LDC: A Dragon on Mars

David Nadlinger (@klickverbot)
DConf 2013
LLVM-based D Compiler
Looking just at the core:

- »Compiler Infrastructure«
- x86, ARM, PPC, Mips, NVPTX, R600, …
- Advanced optimizer
- Apple, Intel, NVidia, AMD, Google, …
- Open Source! (BSD-like)
Status: Overview

- DMD 2.062
- Linux and OS X, x86/x86_64: Just works™
- Passes (almost) the full DMD/drunt ime/Phobos test suites
Status: Target Architectures

- x86/x86_64: Main target, solid support
- PPC64: Standard library compiles, does not pass all tests
- ARM: Painfully close to Hello World
Status: Target Operating Systems

- (GNU) Linux: Main development platform
- OS X: Stable (10.7+)
- Windows: x86/MinGW, x64/MSVC
- *BSD: Not regularly tested
- Android: Bionic specifics need work
- AIX, Haiku, ...
Status: Language support

- Exception chaining
- Inline assembler restrictions
- Multiple extern(C) declaration hacks
- LDC-specific pragmas/intrinsics
- SIMD
Status: DMD compatibility

- ldmd2: Drop-in replacement for DMD
- Supports -deps, ... (RDMD)
- -cov, -profile not available
- Goal for ABI, etc.: As compatible as reasonably possible
The chart compares the relative execution time of different tasks using different compilers:

- **regex-dna (RT)**
- **regex-dna (CT)**
- **D_Raytracing**
- **PFFT (SSE)**
- **DRegs**

The y-axis represents the relative execution time, with lower values indicating better performance. The chart uses three different compilers:

- **DMD 2.062**
- **LDC 0.11.0 beta**
- **GDC 4.8.0 (20130323)**
Thrift serialization benchmark

- binary write
- binary read
- compact write
- compact read
- json write
- json read

rel. execution time (lower is better)

- DMD 2.062
- LDC 0.11.0 beta
- GDC 4.8.0 (20130323)
Hypothesis 1:

Control is the unique offering of systems languages.
Hypothesis 1:  
**Control is the unique offering of systems languages.**

Hypothesis 2:  
**Performance requirements drive the demand for control.**
Hypothesis 1: 
Control is the unique offering of systems languages.

Hypothesis 2: 
Performance requirements drive the demand for control.

Hypothesis 3: 
Zero-cost abstractions matter.
module test;
int add(int a, int b) {
    return a + b;
}

define i32 @_D4test3addFiiZi(i32 %b_arg, i32 %a_arg) {
    entry:
    %tmp2 = add i32 %a_arg, %b_arg
    ret i32 %tmp2
}
LLVM IR

```c
define i32 @_D4test3addFiiZi(i32 %b_arg, i32 %a_arg) nounwind readnone {
  entry:
    %tmp2 = add i32 %a_arg, %b_arg
  ret i32 %tmp2
}
```

```c
_D4test3addFiiZi:
  add EDI, ESI
  mov EAX, EDI
  ret
```
The process starts with a D file (.d), which is then processed through the DMD frontend and the LDC "glue" to produce intermediate representation (LLVM IR).

The LLVM IR is then passed to the LLVM optimizer, which includes D-specific passes.

The optimized LLVM IR is then passed to the LLVM codegen, which generates the final executables (.exe) and shared libraries (.so).

The process also involves the use of external tools like GNU as for assembly generation (.s), and system linker and ar for linking (.o files).
DMD frontend

- Copy in LDC tree, merged on per-release basis
- Inliner, etc. not used
- Minimize changes to »shared« DMD source
DMD frontend

- Lack of documentation
- Implicit invariants often hard to track down
- Layering violations
- “Impedance mismatch”: LLVM IR strictly typed
LLVM: Advantages

- Easy to approach
- Defensively written code
- Effective debugging facilities (graphs, dump(), bugpoint, ...)
- Modular, easy to extend
- Active, helpful community
- Patches easy to upstream
LLVM: Challenges

• LLVM IR is target dependent (varargs, struct ABI, …)
• Liberal in C++ API changes
• No emulation for missing OS functionality
• Windows support not needed by big corporate clients
Source size

![Bar chart showing the source size in total lines of code.]

- **dmd2**: 114k
- **gen/ir**: 33k
- **driver**: 3.6k
- **misc**: 3.0k
Runtime and Tests

- druntime, Phobos, dmd-testsuite are Git submodules
- Pro: consistent state; Contra: merging harder
- druntime/Phobos changes: atomic operations, intrinsics (math/bitops), platform support, x86_64 varargs
Committers

- Tomas Lindquist Olsen
- Christian Kamm
- David Nadlinger
- Alexey Prokhin
- Kai Nacke
- Frits van Bommel
- Kelly Wilson
- Moritz Warning
- Benjamin Kramer
- Robert Clipsham
- Leandro Lucarella
- …
Development

• Source code, issue tracker: http://github.com/ldc-developers

• CI systems:
  • Travis (http://travis-ci.org, pull requests)
  • Lycus Foundation (http://ci.lycus.org, thanks Alex!)

• Forum/mailing list: http://forum.dlang.org

• Developer docs: http://wiki.dlang.org/LDC
Next steps

- Shared library support (DMD 2.063)
- Exception chaining
- x86_64 vararg ABI
- Minimize differences to DMD upstream source, translation to D (?)
Future directions

• Compiler performance work
• Leverage D attributes; further D-specific optimizations?
• Integration of LLVM-based tools
e.g.: AddressSanitizer/ThreadSanitizer
• Link-time optimization
• PNaCl/Emscripten/…
• Scripts for setting up cross toolchains
Summary

LDC …

… is a D2 compiler.

… is »ready« on Linux/OS X, Windows coming.

… produces significantly faster code than DMD.

… provides many promising opportunities.

… is Open Source, and easy to hack on!
LDC needs your help!

You could…

… fix bugs: tracker on GitHub, has »junior jobs«
… champion new platforms (ARM!)
… package LDC for your favorite distro (*Debian*)
… help keeping the wiki/docs up to date
… try LDC on your own projects (DustMite!)
Wish list

- OS X and Win32/MinGW: Need a build slave … Thanks, Brad!
- Idea: D compiler performance tracker
- “Real-world” benchmarks
http://wiki.dlang.org/LDC
http://forum.dlang.org
irc://irc.freenode.net/ldc

David Nadlinger
<david@klickverbot.at>
Backup slides
Status: Performance

Optimized micro-benchmarks:

- Consistently ~30% faster than DMD
- On par with GDC

Further potential:

- D-specific optimizations
- Leverage attributes (scope, pure, …)
regex-dna, The Computer Language Benchmarks Game

- Java SE 7
- Xpressive
- RE 2
- FReD (RT)
- V8
- FReD (CT)
Commit activity
ABI compatibility

- Shared libraries!
- Documentation severely lacking
- GDC/LDC use default EH/… mechanisms
- Implications for packaging tools