CodeView, the MS debug info format, in LLVM

Reid Kleckner

Google

Talk overview

1. Why use CodeView? What are PDBs and CodeView?
2. CodeView format basics
3. CodeView deduplication techniques
4. Implementation status in LLVM
5. Lessons for LLVM

Terminology: What is CodeView?

- CodeView is stored inside object files and PDBs
  - DWARF
- PDB is a container format written by linker and read by debugger
  - dSYM/DWP

CodeView :: DWARF

PDB :: dSYM

PDB :: DWP

Why add CodeView support to LLVM?

Windows has a rich ecosystem built around PDBs:

- Visual Studio debugger
- WinDBG
- Stack dumpers built on dbghelp.dll
- Windows Performance Analyzer (WPA) profiler
- Symbol servers

Object file structure of CodeView

- Type information lives in one `.debug$T` section
- Symbol information (everything else) lives in `.debug$S` sections
- Symbol section broken down into subsections:
  - Symbol records (most stuff), line table, string table, unwind info, etc

```
struct Point { int x, y; };
int main() {
  Point p{1, 2};
  printf("%d %d\n", p.x, p.y);
}
```

Why split out type information?

- Type information is repeated in every TU
  - Often dominates link input size
- Deduplicating type information:
  - Reduces PDB output size
  - Speeds up links (less IO)
  - Speeds up debugging (more compact PDB)
- Splitting out type info makes this easier
- DWARF type units are similar

Type deduplication strategy

- Build a graph of type records
- Type records will be our graph nodes
- Type indices will be our graph edges
- Merge type graphs to deduplicate

Problem: Graph isomorphism is slow!

Type record format

Sequence of 4-byte aligned record prefixed by 16-bit length and 16-bit kind:

```
typedef struct TYPTYPE {
    unsigned short  len;
    unsigned short  leaf;
    unsigned char   data[CV_ZEROLEN];
} TYPTYPE;          // general types record
```

- Described in cvinfo.h, published by Microsoft on GitHub
- 16-bit size means large records must be split or truncated
- Compare bytes for equivalence
- Amenable to memory-mapped IO, unlike DWARF abbreviations

Type graph representation

- Assign all types a "type index"
- Simple types have reserved indices below `0x1000`:
  - `int`, `short`, `int*`, `void*`, etc
- Type records refer to other types by index
- Assign the type index `0x1000 + N` to the Nth type record

Cycles in type graph

Consider a linked list:

```
struct Foo { Foo *p; };
```

Make the type graph a DAG

- Only records introduce cycles
- Always refer to records by forward declaration

Make the type graph a DAG

- Only records introduce cycles
- Always refer to records by forward declaration

```
0x1000: struct Foo;
0x1001: Foo*          # <0x1000>
0x1003: { Foo *p; };  # <0x1001>
0x1004: struct Foo <0x1003>
```

Make the type graph a DAG

- Only records introduce cycles
- Always refer to records by forward declaration

```
0x1000: struct Foo;
0x1001: Foo*           # <0x1000>
0x1003: { Foo *p; };   # <0x1001>
0x1004: struct Foo <0x1003>
```

- A type record may only use type indices smaller than its index
- Type info stream is always a topologically sorted DAG
- Type records using the same type indices should be bitwise identical

Deduplicating types and merging streams

- Inputs: `dst` type stream, `src` type stream
- `recordmap`: Map from `dst` type record contents to type index
- `src2dst`: Map from `src` type index to `dst` type index

Deduplicating types and merging streams

- For each type record `r` in `src`:
  - Rewrite type indices in `r` using `src2dst`
  - Look up any existing index for `r` in `recordmap`
  - If not found, append `r` to `dst` and update `recordmap`
  - Update `src2dst` to map from old index to new index

Type server optimization (/Zi)

- **Problem:** Linker inputs are still too large due to type info
- **Solution:** Move type merging work from linking to compilation

Type server optimization (/Zi)

- Use the same type server PDB for many compilations (/Fd)
- Start common mspdbsrv.exe process
- For each type record, IPC with mspdbsrv to get type index
  - Insert type record into PDB if not already present
- Link step merges type server PDBs as before, but with less input

Can apply this idea to code, see Paul Bowen-Hugget's talk

Issues with type servers

- Not currently pursuing LLVM implementation
- Compilation must block on IPC to get type index
  - Consider using content hash to identify types
- IPC doesn't distribute well, blocking RPC would be a disaster
- mspdbsrv IPC protocol is undocumented

Might revisit building llvm-pdbsrv in the future

Symbol information format

```
// Generic layout for symbol records
typedef struct SYMTYPE {
    unsigned short      reclen;     // Record length
    unsigned short      rectyp;     // Record type
    char                data[CV_ZEROLEN];
} SYMTYPE;
```

- Very familiar, with key differences:
  - No indices or other cross-record references
  - Symbol records "contain" other symbol records
  - Has relocations against .text, .data, etc

Symbol information example

Describes scopes with XML-like start/end record pairs

```
volatile int y = 0;
static void h(int x) { y = x; }
static void g(int x) { h(x); }
int f(int x) {
  if (x) {
    int z = y;
    g(x);
    x += z;
  }
  return x;
}
```

```
- S_GPROC32 f
  - S_LOCAL x
  - S_BLOCK32
    - S_LOCAL z
    - S_INLINESITE g
      - S_INLINESITE h
        - S_LOCAL x
      - S_INLINESITE_END
    - S_INLINESITE_END
  - S_END
- S_END
```

COMDATs in CodeView

One `.debug$S` section per COMDAT function or global

```
inline void f(void) {}
inline void g(void) {}
inline void h(void) {}
int main() { f(); g(); h(); }

$ clang -S t.cpp -g -gcodeview \
  --target=x86_64-windows -o - | \
  grep 'debug\$S'
```

```
.section .debug$S,"dr"
.section .debug$S,"dr",associative,"?f@@YAXXZ"
.section .debug$S,"dr",associative,"?g@@YAXXZ"
.section .debug$S,"dr",associative,"?h@@YAXXZ"
.section .debug$S,"dr"
```

DWARF uses monolithic sections

```
inline void f(void) {}
inline void g(void) {}
inline void h(void) {}
int main() { f(); g(); h(); }
```

```
$ clang -c t.cpp -g \
  --target=x86_64-linux -o - | \
  llvm-objdump -r - | \
  grep -v '32 \.debug'
```

```
...
RELOCATION RECORDS FOR [.rela.debug_info]:
000000000000002b R_X86_64_64 .text+0
0000000000000044 R_X86_64_64 .text._Z1fv+0
000000000000005d R_X86_64_64 .text._Z1gv+0
0000000000000076 R_X86_64_64 .text._Z1hv+0

RELOCATION RECORDS FOR [.rela.debug_ranges]:
0000000000000000 R_X86_64_64 .text+0
0000000000000008 R_X86_64_64 .text+23
0000000000000010 R_X86_64_64 .text._Z1fv+0
0000000000000018 R_X86_64_64 .text._Z1fv+6
...
```

LLVM implementation status

- Basics: functions, globals, line tables
- Optimized debug info:
  - Inlined call frames and line tables
  - Register allocated locals
  - Scalarized aggregates (SROA)
- PDB writing under development

Canned demo time

Use **ALL** the optimized debug info features!

```c
#include <stdio.h>
struct IntPair { int x, y; };
int __declspec(noinline)
g(int r) { return r + 1; }
int i, n = 4;
static inline int loop_csr() {
  struct IntPair o = {0, 0};
  for (i = 0; i < n; i++) {
    o.x = g(o.x);
    o.y = g(o.y);
  }
  return o.x + o.y;
}
int main() {
  return loop_csr();
}
```

```asm
...
  xorl    %edi, %edi
  movl    %ebx, %ebp
  xorl    %esi, %esi
.LBB1_3:
  movl    %edi, %ecx
  callq   g
  movl    %eax, %edi
  movl    %esi, %ecx
  callq   g
  movl    %eax, %esi
  decl    %ebp
  jne     .LBB1_3
...
```

t.exe - WinDbg:10.0.10586.567 AMD64

File Edit View Debug Window Help

C:\src\llvm\build\t.cpp

```
#include <stdio.h>
struct IntPair { int x, y; };
int __declspec(noinline) g(int r) { return r + 1; }
int i, n = 4;
volatile int v;
static inline int loop_csr() {
  struct IntPair o = {0, 0};
  for (i = 0; i < n; i++) {
    o.x = g(o.x);
    o.y = g(o.y);
  }
  return o.x + o.y;
}
int main() {
  return loop_csr();
}
```

Command

```
ModLoad: 00007ff6`a8f00000 00007ff6`a8f5d000   t.exe
ModLoad: 00007ffa`fd820000 00007ffa`fd9e1000   ntdll.dll
ModLoad: 00007ffa`fcd50000 00007ffa`fcdfd000   C:\Windows\sy
ModLoad: 00007ffa`f9ed0000 00007ffa`fa0b8000   C:\Windows\sy
(11cf8.11ea4): Break instruction exception - code 80000003 (
ntdll!LdrpDoDebuggerBreak+0x30:
00007ffa`fd8eaa60 cc              int     3
0:000> bp t!main
*** WARNING: Unable to verify checksum for t.exe
0:000> g
Breakpoint 0 hit
t!main:
00007ff6`a8f01010 56              push    rsi
0:000> k
 # Child-SP          RetAddr           Call Site
00 0000009f`196ffc58 00007ff6`a8f01289 t!main [C:\src\llvm\b
*** ERROR: Symbol file could not be found.  Defaulted to exp
01 (Inline Function) --------`-------- t!invoke_main+0x22 [f
02 0000009f`196ffc60 00007ffa`fcd68102 t!__scrt_common_main_
03 0000009f`196ffca0 00007ffa`fd87c5b4 KERNEL32!BaseThreadIr
04 0000009f`196ffcd0 00000000`00000000 ntdll!RtlUserThreadSt
```

0:000>

Ln 14, Col 1 | Sys 0:<Local> | Proc 000:11cf8 | Thrd 000:11ea4 | ASM | OVR | CAPS | NUM

t.exe - WinDbg:10.0.10586.567 AMD64

File Edit View Debug Window Help

C:\src\llvm\build\t.cpp

```
#include <stdio.h>
struct IntPair { int x, y; };
int __declspec(noinline) g(int r) { return r + 1; }
int i, n = 4;
volatile int v;
static inline int loop_csr() {
  struct IntPair o = {0, 0};
  for (i = 0; i < n; i++) {
    o.x = g(o.x);
    o.y = g(o.y);
  }
  return o.x + o.y;
}
int main() {
  return loop_csr();
}
```

Command

```
t!main:
00007ff6`a8f01010 56              push    rsi
0:000> k
 # Child-SP          RetAddr           Call Site
00 0000009f`196ffc58 00007ff6`a8f01289 t!main [C:\src\llvm\b
*** ERROR: Symbol file could not be found.  Defaulted to exp
01 (Inline Function) --------`-------- t!invoke_main+0x22 [f
02 0000009f`196ffc60 00007ffa`fcd68102 t!__scrt_common_main_
03 0000009f`196ffca0 00007ffa`fd87c5b4 KERNEL32!BaseThreadIr
04 0000009f`196ffcd0 00000000`00000000 ntdll!RtlUserThreadSt
0:000> t
t!loop_csr [inlined in t!main+0x8]:
00007ff6`a8f01018 c7052e3a050000000000 mov dword ptr [t!i (0
0:000> k
 # Child-SP          RetAddr           Call Site
00 (Inline Function) --------`-------- t!loop_csr [C:\src\l]
01 0000009f`196ffc10 00007ff6`a8f01289 t!main+0x8 [C:\src\l]
02 (Inline Function) --------`-------- t!invoke_main+0x22 [f
03 0000009f`196ffc60 00007ffa`fcd68102 t!__scrt_common_main_
04 0000009f`196ffca0 00007ffa`fd87c5b4 KERNEL32!BaseThreadIr
05 0000009f`196ffcd0 00000000`00000000 ntdll!RtlUserThreadSt
```

0:000>

Ln 8, Col 1 | Sys 0:<Local> | Proc 000:11cf8 | Thrd 000:11ea4 | ASM | OVR | CAPS | NUM

t.exe - WinDbg:10.0.10586.567 AMD64

File Edit View Debug Window Help

C:\src\llvm\build\t.cpp

```
#include <stdio.h>
struct IntPair { int x, y; };
int __declspec(noinline) g(int r) { return r + 1; }
int i, n = 4;
volatile int v;
static inline int loop_csr() {
  struct IntPair o = {0, 0};
  for (i = 0; i < n; i++) {
    o.x = g(o.x);
    o.y = g(o.y);
  }
  return o.x + o.y;
}
int main() {
  return loop_csr();
}
```

Command

```
ModLoad: 00007ffa`fd820000 00007ffa`fd9e1000   ntdll.dll
ModLoad: 00007ffa`fcd50000 00007ffa`fcdfd000   C:\Windows\sy
ModLoad: 00007ffa`f9ed0000 00007ffa`fa0b8000   C:\Windows\sy
(1197c.10d6c): Break instruction exception - code 80000003 (
ntdll!LdrpDoDebuggerBreak+0x30:
00007ffa`fd8eaa60 cc              int     3
0:000> bp t!main
*** WARNING: Unable to verify checksum for t.exe
0:000> g
Breakpoint 0 hit
t!main:
00007ff6`a8f01010 56              push    rsi
0:000> t
t!loop_csr [inlined in t!main+0x8]:
00007ff6`a8f01018 c7052e3a050000000000 mov dword ptr [t!i ((
0:000> t
t!main+0x1e:
00007ff6`a8f0102e 89dd            mov     ebp,ebx
0:000> t
t!loop_csr+0x28 [inlined in t!main+0x30]:
00007ff6`a8f01040 89f9            mov     ecx,edi
```

0:000>

Ln 9, Col 1 | Sys 0:<Local> | Proc 000:1197c | Thrd 000:10d6c | ASM | OVR | CAPS | NUM

t.exe - WinDbg:10.0.10586.567 AMD64

File Edit View Debug Window Help

C:\src\llvm\build\t.cpp

```
#include <stdio.h>
struct IntPair { int x, y; };
int __declspec(noinline) g(int r) { return r + 1; }
int i, n = 4;
volatile int v;
static inline int loop_csr() {
  struct IntPair o = {0, 0};
  for (i = 0; i < n; i++) {
    o.x = g(o.x);
    o.y = g(o.y);
  }
  return o.x + o.y;
}
int main() {
  return loop_csr();
}
```

Command

```
t!loop_csr [inlined in t!main+0x8]:
00007ff6`a8f01018 c7052e3a050000000000 mov dword ptr [t!i (0
0:000> t
t!main+0x1e:
00007ff6`a8f0102e 89dd            mov     ebp,ebx
0:000> t
t!loop_csr+0x28 [inlined in t!main+0x30]:
00007ff6`a8f01040 89f9            mov     ecx,edi
0:000> t
t!g:
00007ff6`a8f01000 8d4101          lea     eax,[rcx+1]
0:000> k
 # Child-SP          RetAddr           Call Site
00 000000f9`172ff798 00007ff6`a8f01047 t!g [C:\src\llvm\bui
01 (Inline Function) --------`-------- t!loop_csr+0x2f [C:\s
02 000000f9`172ff7a0 00007ff6`a8f01289 t!main+0x37 [C:\src\
*** ERROR: Symbol file could not be found.  Defaulted to exp
03 (Inline Function) --------`-------- t!invoke_main+0x22 [f
04 000000f9`172ff7f0 00007ffa`fcd68102 t!__scrt_common_main_
05 000000f9`172ff830 00007ffa`fd87c5b4 KERNEL32!BaseThreadIn
06 000000f9`172ff860 00000000`00000000 ntdll!RtlUserThreadSt
```

0:000>

Ln 3, Col 1 | Sys 0:<Local> | Proc 000:1197c | Thrd 000:10d6c | ASM | OVR | CAPS | NUM

t.exe - WinDbg:10.0.10586.567 AMD64

File Edit View Debug Window Help

C:\src\llvm\build\t.cpp

```
#include <stdio.h>
struct IntPair { int x, y; };
int __declspec(noinline) g(int r) { return r + 1; }
int i, n = 4;
volatile int v;
static inline int loop_csr() {
  struct IntPair o = {0, 0};
  for (i = 0; i < n; i++) {
    o.x = g(o.x);
    o.y = g(o.y);
  }
  return o.x + o.y;
}
int main() {
  return loop_csr();
}
```

Command

```
05 000000f9`172ff830 00007ffa`fd87c5b4 KERNEL32!BaseThreadIr
06 000000f9`172ff860 00000000`00000000 ntdll!RtlUserThreadSt
0:000> t
t!loop_csr+0x2f [inlined in t!main+0x37]:
00007ff6`a8f01047 89c7            mov     edi,eax
0:000> t
t!loop_csr+0x31 [inlined in t!main+0x39]:
00007ff6`a8f01049 89f1            mov     ecx,esi
0:000> t
t!g:
00007ff6`a8f01000 8d4101          lea     eax,[rcx+1]
0:000> t
t!loop_csr+0x38 [inlined in t!main+0x40]:
00007ff6`a8f01050 89c6            mov     esi,eax
0:000> t
t!loop_csr+0x3a [inlined in t!main+0x42]:
00007ff6`a8f01052 ffcd            dec     ebp
0:000> dt o
Local var Type IntPair
   +0x000 x                 : 0n1 (edi)
   +0x004 y                 : 0n1 (esi)
```

0:000>

Ln 8, Col 1 | Sys 0:<Local> | Proc 000:1197c | Thrd 000:10d6c | ASM | OVR | CAPS | NUM

t.exe - WinDbg:10.0.10586.567 AMD64

File Edit View Debug Window Help

C:\src\llvm\build\t.cpp

```
#include <stdio.h>
struct IntPair { int x, y; };
int __declspec(noinline) g(int r) { return r + 1; }
int i, n = 4;
volatile int v;
static inline int loop_csr() {
  struct IntPair o = {0, 0};
  for (i = 0; i < n; i++) {
    o.x = g(o.x);
    o.y = g(o.y);
  }
  return o.x + o.y;
}
int main() {
  return loop_csr();
}
```

Command

```
00007ff6`a8f01050 89c6            mov     esi,eax
0:000> t
t!loop_csr+0x3a [inlined in t!main+0x42]:
00007ff6`a8f01052 ffcd            dec     ebp
0:000> dt o
Local var Type IntPair
   +0x000 x                 : 0n1 (edi)
   +0x004 y                 : 0n1 (esi)
0:000> p
t!loop_csr+0x28 [inlined in t!main+0x30]:
00007ff6`a8f01040 89f9            mov     ecx,edi
0:000> p
t!loop_csr+0x31 [inlined in t!main+0x39]:
00007ff6`a8f01049 89f1            mov     ecx,esi
0:000> p
t!loop_csr+0x3a [inlined in t!main+0x42]:
00007ff6`a8f01052 ffcd            dec     ebp
0:000> dt o
Local var Type IntPair
   +0x000 x                 : 0n2 (edi)
   +0x004 y                 : 0n2 (esi)
```

0:000>

Ln 8, Col 1 | Sys 0:<Local> | Proc 000:1197c | Thrd 000:10d6c | ASM | OVR | CAPS | NUM

t.exe - WinDbg:10.0.10586.567 AMD64

File Edit View Debug Window Help

C:\src\llvm\build\t.cpp

```
#include <stdio.h>
struct IntPair { int x, y; };
int __declspec(noinline) g(int r) { return r + 1; }
int i, n = 4;
volatile int v;
static inline int loop_csr() {
  struct IntPair o = {0, 0};
  for (i = 0; i < n; i++) {
    o.x = g(o.x);
    o.y = g(o.y);
  }
  return o.x + o.y;
}
int main() {
  return loop_csr();
}
```

Command

```
00007ff6`a8f01049 89f1            mov     ecx,esi
0:000> p
t!loop_csr+0x3a [inlined in t!main+0x42]:
00007ff6`a8f01052 ffcd            dec     ebp
0:000> dt o
Local var Type IntPair
   +0x000 x                 : 0n2 (edi)
   +0x004 y                 : 0n2 (esi)
0:000> p
t!loop_csr+0x28 [inlined in t!main+0x30]:
00007ff6`a8f01040 89f9            mov     ecx,edi
0:000> p
t!loop_csr+0x31 [inlined in t!main+0x39]:
00007ff6`a8f01049 89f1            mov     ecx,esi
0:000> p
t!loop_csr+0x3a [inlined in t!main+0x42]:
00007ff6`a8f01052 ffcd            dec     ebp
0:000> dt o
Local var Type IntPair
   +0x000 x                 : 0n3 (edi)
   +0x004 y                 : 0n3 (esi)
```

0:000>

Ln 8, Col 1 | Sys 0:<Local> | Proc 000:1197c | Thrd 000:10d6c | ASM | OVR | CAPS | NUM

t.exe - WinDbg:10.0.10586.567 AMD64

File Edit View Debug Window Help

C:\src\llvm\build\t.cpp

```
#include <stdio.h>
struct IntPair { int x, y; };
int __declspec(noinline) g(int r) { return r + 1; }
int i, n = 4;
volatile int v;
static inline int loop_csr() {
  struct IntPair o = {0, 0};
  for (i = 0; i < n; i++) {
    o.x = g(o.x);
    o.y = g(o.y);
  }
  return o.x + o.y;
}
int main() {
  return loop_csr();
}
```

Command

```
00007ff6`a8f01049 89f1            mov     ecx,esi
0:000> p
t!loop_csr+0x3a [inlined in t!main+0x42]:
00007ff6`a8f01052 ffcd            dec     ebp
0:000> dt o
Local var Type IntPair
   +0x000 x                 : 0n3 (edi)
   +0x004 y                 : 0n3 (esi)
0:000> p
t!loop_csr+0x28 [inlined in t!main+0x30]:
00007ff6`a8f01040 89f9            mov     ecx,edi
0:000> p
t!loop_csr+0x31 [inlined in t!main+0x39]:
00007ff6`a8f01049 89f1            mov     ecx,esi
0:000> p
t!loop_csr+0x3a [inlined in t!main+0x42]:
00007ff6`a8f01052 ffcd            dec     ebp
0:000> dt o
Local var Type IntPair
   +0x000 x                 : 0n4 (edi)
   +0x004 y                 : 0n4 (esi)
```

0:000>

Ln 8, Col 1 | Sys 0:<Local> | Proc 000:1197c | Thrd 000:10d6c | ASM | OVR | CAPS | NUM

Optimized debug info works!

Optimized debug info works!

... but optimized debug info needs more work

Hopefully today's BoF was productive

What about LLD?

- Why use LLD:
  - Enables LTO
  - Twice as fast as MSVC
- PDB writing in LLD is under development
- Building YAML roundtripping into llvm-pdbdump

Takeaways and lessons

- Clang/LLVM CodeView support is feature complete, use it and file bugs!
  - PDB support in LLD is coming soon

Takeaways and lessons

- Clang/LLVM CodeView support is feature complete, use it and file bugs!
  - PDB support in LLD is coming soon
- Three debug info linking optimization techniques:
  - Merge the type graph with DAGs
  - Type server optimization
  - COMDAT elimination for symbol info

Takeaways and lessons

- Clang/LLVM CodeView support is feature complete, use it and file bugs!
  - PDB support in LLD is coming soon
- Three debug info linking optimization techniques:
  - Merge the type graph with DAGs
  - Type server optimization
  - COMDAT elimination for symbol info
- LLVM should reuse the type merging algorithm for:
  - IR types, DI type metadata, and DWARF types

Takeaways and lessons

- Clang/LLVM CodeView support is feature complete, use it and file bugs!
  - PDB support in LLD is coming soon
- Three debug info linking optimization techniques:
  - Merge the type graph with DAGs
  - Type server optimization
  - COMDAT elimination for symbol info
- LLVM should reuse the type merging algorithm for:
  - IR types, DI type metadata, and DWARF types

Questions?