The **LLVM** Compiler Infrastructure

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

---

**EuroLLVM**

Edinburgh, Scotland

April 7-8, 2014

---

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 31½ mins 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!*
The **LLVM** Compiler Infrastructure

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

**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

<table>
<thead>
<tr>
<th>Threads</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>361,720ms</td>
</tr>
<tr>
<td>2</td>
<td>224,299ms</td>
</tr>
<tr>
<td>3</td>
<td>175,065ms</td>
</tr>
<tr>
<td>4</td>
<td>152,865ms</td>
</tr>
<tr>
<td>5</td>
<td>150,322ms</td>
</tr>
<tr>
<td>6</td>
<td>146,578ms</td>
</tr>
<tr>
<td>7</td>
<td>145,767ms</td>
</tr>
</tbody>
</table>
| 8       | 145,018ms| ← Minimum
| 9       | 145,409ms|
| 10      | 146,313ms|
| 11      | 147,155ms|
| 12      | 147,234ms|

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
## Distributed builds

<table>
<thead>
<tr>
<th>Machine</th>
<th>File 1</th>
<th>File 2</th>
<th>File 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>My machine</td>
<td>u1.cpp</td>
<td>u2.cpp</td>
<td>u3.cpp</td>
</tr>
<tr>
<td>Greg's machine</td>
<td>u4.cpp</td>
<td>u5.cpp</td>
<td>u6.cpp</td>
</tr>
<tr>
<td>Andrea's machine</td>
<td>u7.cpp</td>
<td>u8.cpp</td>
<td></td>
</tr>
<tr>
<td>Rob's machine</td>
<td>u9.cpp</td>
<td>u10.cpp</td>
<td>u11.cpp</td>
</tr>
</tbody>
</table>

Four times the cores  
Could use Amazon EC2

---

## 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

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

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
The **LLVM** Compiler Infrastructure

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

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]
The **LLVM** Compiler Infrastructure

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

## 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!