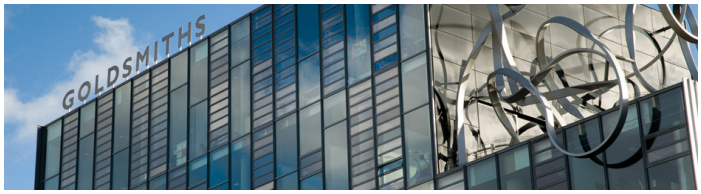
The LLVM Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!

The LLVM Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!



Goldsmiths
UNIVERSITY OF LONDON

IGGI Doctoral Center - PHD funding for LLVM[link]

Gamification (FoldIt, DockIt: Imperial) [link]

Art (Mutator II: Brighton/Brussels) [link]

MSc Computer Games and Entertainment [link]

Bioinformatics (Rosalind tools: KCL) [link]

Past: Compilers, Games, TAOS JIT, Sinclair PGC7600, Psygnosis Tech Group, Sony Computer Entertainment (SNC)

The **LLVM** Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!

The M x N problem

M modules
N include files

a.h	a.h	a.h	a.h
b.h	b.h	b.h	b.h
c.h	c.h	c.h	c.h
d.h	d.h	d.h	d.h
a.cpp	b.cpp	c.cpp	d.cpp

M + M * N * k files (k ~= 0.5)
clang: M = 1,387, N = 1,334+
Approx 1,000,000 files parsed
About 31½mins on decent machine @ -O0

The **LLVM** Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!

Unity builds

a.h	a.h
b.h	b.h
c.h	c.h
d.h	d.h
a.cpp	c.cpp
b.cpp	d.cpp
u1.cpp	u2.cpp

We build M + U + U * N * k files
Approx 10,000 files parsed in unity build with U=12
About 2½ mins on same machine(> 12 times faster)
EDG (same size as Clang) builds in 1sec
Still not good enough!

Unity builds - other benefits

COFF/ELF files much smaller - possible to use -g
Link much faster - less duplication
Link time code generation not necessary in most cases

The **LLVM** Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!

Unity builds - problems

static variables/functions

anonymous namespaces

"using namespace"

The **LLVM** Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!

Process-level parallelism: 4 core machine building clang.exe make -j n

361,720ms	1	
224,299ms	2	
175,065ms	3	
152,865ms	4	
150,322ms	5	
146,578ms	6	
145,767ms	7	
145,018ms	8	← Minimum
145,409ms	9	
146,313ms	10	
147,155ms	11	
147,234ms	12	

The **LLVM** Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!

PCH files

```
#include "stdafx.h"
#pragma hdrstop
```

Only accelerates the parsing
Need all .h files in master include

The **LLVM** Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!

Distributed builds

My machine	u1.cpp	u2.cpp	u3.cpp
Greg's machine	u4.cpp	u5.cpp	u6.cpp
Andrea's machine	u7.cpp	u8.cpp	
Rob's machine	u9.cpp	u10.cpp	u11.cpp

Four times the cores
Could use Amazon EC2

The **LLVM** Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!

Distributed builds - problems

Must distribute the source
Must collect the object code
Some serial processes - **tablegen**
Limited by longest file build time
Cost and convenience

The **LLVM** Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!

Distributed builds

SN DBS distributed build system
Incredibuild
Buildbot
DistCC

The **LLVM** Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!

Sourceforge project

http://sourceforge.net/projects/llvm-unity
Based on llvm-3.4
Hundreds of edits to source
LLVM without tablegen in about 30 secs
Clang edits in about 30 secs

The **LLVM** Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!

Slowest files

8986ms	tools/clang/lib/ASTMatchers/Dynamic/Registry.cpp	24,137,394 obj bytes
8518ms	tools/clang/lib/Sema/SemaExpr.cpp	9,394,440 obj bytes
7598ms	lib/Target/X86/InstPrinter/X86ATTInstPrinter.cpp	330,319 obj bytes
7238ms	tools/clang/lib/Serialization/ASTWriter.cpp	5,017,589 obj bytes
7207ms	tools/clang/lib/Sema/SemaTemplate.cpp	7,481,788 obj bytes
7098ms	tools/clang/lib/Sema/SemaDeclCXX.cpp	7,510,102 obj bytes

Total: 851,736,448 obj bytes for conventional build
Total: 448,284,424 obj bytes for unity build

The **LLVM** Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!

Slowest files - causes

Over-abstraction - especially STL
Multiple inheritance - huge numbers of thunks
redundancy - many functions unused
LLVM better than clang - has own vector and map classes

Better unity builds

Level 1: unity files contain existing .cpp files
Level 2: Headers sorted by dependency order.
Level 3: One .cpp file and all functions inline (.inl or .h)

Many large projects build in under a second
Stateless builds are best - no configure needed
No need for dependency checking
Future source-only distribution.

The **LLVM** Compiler Infrastructure

How to build LLVM in ten seconds - or die trying!

Future work

Unity build tablegen files
Investigate clang file slowdowns in detail
Build with clang and use stats
Cut down build: lose static analysis, JIT and Disassembler.
Improve latency for delta-builds
Improve dependency generation and checking
Language features for unity builds
LLVM modules proposal [link]
llvm-build [link]

Conclusions

Still chasing 10 seconds - currently 2 mins

Will the community accept unity builds?

Do we want to build LLVM faster?

Questions?

Edit/Compile/Link time?

Will this introduce bugs?

I really don't like unity builds!