COMPILER C | R RESEARCH

HSF-INDIA

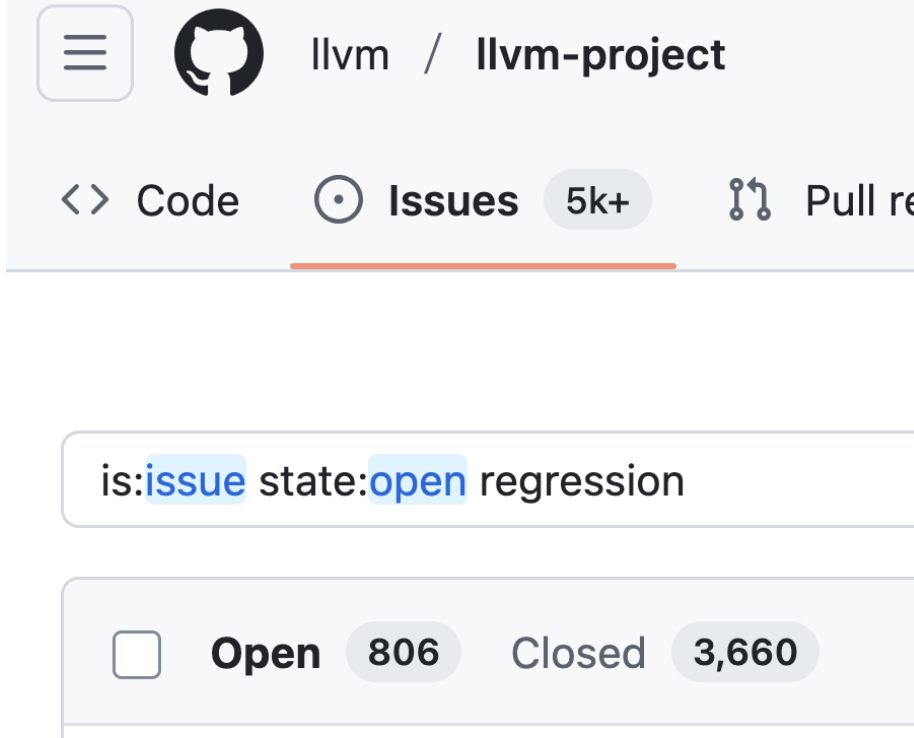
Debugging Regressions: Interactive Differential Debugging

VIPUL CARIAPPA, MARTIN VASSILEV, ALEXANDER PENEV, VASSIL VASSILEV

This project is partially supported by the National Science Foundation under award OISE-2201990

Problem

- Modern software systems are complex, with millions of lines of code, making debugging difficult.
- Differential debugging simplifies the process by comparing the current system to a previous version as a baseline.
- Current debugging practice involves running two separate debugger instances without communication about their execution states.

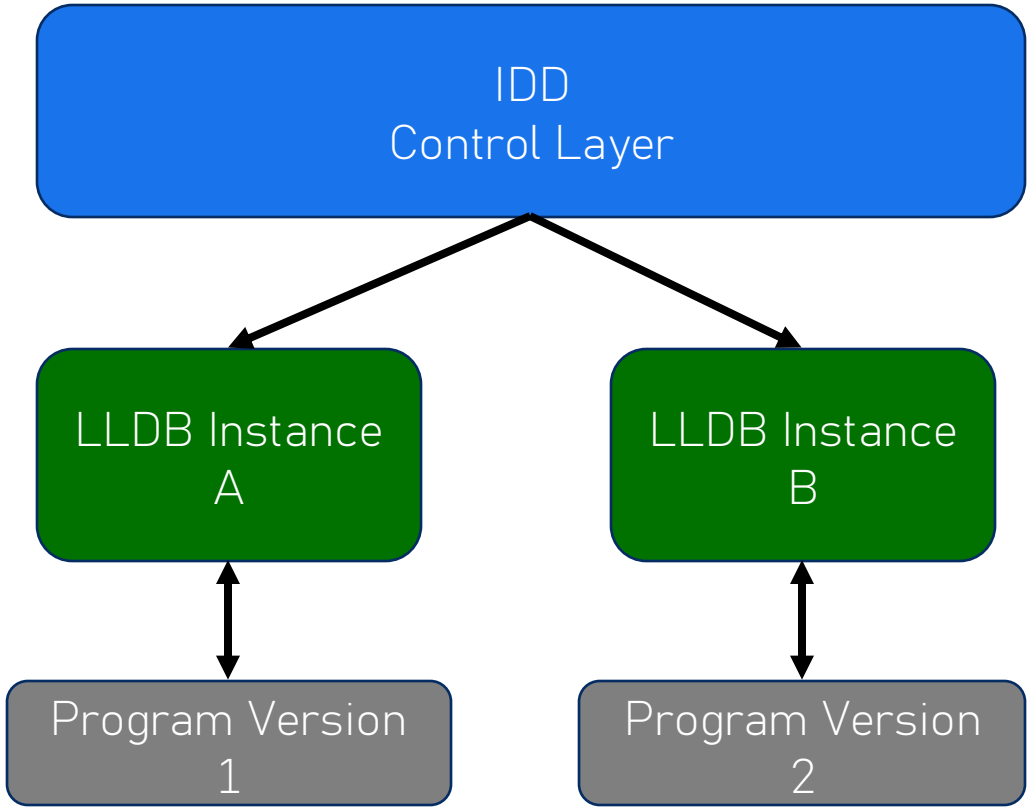


IDD: Interactive Differential Debugging

- IDD automates the process of filtering out irrelevant execution paths between a reference and the regressed software system.
- IDD loads two versions of the system
  - Base: The version of the system we expect to be fine.
  - Regressed: The version that has a regression introduced in it.
- IDD uses LLDB or GDB to inspect both versions of the system simultaneously.
- IDD offers diff-view to look at the differences between the states of both systems.
- IDD helps find the regression faster by ignoring common/irrelevant execution paths

IDD Architecture

- IDD is the core controller, issuing commands to both LLDB instances.
- Each LLDB instance debugs a different version of the same program.
- IDD synchronizes their execution and filters out shared state to highlight differences.



DiffDebug

Base Version View

```
055556128f73e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
055556128d88e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
55556128ae84 clang++`clang::Parser::ParseAssignmentExpression(clang::Pars
0555561278c44 clang++`clang::Parser::ParseInitializer() at Parser.h:2119
555561 5c8bb clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAtt
555561 25b4f4 clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec
555561 259ede clang++`clang::Parser::ParseSimpleDeclaration(clang::Declara
```

```
Base Locals
(clang::ExprResult)Res=None
(clang::tok::TokenKind)SavedKind=unknown
(clang::PreferredTypeBuilder)SavedType=Non
(bool)AllowSuffix=false
```

```
Base Args
- (clang::Parser *)this=0x00005555647486 00
(clang::Parser::CastParseKind)ParseKind=An
(bool)isAddressOfOperand=false
- (bool &)NotCastExpr=0x00007fffffff72b7
(clang::Parser::TypeCastState)isTypeCast=N
(bool)isVectorLiteral=false
(bool *)NotPrimaryExpression=0x00000000000
```

```
Enter your base command here...
```

```
Base Diff
- -> 712          ExprResult Res = ParseCastExpression(ParseKind,
                                                      ^
-    713                                                isAddressOfOperand,
-    714                                                NotCastExpr,
-    715                                                isTypeCast,
s
- Process 6783 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
-      frame #0: 0x000055556128f73e clang++`clang::Parser::ParseCastExpression(this=0x00005
-    1052                                               TypeCastState isTypeCast,
-    1053                                               bool isVectorLiteral,
-    1054                                               bool *NotPrimaryExpression) {
- -> 1055         ExprResult Res;
                    ^
-    1056         tok::TokenKind SavedKind = Tok.getKind();
-    1057         auto SavedType = PreferredType;
-    1058         NotCastExpr = false;
```

Regressed Version View

```
Regression StackFrame
+    frame #0: 0x00555561a4f8ec clang++`clang::Parser::ParseCastExpression(
+    frame #1: 0x00555561a4da94 clang++`clang::Parser::ParseCastExpression(
+   frame #2: 0x00555561a4b062 clang++`clang::Parser::ParseAssignmentExpres
+    frame #3: 0x00555561a38f5c clang++`clang::Parser::ParseInitializer() a
+   frame #4: 0x00555561a1ca26 clang++`clang::Parser::ParseDeclarationAfter
+   frame #5: 0x00555561a1b64b clang++`clang::Parser::ParseDeclGroup(clang:
+   frame #6: 0x00555561a1a03e clang++`clang::Parser::ParseSimpleDeclaratio
```

```
Regression Locals
(clang::ExprResult)Res=None
(clang::tok::TokenKind)SavedKind=unknown
(clang::PreferredTypeBuilder)SavedType=Non
(bool)AllowSuffix=false
```

```
Regression Args
+ (clang::Parser *)this=0x00005555648901a0
(clang::Parser::CastParseKind)ParseKind=Any
(bool)isAddressOfOperand=false
+ (bool &)NotCastExpr=0x00007fffffff03f7
(clang::Parser::TypeCastState)isTypeCast=No
(bool)isVectorLiteral=false
(bool *)NotPrimaryExpression=0x000000000000
```

Single instance commands

```
Enter your regression command here...
```

```
Regression Diff
+ -> 729          ExprResult Res = ParseCastExpression(ParseKind,
                                                      ^
+    730                                                isAddressOfOperand,
+    731                                                NotCastExpr,
+    732                                                isTypeCast,
s
+ Process 6782 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
+      frame #0: 0x0000555561a4f8ec clang++`clang::Parser::ParseCastExpression(this=0x000055
+    1068                                               TypeCastState isTypeCast,
+    1069                                               bool isVectorLiteral,
+    1070                                               bool *NotPrimaryExpression) {
+ -> 1071         ExprResult Res;
                    ^
+    1072         tok::TokenKind SavedKind = Tok.getKind();
+    1073         auto SavedType = PreferredType;
+    1074         NotCastExpr = false;
```

```
n
```

Commands to both instances

^p palette

Diff Pane Frames

Diff Pane Variables

Diff Pane Execution

DiffDebug

Base Stackframe

```
-    frame #0: 0x0055556128f73e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-    frame #1: 0x0055556128d88e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-   frame #2: 0x0055556128ae84 clang++`clang::Parser::ParseAssignmentExpression(clang::Pars
-    frame #3: 0x00555561278c44 clang++`clang::Parser::ParseInitializer() at Parser.h:2119
-   frame #4: 0x0055556125c8bb clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAtt
-   frame #5: 0x0055556125b4f4 clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec
-   frame #6: 0x00555561259ede clang++`clang::Parser::ParseSimpleDeclaration(clang::Declara
```

Regression Stackframe

```
+    frame #0: 0x00555561a4f8ec clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+    frame #1: 0x00555561a4da94 clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+   frame #2: 0x00555561a4b062 clang++`clang::Parser::ParseAssignmentExpression(clang::Parse
+    frame #3: 0x00555561a38f5c clang++`clang::Parser::ParseInitializer() at Parser.h:2125 (
+   frame #4: 0x00555561a1ca26 clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAttr
+   frame #5: 0x00555561a1b64b clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec&
+   frame #6: 0x00555561a1a03e clang++`clang::Parser::ParseSimpleDeclaration(clang::Declarat
```

Base Locals

```
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBuilder)SavedType=Non
  (bool)AllowSuffix=false
```

Base Args

```
- (clang::Parser *)this=0x0000555564748600
  (clang::Parser::CastParseKind)ParseKind=An
  (bool)isAddressOfOperand=false
- (bool &)NotCastExpr=0x00007fffffff72b7
  (clang::Parser::TypeCastState)isTypeCast=N
  (bool)isVectorLiteral=false
  (bool *)NotPrimaryExpression=0x00000000000
```

Regression Locals

```
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBuilder)SavedType=Non
  (bool)AllowSuffix=false
```

Regression Args

```
+ (clang::Parser *)this=0x000055556489 01a0
  (clang::Parser::CastParseKind)ParseKind=Any
  (bool)isAddressOfOperand=false
+ (bool &)NotCastExpr=0x00007fffffff03f7
  (clang::Parser::TypeCastState)isTypeCast=No
  (bool)isVectorLiteral=false
  (bool *)NotPrimaryExpression=0x000000000000
```

Enter your base command here...

Enter your regression command here...

Base diff

```
- -> 712          ExprResult Res = ParseCastExpression(ParseKind,
                                                       ^
-    713                                                isAddressOfOperand,
-    714                                                NotCastExpr,
-    715                                                isTypeCast,
s
- Process 6783 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
-      frame #0: 0x00005555 6128f73e clang++`clang::Parser::ParseCastExpression(this=0x00005
-    1052                                               TypeCastState isTypeCast,
-    1053                                               bool isVectorLiteral,
-    1054                                               bool *NotPrimaryExpression) {
- -> 1055          ExprResult Res;
                      ^
-    1056          tok::TokenKind SavedKind = Tok.getKind();
-    1057          auto SavedType = PreferredType;
-    1058          NotCastExpr = false;
```

Regression diff

```
+ -> 729          ExprResult Res = ParseCastExpression(ParseKind,
                                                       ^
+    730                                                isAddressOfOperand,
+    731                                                NotCastExpr,
+    732                                                isTypeCast,
s
+ Process 6782 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
+      frame #0: 0x0000555561a4f8ec clang++`clang::Parser::ParseCastExpression(this=0x000055
+    1068                                               TypeCastState isTypeCast,
+    1069                                               bool isVectorLiteral,
+    1070                                               bool *NotPrimaryExpression) {
+ -> 1071          ExprResult Res;
                      ^
+    1072          tok::TokenKind SavedKind = Tok.getKind();
+    1073          auto SavedType = PreferredType;
+    1074          NotCastExpr = false;
```

n

^p palette

Demo

Future Work

- Improved semantic diff. on pointers across instances
- Automatically break at diverging stack frames
- Watchpoints for diverging variables values of interest
- Better gdb support

Thank You

🔗 GitHub: github.com/compiler-research/idd

📦 PyPI: pypi.org/project/idd

📌 Install via: **pip install idd**