Interesting things with LLVM

Brainstorming session

Robert N. M. Watson
FreeBSD Toolchain Summit - 10 May 2010

UNIVERSITY OF CAMBRIDGE

New technology. New opportunities?

- LLVM = Low Level Virtual Machine
  - Portable intermediate representation (IR)
  - Link-time optimization (LTO)
  - Easily analyzed, instrumented, extended
- Lots of ideas floating around
  - Idle-time code reoptimization, static analysis, incremental recompilation, ...

LLVM internal representation (IR)

- Native code generation from portable virtual machine instruction set ("bitcode")
  - Low-level: RISCish (with casts, unwind)
  - Includes type annotations (cast instr)
- Static analysis, optimization (especially inter-procedural, inter-language), transforms, ...

Sorts of things we might think about...

- C language extensions (K, Blocks)
- Static analysis aware of our conventions/quirks
- Language-aware instrumentation for invariants, debugging, DTrace probe insertion, locking asserts..
- Statically testable language and bitcode subsets ("driver subset","BPF-like","TCB")
- Fat binaries: default target, bitcode, locally reoptimized strategically, runtime rewriting and even live kernel/process rewriting
- Your ideas here…

Risks and problems

- When is it OK to depend on a feature specific to clang/LLVM/...?
  - Ever?
  - Certain architectures?
  - Certain OS features?
  - Debugging only?
- What about other tools: Coverity, fxr, ...?

Brainstorming...