What's New in the LLVM Compiler

Chris Lattner
LLVM Chief Architect

Apple Compiler Evolution

- Renovating the Apple compiler landscape
  - Phasing out GCC-based tools
  - Driving innovation in LLVM-based technologies
- This talk focuses on LLVM technology

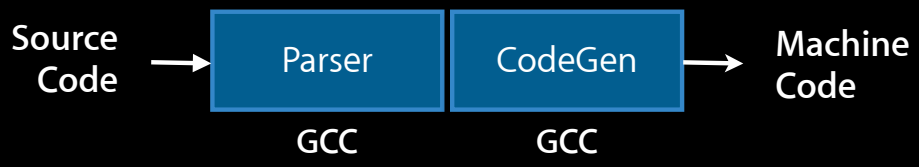
About the LLVM Project

- Focused on building great tools
- Radically rethink how compilers are built and used
  - Build compilers as a set of reusable libraries
  - Support new applications with shared components
- Open source!
  - http://llvm.org/

Compiler Landscape

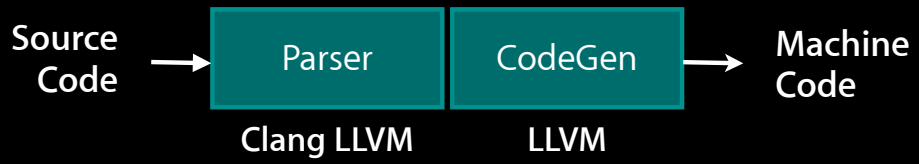
Three compilers

- GCC 4.2



- LLVM-GCC

- LLVM Compiler



Compiler Landscape

Two platforms

Compiler Landscape

Two releases

- Xcode 3.2.3
- Xcode 4 Developer Preview

LLVM in Xcode 3.2.3

New in Xcode 3.2.3

Updated LLVM-GCC and LLVM Compiler

- iPhone OS support
- Major enhancements for Mac OS X

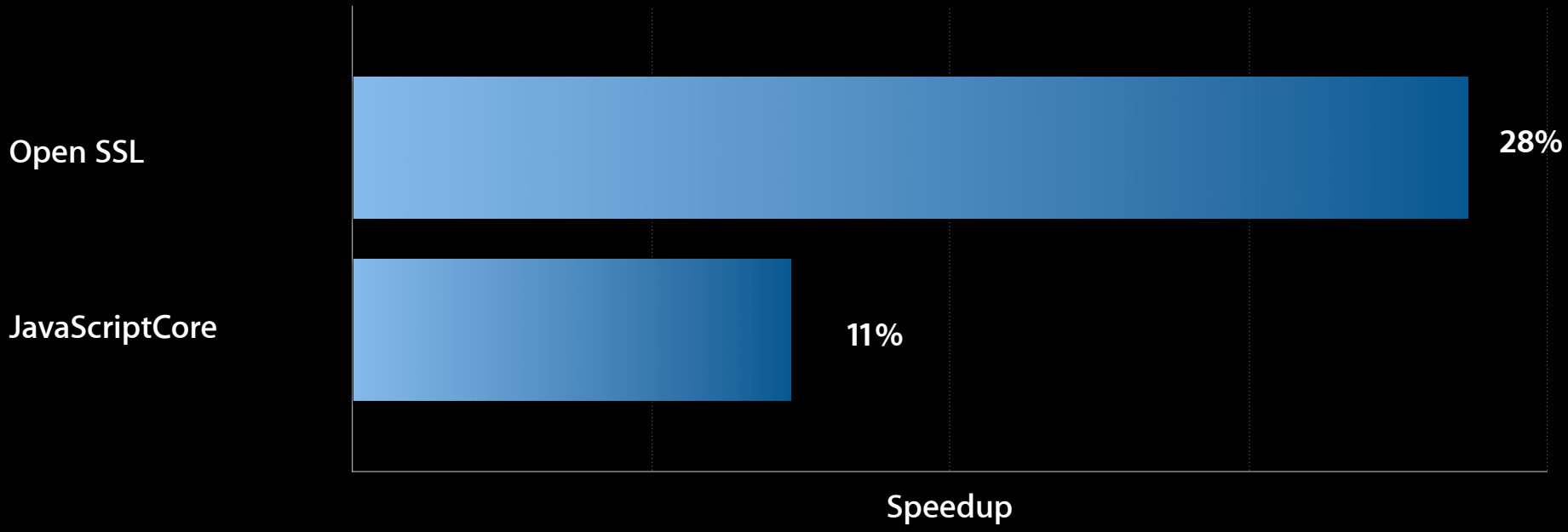
LLVM-GCC

GCC Parser with LLVM code Generation

- Combines:
  - Advantages of the LLVM back end
  - Production quality C++ support

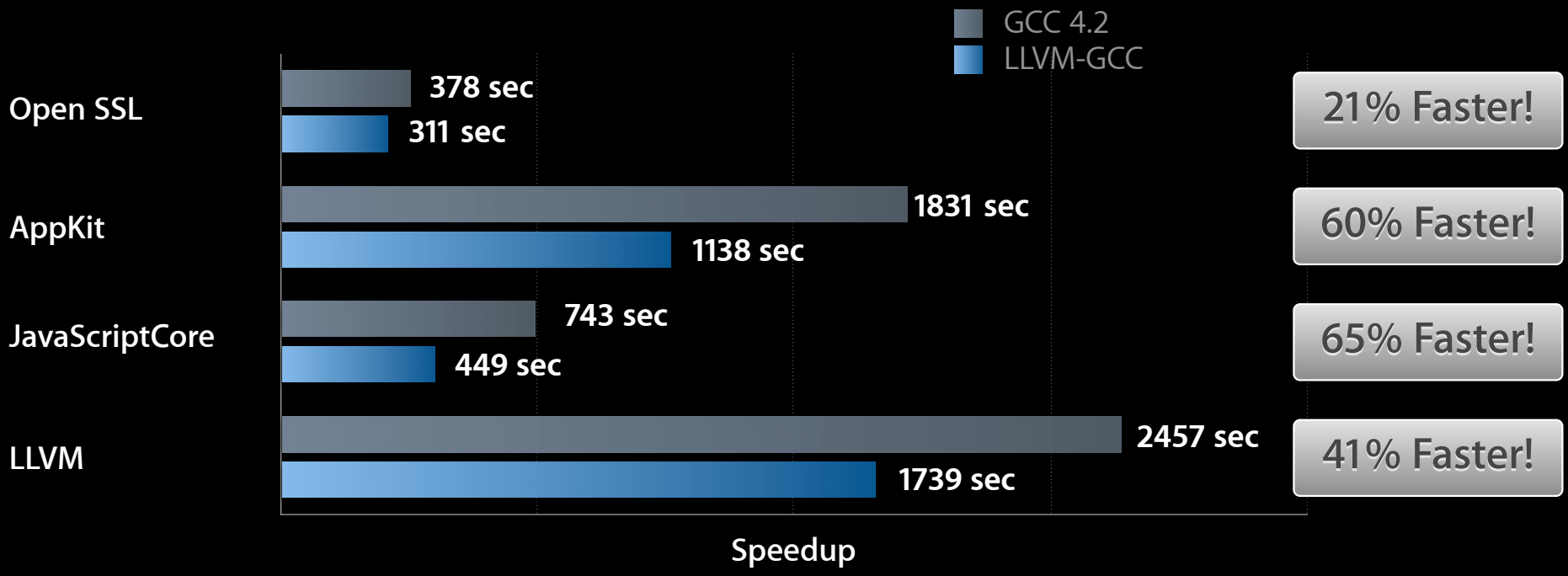
Most mature and stable compiler of the three

Example Performance Wins



- Built with LLVM in iOS 4
  - WebKit, JavaScriptCore, OpenSSL

Faster Release Builds Than GCC



What about debug builds?

LLVM Compiler 1.5

Clang Parser with LLVM code generation

Now available for iPhone OS!

What Is the Clang Frontend?

- Parser for C family of languages
- LLVM library-based design approach
- Designed to build source-level tools
- http://clang.llvm.org/

Why Do We Care?

We want an amazing compiler experience!

- Fast compile times
- Great user features
- Beautiful error messages

Must keep great compatibility with GCC!

Last Year—Faster Debug Builds…

Error and Warning Improvements

- "Diagnostics" are error and warning reports
  - Compiler detects something is wrong
  - Tries to guess why and explain it
- GCC diagnostics are often not helpful
  - Confusing, poorly worded, not precise

```
$ gcc test.m
test.m:2: error: expected ‘=’, ‘,’, ‘;’, ‘asm’ or ‘__attribute__’ before ‘*’ token
```

```
$ clang test.m
test.m:2:1: error: unknown type name 'NSstring'
NSstring *P = @"good stuff";
^
```

Spell Checking in Clang

- Clang helps identify typos based on the names in your source!
- Spell checking catches many simple errors for you

```
$ clang test.m
test.m:2:1: error: unknown type name 'NSstring'; did you mean 'NSString'?
NSstring *P = @"good stuff";
^~~~~~~
```

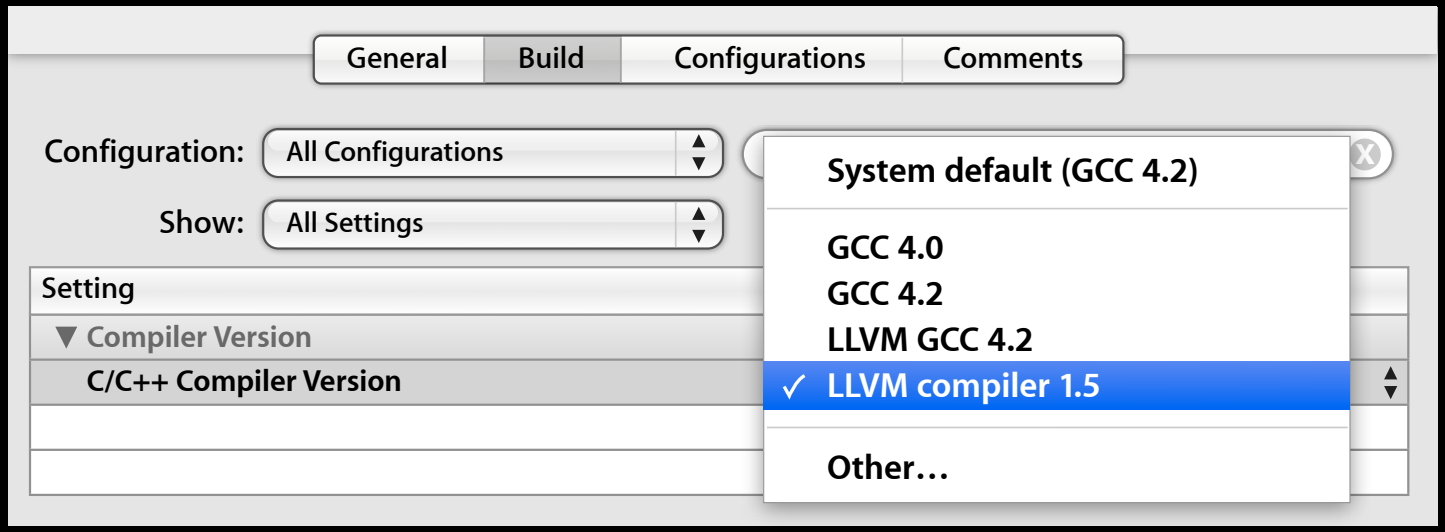
LLVM Technologies In Depth	Nob Hill Thursday 3:15PM

New Warnings Available

- Format string warnings: -Wformat
- Control-flow warnings: -Wreturn-type
- Value warnings: -Wshorten-64-to-32, -Wconversion, -Wsign-compare
- …and many more!

```
test.c:2:17: warning: conversion specifies type 'int' but the argument has type
      'double' [-Wformat]
  printf("<%s, %d>", key, value);
              ~^          ~~~~~
```

Selecting a Compiler in Xcode



More Than Just Compilers

- Modular, library-based design encourages reuse!
- Clang is part of many great new tools
  - LLVM Compiler
  - Xcode Static Analyzer
  - Xcode 4
  - LLDB Debugger
  - OpenCL

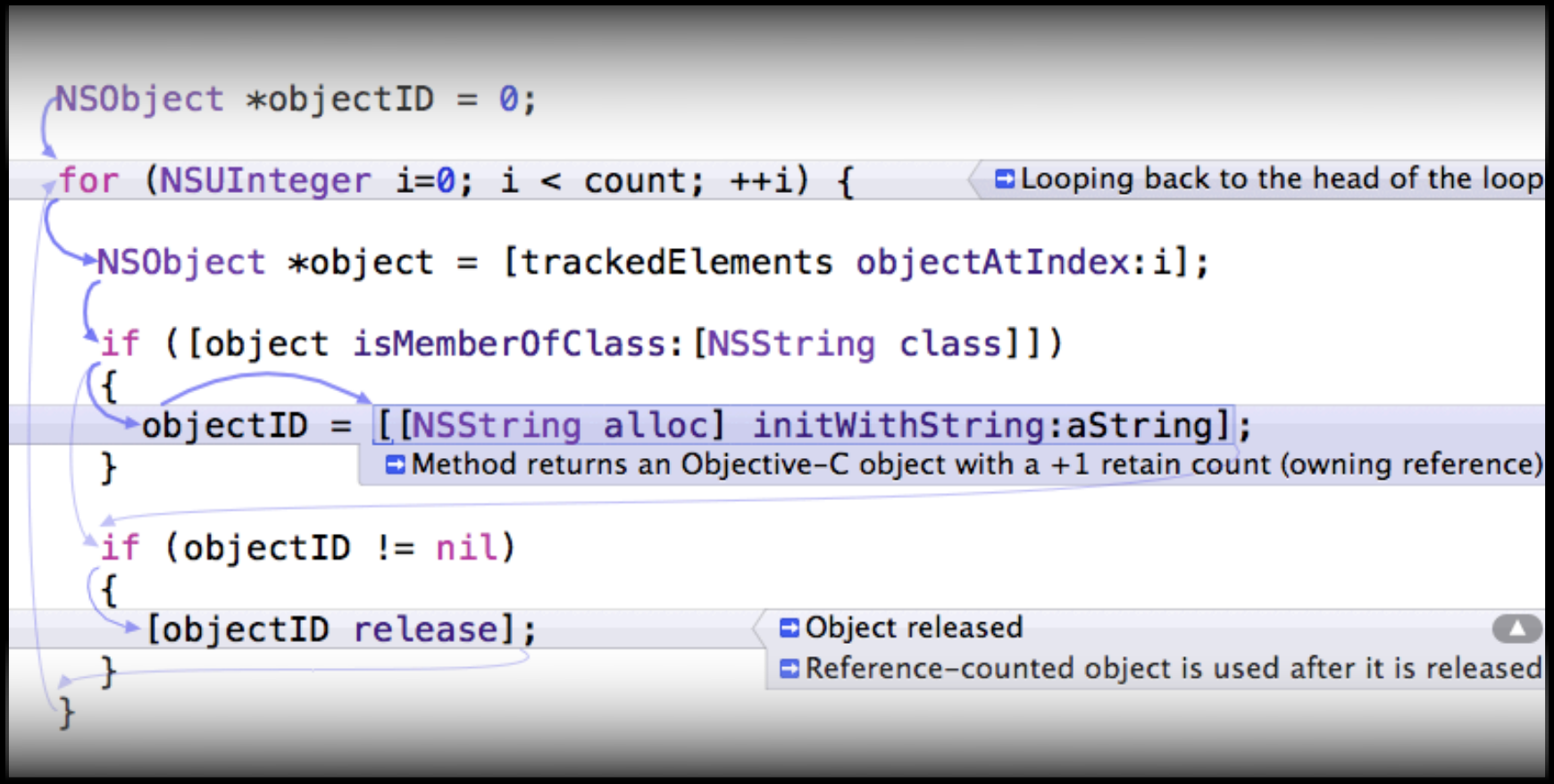
Xcode Static Analyzer

Automatically find bugs by analyzing your code

- Early detection of bugs via static code analysis
- Works on Mac and iPhone OS projects
- Easy to use—Build and Analyze

Xcode Static Analyzer

Automatically find bugs by analyzing your code



Static Analyzer Improvements in Xcode 3.2.3

- Numerous "under-the-cover" improvements
  - Analysis support for Blocks
  - Better precision for array and field values
  - More API checks (e.g., Grand Central Dispatch)
- Finds more bugs with fewer false positives!
- http://clang-analyzer.llvm.org/

LLVM in Xcode 4

Doug Gregor
Clang C++ Technical Lead

LLVM in Xcode 4

Preview

- New tools
  - LLVM Compiler 2
  - New LLVM-based Debugger (LLDB)
- Clang integrated into Xcode 4
  - Source code indexing
  - Syntax highlighting
  - Code completion
  - Live warnings and Fix-its

LLVM Technologies In Depth	Nob Hill Thursday 3:15PM

Compiler Landscape in Xcode 4

GCC 4.0

GCC 4.2

LLVM-GCC 4.2

Default compiler

LLVM Compiler 2

Default for
new projects

LLVM Compiler 2

All C++ features implemented*

except exported temp

boost

FreeBSD

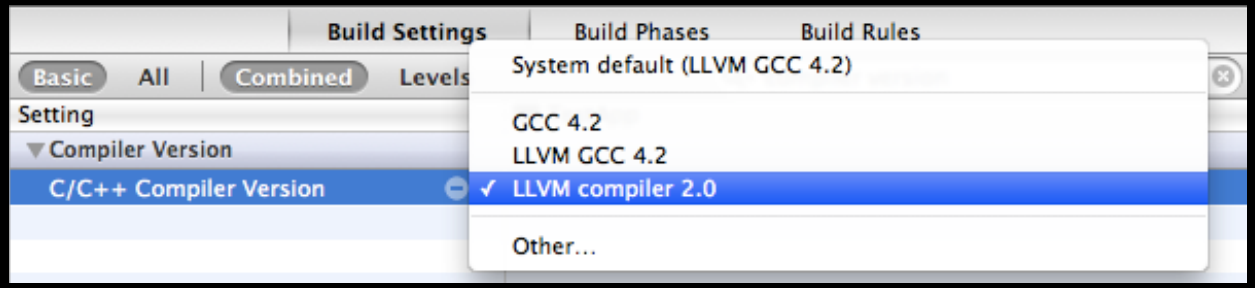
* The mark FreeBSD is a registered trademark of The FreeBSD Foundation and is used by Apple, Inc. with the permission of The FreeBSD Foundation.

Benefits of LLVM Compiler 2 C++

- Faster compilation
- Greatly improved warnings and errors
- Complete binary compatibility
- Better C++ standards conformance
- Objective-C++ support
- http://clang.llvm.org/cxx_compatibility.html

Trying Out the New LLVM Compiler 2

- Within Xcode 4
  - Default compiler for new projects
  - Select your compiler in the project build settings

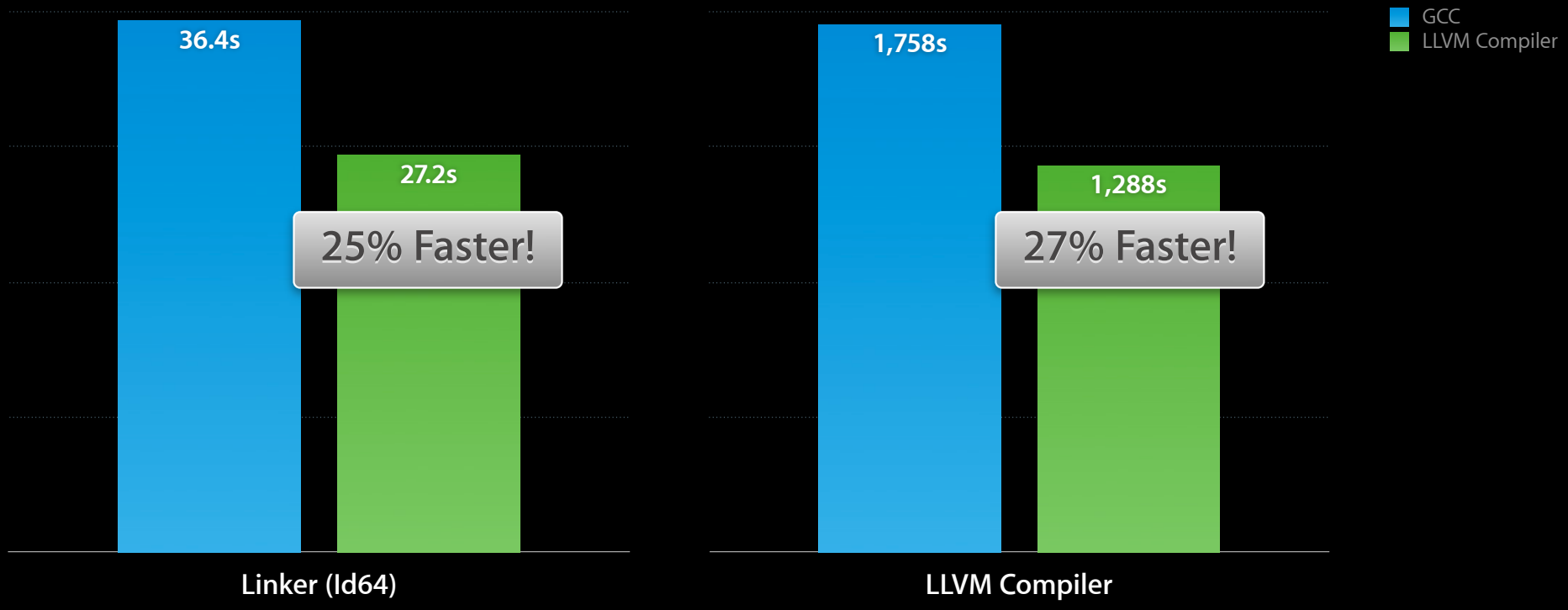


- Command line

```
$ /Xcode4/usr/bin/clang -c myfile.m
$ /Xcode4/usr/bin/clang++ -c another_file.cpp
```

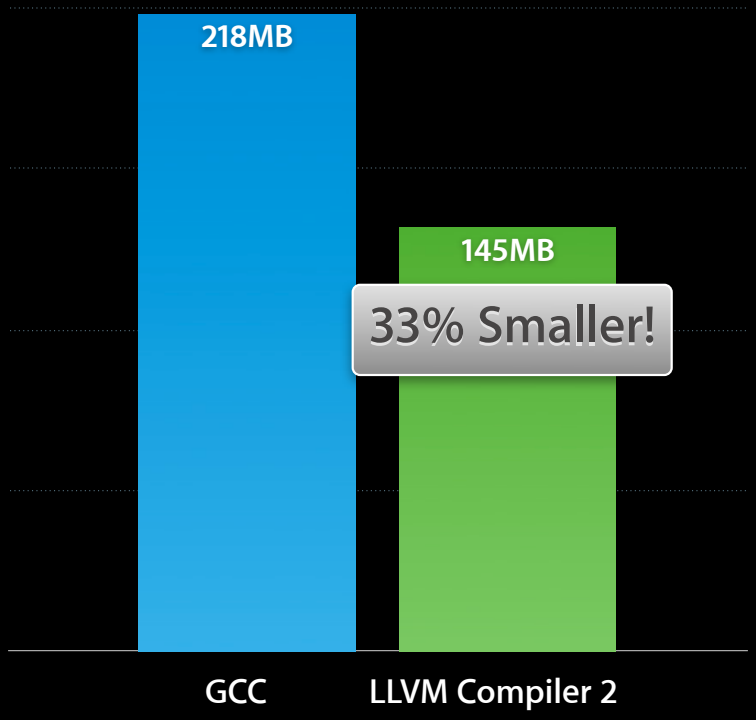
Performance

C++ Compile-Time Performance



C++ Memory Usage

Building—Mac OS X Linker (ld64)



Errors and Warnings

Precise, Informative Diagnostics

```
template<typename T> class VectorImpl {
public:
  void clear();
};


template<typename T> class Vector : VectorImpl<T> { };


void clear(Vector<int> &v) { v.clear(); }
```

- LLVM Compiler 2
- GCC 4.2

```
$ clang++ access.cpp
access.cpp:8:32: error: 'clear' is a private member of 'VectorImpl<int>'
void clear(Vector<int> &v) { v.clear(); }
                               ^
access.cpp:3:8: note: declared private here
  void clear();
       ^
```

```
$ g++ access.cpp
access.cpp: In function 'void f(Vector<int>)':
access.cpp:3: error: 'void VectorImpl<T>::clear() [with T = int]' is inaccessible
access.cpp:8: error: within this context
class Vector : VectorImpl<T> {
               ^~~~~~~~~~~~
access.cpp:6:37: note: constrained by implicitly private inheritance here
```

Ambiguities

```
class Person { };
class Teacher : public Person { };
class Student : public Person { };
class TeachingAssistant : public Teacher, public Student { };

void greet(Person *);
void startClass(TeachingAssistant *TA) { greet(TA); }
```

```
people.cpp:7:48: error: ambiguous conversion from derived class
      'TeachingAssistant' to base class 'Person':
    class TeachingAssistant -> class Teacher -> class Person
    class TeachingAssistant -> class Student -> class Person
  void startClass(TeachingAssistant *TA) { greet(TA); }
                                                 ^~
```

Spell Checking

```
typo.cpp:4:8: error: no template named 'Vector' in namespace 'std'; did you mean
      'vector'?
  std::Vector<float> grades;
  ~~~~~^~~~~~
       vector
In file included from typo.cpp:1:
In file included from /usr/include/c++/4.2.1/vector:69:
/usr/include/c++/4.2.1/bits/stl_vector.h:162:11: note: 'vector' declared here
    class vector : protected _Vector_base<_Tp, _Alloc>
          ^
```

Understanding Overload Failures

```
class Canvas {
public:
  void Draw(Image *image, Point top_left);
  void Draw(Point *points, unsigned num_points, Color color);
};

void DrawPoly(Canvas *canvas, Point *firstPt, Point *lastPt, Color c) {
  canvas->Draw(firstPt, lastPt, c);
}
```

```
draw.cpp:16:11: error: no matching member function for call to 'Draw'
  canvas->Draw(firstPt, lastPt, c);
  ~~~~~~~~^~~~

draw.cpp:12:8: note: candidate function not viable: no known conversion from
      'Point *' to 'unsigned int' for 2nd argument
  void Draw(Point *points, unsigned num_points, Color color);
       ^

draw.cpp:11:8: note: candidate function not viable: requires 2 arguments, but 3
      were provided
  void Draw(Image *image, Point top_left);
       ^
```

C++ Moving Forward

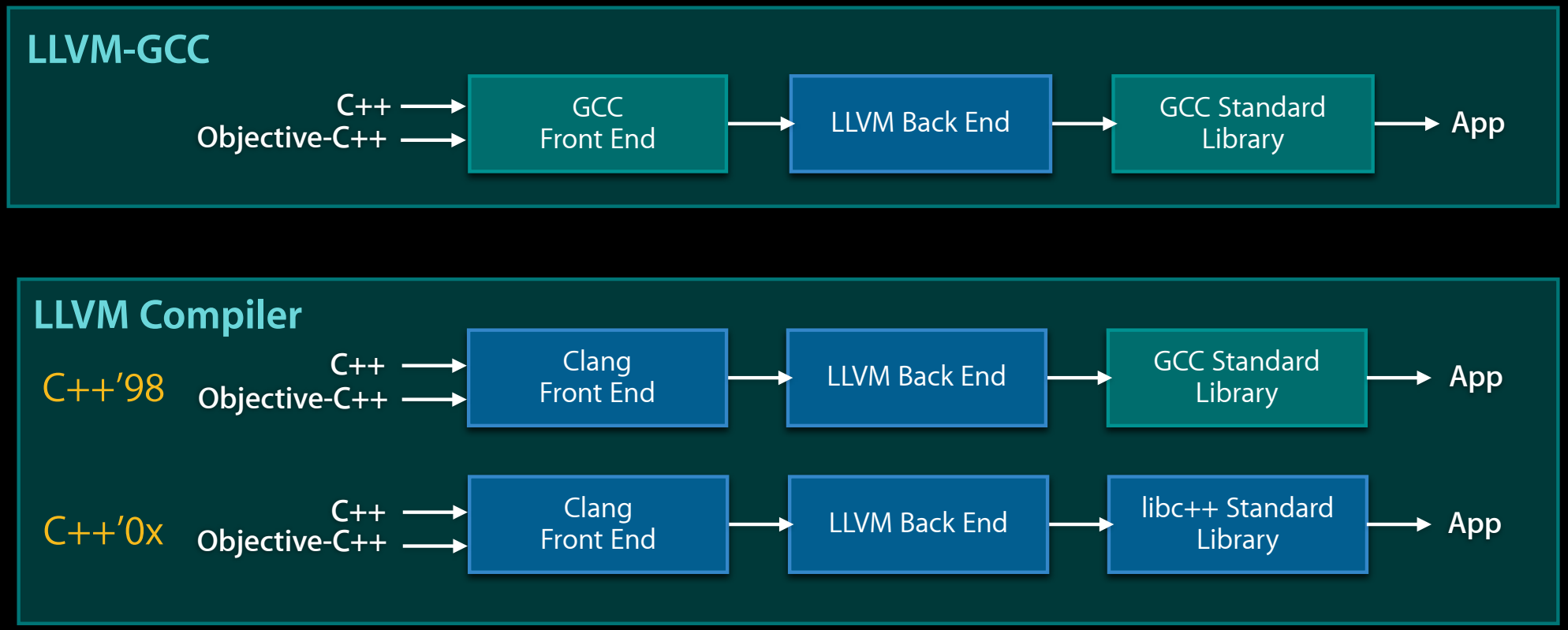
Upcoming C++ ‘0x Standard

- Major upgrade to C++ language and library
- Backward compatible with current C++ standard
- ISO C++ Standard Expected in 2011

libc++: A New C++ Standard Library

- Designed for C++'0x
- Better performance
- 100% open source
  - http://libcxx.llvm.org/

C++ Tools Evolution



Summary

Xcode 3.2.3

- LLVM-GCC
  - Supports iPhone OS development
  - Better optimized code and smaller compiled binaries
- LLVM Compiler 1.5
  - Super fast compiles
  - New warnings and static analyzer improvements

Summary

Xcode 4 preview

- New tools
  - LLVM Compiler 2 (with C++ support)
  - LLDB (new LLVM-based debugger)
- Clang-powered Xcode features
  - Clang-based code completion and source indexing
  - Live warnings and Fix-its

More Information

Michael Jurewitz
Developer Tools Evangelist
jurewitz@apple.com

LLVM Project
Open Source LLVM Project Home
http://llvm.org

Apple Developer Forums
http://devforums.apple.com

Related Sessions

LLVM Technologies In Depth	Nob Hill Thursday 3:15PM
Debugging with Xcode 4 and LLDB	Mission Friday 9:00AM

Labs

LLVM Lab	Developer Tools Lab B Thursday 4:30PM
Xcode 4 Lab	Developer Tools Lab B Friday 9:00AM

Q&A


WWDC10