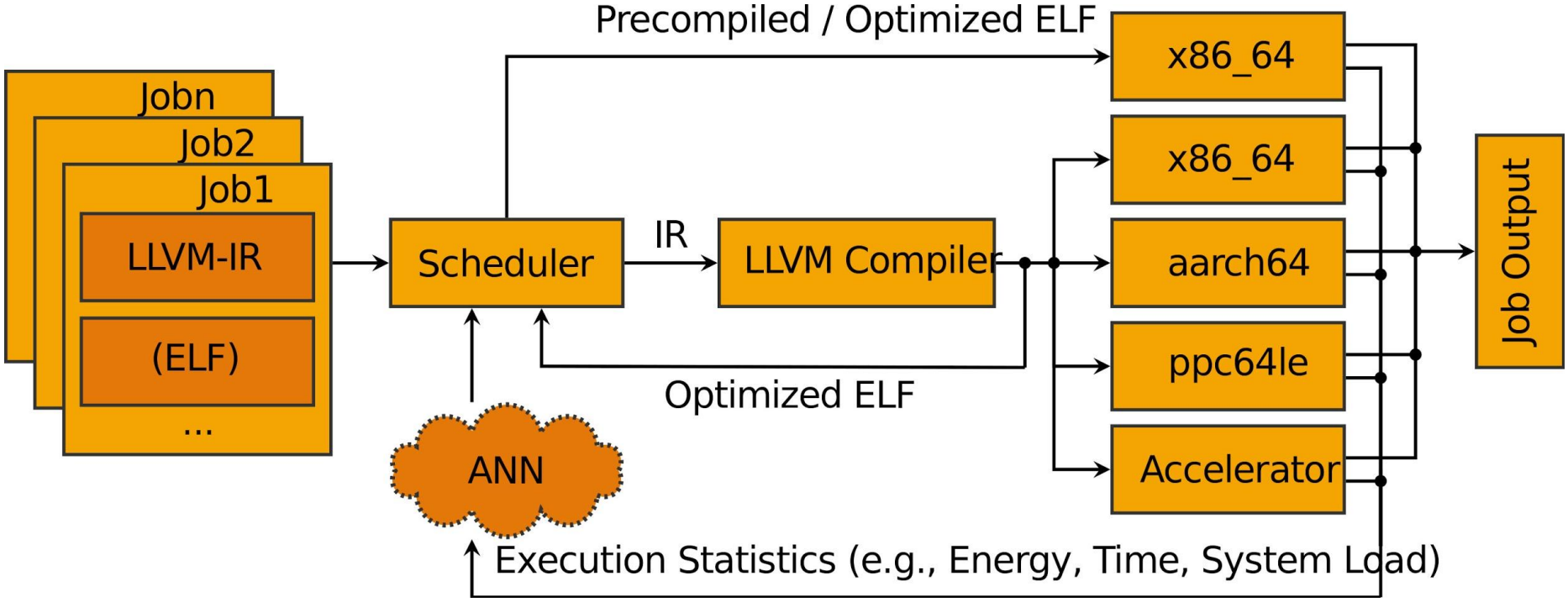
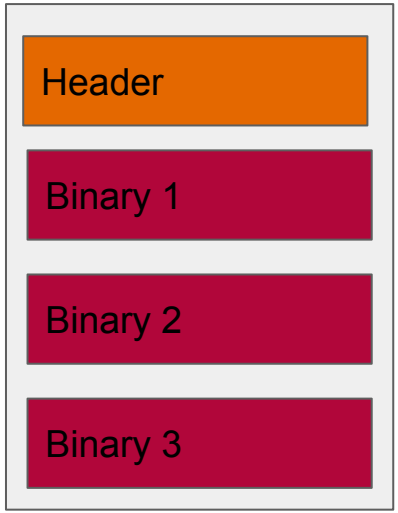


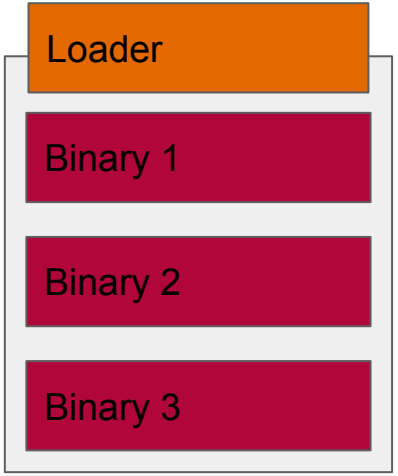
Image by Sven Köhler

Existing solutions

Fat Binaries



Version 1
Execute appropriate binary with OS support
(Apple Universal Binaries [1], FatELF [2])



Version 2
Execute a program that picks the appropriate binary at run time without OS support [3]

[1] https://web.archive.org/web/20050906001154/http://developer.apple.com/documentation/MacOSX/Conceptual/universal_binary/universal_binary.pdf
[2] https://icculus.org/fatelf/
[3] https://github.com/mpekatsoula/Fat-binaries

IBMi *PGM Objects

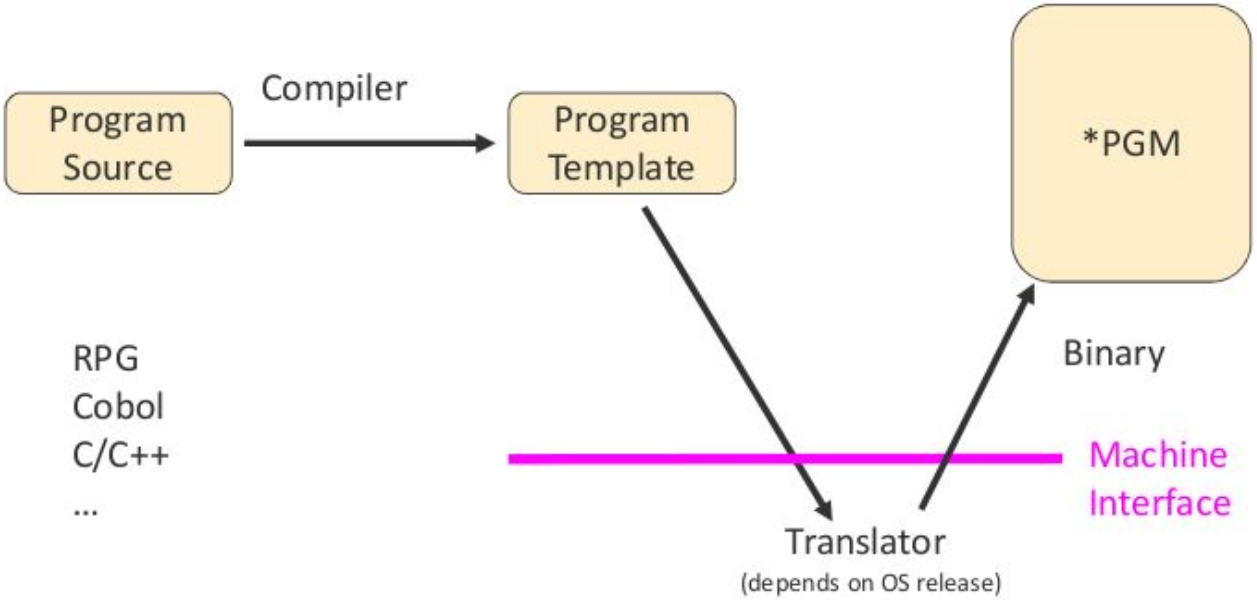


Fig. 1: compilation process

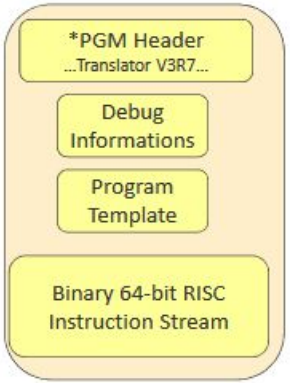


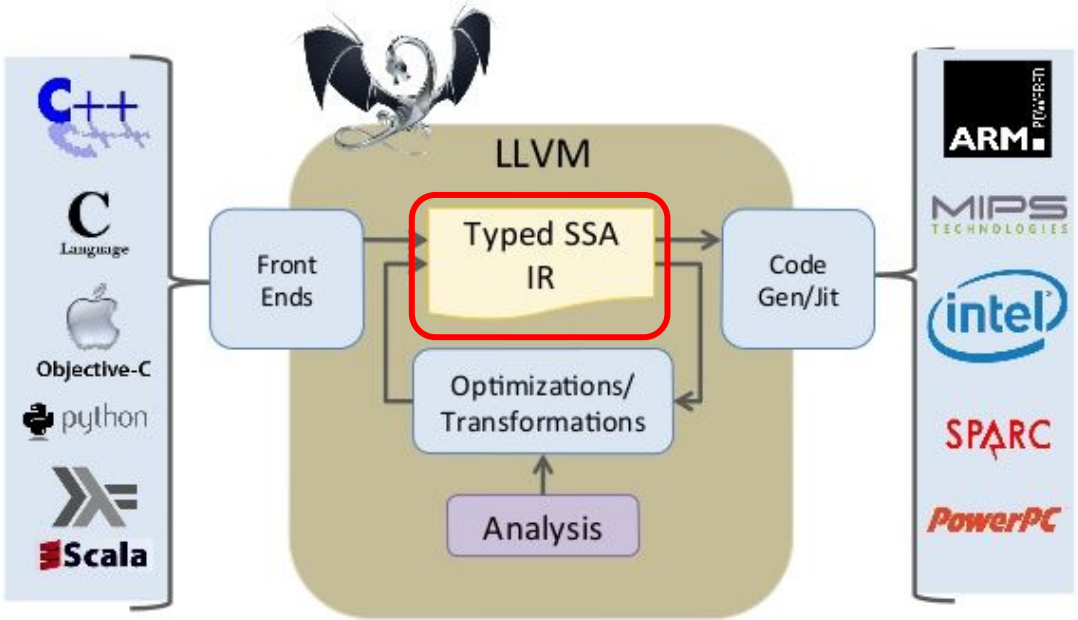
Fig. 2: *PGM objects

Our Idea

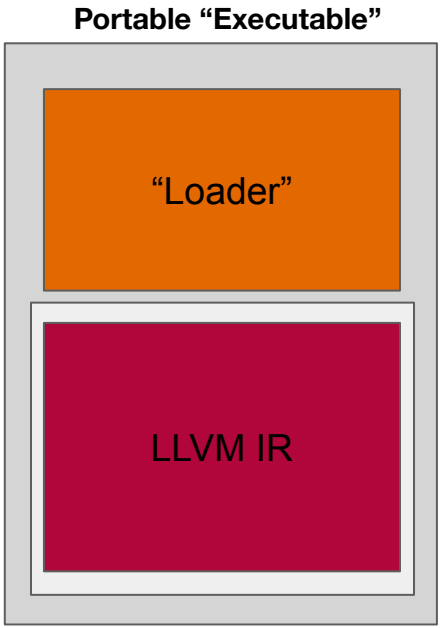
(Note: We didn’t actually come up with this approach. Credit goes to Sven Köhler and Andreas Grapentin.)

LLVM Compiler Infrastructure

[Lattner et al.]



Bringing it all together



Flow of Control

- “Loader” (scripting language) is executed
- collect information about the ISA and available accelerators for the current machine
- create an executable from the LLVM IR that is optimized for the current maschine
- run it

What?