facebook

facebook

Targeting HHVM's JIT compiler to LLVM

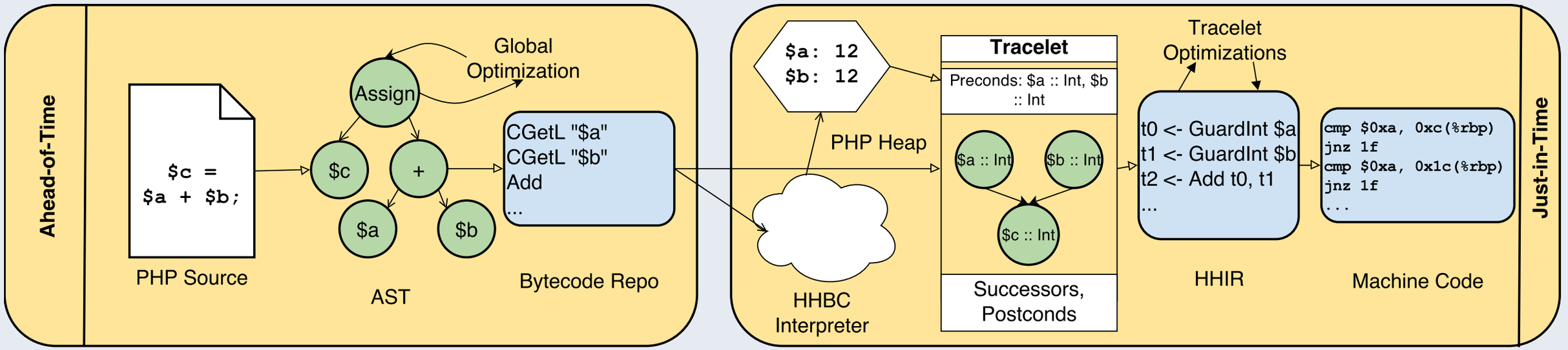
Jason Evans
Software Engineer
April 8, 2014

HHVM

HHVM Background

- 2009: HPHPc deployed (PHP→C++→x64)
- 2013: HHVM deployed
  - v1: PHP→HHBC→x64
  - v2: PHP→HHBC→HHIR→native (x64/aarch64)
  - v3: PHP→HHBC→HHIR→???→native
- Age-old question: Why not use LLVM?

HHVM fundamentals



- “Tracelet”-at-a-time JIT
- Observed types are burned in as preconditions
- Different input types → different tracelets
- Tracelet exits call JIT, smashed to call generated code
- Transition between JIT-generated code and interpreter at *any* HHBC instruction boundary

HHVM

Why not use LLVM?

- Then:
  - Big bet on very fast JIT codegen
  - Unladen Swallow was having a rough time
  - Added project uncertainty
    - LLVM didn’t solve the hard problems for us
- Now:
  - Too slow at codegen?
  - Integration headaches?

Current HHVM/LLVM challenges

- Calling convention modularity
- HHVM currently reserves CPU registers (VmFp, VmSp)
- Code smashing requires special alignment
- LLVM (MCJIT) does too much
  - GDB integration hook conflict
  - RTDyldMemoryManager lobotomy
  - Disable eh_frame generation (already have master eh_frame)
  - Don’t need run-time linking/loading

HHVM

For more information

- http://hhvm.com
- https://github.com/facebook/hhvm

facebook