LLVM: 10 Short Years Since 1.0

Vikram Adve

LLVM Compiler Research Group

Professor, Computer Science Department
University of Illinois at Urbana-Champaign

• Heritage
• Research
• Impact
• Future
1. Separation of Concerns [Auslander & Hopkins, CC82]
   - Architecture of “The PL.8 compiler”
   - Each pass focuses on one task; leave “cleanup” to other passes
   - Assumes: strong register allocation; strong (global) optimizations
   - Flexible pass reordering
   - Few corner cases

2. SSA [Cytron et al., TOPLAS’91]; SSA-based optzns

3. Mid-level IR + Machine IR: between SGI and then-GCC

4. Link-time cross-module opt. [Ayers et al., PLDI 1998]
5. Pattern matching ISEL, initially BURG [Fraser, PLDI '91], later DAG ISEL

6. Linear-scan reg. alloc. [Poletto & Sarkar, TOPLAS '99]

7. (Non-adaptive) JIT compilation [Höelzle et al.; others]

... And many more ...
Non-Heritage: Ideas Not Adopted

- UNCOL [1958], ANDF [1991]
- Low-level IR [RTL: Davidson & Fraser, adopted by GCC]
- Bit-vector dataflow analysis [long history]
- Graph coloring reg. alloc [Chaitin; Briggs]
- Compile-time interprocedural compilation (with caching) [Hall & Kennedy, Burke and Torczon]
- Adaptive JIT optimization [Self: Chambers & Ungar]

Big fail: Portability! LLVM did not try to solve this
Research Goals

Novel techniques for dynamic compilation [CGO’04]
- Flexible IR for dynamic optimization of C, C++,
- Division of labor between AOT and JIT optimization
- “Lifelong compilation” for arbitrary languages
- Compile-time, link-time, install-time, load/run-time, idle-time

Language-independent compilation services
- Optimization, codegen, JIT management, exception handling, GC

Memory hierarchy optimization [PLDI’05, PLDI’07]
- Data Structure Analysis: Identifying pointer-based data structures
- Automatic Pool Allocation: Controlling data structure layout
# Impact of Lifelong Compilation

<table>
<thead>
<tr>
<th></th>
<th>Compile</th>
<th>Link</th>
<th>Install</th>
<th>Load/Run</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mac OpenGL</td>
<td>✔️</td>
<td></td>
<td></td>
<td>✔️</td>
<td>♫</td>
</tr>
<tr>
<td>XCode</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenCL, CUDA</td>
<td>✔️</td>
<td></td>
<td></td>
<td>✔️</td>
<td>♫</td>
</tr>
<tr>
<td>Renderscript</td>
<td>✔️</td>
<td></td>
<td></td>
<td>✔️</td>
<td>♫</td>
</tr>
<tr>
<td>PNaCl</td>
<td>✔️</td>
<td></td>
<td></td>
<td>✔️</td>
<td>♫</td>
</tr>
<tr>
<td>Research</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>
Impact of Language-Independent Services

Static Languages

- **Imperative:** C, C++, Obj-C, D, Fortran, Ada, Erlang
- **Functional:** Haskell, Ocaml, Pure

“Managed” Languages and Run-times

- JVM, .NET

Scripting Languages

- Python, Ruby, Javascript, ActionScript, Julia

Explicitly Parallel Languages

- CUDA, OpenCL, Renderscript, OpenMP
Static Languages:
C, C++, D, Objective C, Objective C++, Fortran, Ada, Erlang, Haskell, OCaml, Pure

Managed and Scripting Languages:
Javascript, ActionScript, Python, Ruby, Julia, .NET

GPU and Parallel Languages:
OpenGL, CUDA, OpenCL, Renderscript, OpenMP

Only GCC has comparable (or broader) reach, and only for static languages
“There’s one thing [Apple and Google] still have in common, one last piece of technological brilliance they freely share with one another.”

—Wired.com, July 2013

“The only thing better than a love letter from Wired is a long love letter from Wired!”
Notable Accomplishments

- First production JIT compiler for C-based languages
- Clang/LLVM have fully replaced GCC in XCode 5
- Used on both major mobile platforms: iOS and Android
- Most GPU compute languages (OpenCL, CUDA, Renderscript) use LLVM
- First complete C++-11x: language + library
- First ARM64 compiler in production (iPhone 5s)!
And some nice awards along the way!

**2010 SIGPLAN Programming Languages Software Award:**
“LLVM has had a dramatic impact on our field ... significant influence on academic research, not just in compilers but also other areas ...”

**2012 ACM Software System Award**
- Given to *one software system* worldwide every year.
- Recent winners include Eclipse, VMWare, Eiffel, Boyer-Moore, SSL

**2013 CS@Illinois Distinguished Alumni Achievement**
- “… graduates who have made professional and technical contributions that bring distinction to themselves, the department, and the University.”
Lots More To Come: Functional Improvements

• More complete Windows support
• More effective profile-guided optimization
• Improved usability, parallelization for LTO
• Improved autovectorization
• Improved debugging support
• State-of-the-art pointer analysis

…
New Domains and Directions

Any place compiler technology is used!

- Javascript in Web browsers
- Java in production and research (Hotspot, JikesRVM)
- Linux kernels and derivatives: Android, ChromeOS, ...
- Embedded systems
- Heterogeneous SoCs
- More IDEs: Xcode only works for MacHeads!

Ok, some of these sound a bit crazy.

But 10 years ago, who would have thought LLVM would completely replace GCC on all Apple systems?
Thank You!

- Developer community
- Broad user community (but get involved!)
- Critical early research funding: NSF, UIUC

Advice to junior faculty:
- Find students smarter than you.
- Sit back and enjoy the ride!