



# 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.



### Solution: Interactive Differential Debugging (idd)

IDD automates the process of filtering out irrelevant execution paths between a reference and the regressed software system.

#### How does it work?

- Load 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.
- Use LLDB/GDB to inspect both versions of the system simultaneously.
- Leverage diff-view to look at the differences between the states of both systems.
- Deduce the cause of the regression faster by ignoring common/irrelevant execution paths.

The result is a focused display of debugger states that differ between the two versions.

#### Architecture

- Common place for data exchange between two debuggers. Works with either lldb or gdb.
  - Using Ildb python API to steer the Ildb debugger.
  - Similar but less powerful approach with gdb due to the limited API supporting tools.
- Git style difference viewer to easily recognize differences between two versions.
- Organizable UI with CSS to concentrate on what matter to you.

o DiffDebug				
<pre>- Base StateKTrame- - 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: 0x005555612<u>5c8bb</u> clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAtt - frame #5: 0x005555612<u>5b4f4</u> clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec - frame #6: 0x005555612<u>59ed</u>e clang++`clang::Parser::ParseSimpleDeclaration(clang::Declara - Base Locals (clang::ExprResult)Res=None</pre>		<pre>+ frame #0: 0x005555 + frame #1: 0x005555 + frame #1: 0x0055556 + frame #3: 0x0055556 + frame #4: 0x0055556 + frame #5: 0x0055566 + frame #6: 0x0055566 + frame #6: 0x0055566 - Regression Locals - (clang::ExprResult)Ref ( clang::ExprResult)Ref ( clang::ExprResult)Re</pre>	561a4f8ec clang++`clang::Parser:: 561a4da94 clang++`clang::Parser:: 51a4b062 clang++`clang::Parser:: 561a38f5c clang++`clang::Parser:: 51a1ca2 <u>6</u> clang++`clang::Parser:: 51 <mark>a1b64b</mark> clang++`clang::Parser:: 51 <mark>a1a03</mark> e clang++`clang::Parser:: 51a1a03e clang++`clang++`clang::Parser:: 51a1a03e clang++`clang++`clang++`clang+++`clang+++`clang++++++++++++++++++++++++++++++++++++	ParseCastExpression(clang::Parser::Ca ParseCastExpression(clang::Parser::Ca ParseAssignmentExpression(clang::Parse ParseInitializer() at Parser.h:2125 ( ParseDeclarationAfterDeclaratorAndAttr ParseDeclGroup(clang::ParsingDeclSpec& ParseSimpleDeclaration(clang::Declarat estimates and the set of th
(clang::PreferredTypeBu; (bool)AllowSuffix=false Base Version View	<pre>isAddressOfOperand=false     S)NotCastExpr=0x00007fffffff72b7     S::Parser::TypeCastState)isTypeCast=N     isVectorLiteral=false     *)NotPrimaryExpression=0x0000000000</pre>	(clang::PreferredType (bool)AllowSuffix=fa	Regressed Version View	AddressOfOperand=false otCastExpr=0x00007fffffff03f7 arser::TypeCastState)isTypeCast=No ectorLiteral=false otPrimaryExpression=0x000000000000
Enter your base command here		Enter your regression		
Base Diff		- Regression Diff		
> /12 EXPRRESULT Res = ParsecastExpression ^	i(Parsekind,	+ -> /29 Expri	<pre>kesult kes = Parsecastexpression(</pre>	Parsekind,
- 713	isAddressOfOperand,	+ 730		isAddressOfOperand,
- 714	NotCastExpr,	+ 731		NotCastExpr,
- 7 <u>15</u>	isTypeCast,	+ 7 <u>32</u>		isTypeCast,
- Process 6783 stopped		S		
		+ Process 0/84 slopped		
* thread #1, name = 'clang++', stop reason = breakp	pint 1.1	<pre>+ Process 6782 scopped     * thread #1, name =</pre>	'clang++', stop reason = breakpoi	int 1.1
<pre>* thread #1, name = 'clang++', stop reason = breakp - frame #0: 0x000055556128f73e clang++`clang::Pa 1052</pre>	pint 1.1 :ser::ParseCastExpression(this=0x00005	+ process 0782 scopped + thread #1, name = + frame #0: 0x0000	'clang++', stop reason = breakpoi 0555561a4f8ec clang++`clang::Pars	int 1.1 ser::ParseCastExpression(this=0x000055
<pre>* thread #1, name = 'clang++', stop reason = breakp - frame #0: 0x000055556128f73e clang++`clang::Pa - 10<u>52</u> - 1053</pre>	pint 1.1 rser::ParseCastExpression(this=0x00005 TypeCastState isTypeCast, bool isVectorLiteral.	<pre>+ process 0782 scopped  * thread #1, name = + frame #0: 0x0000 + 1068 + 1069</pre>	'clang++', stop reason = breakpo: 0555561a4f8ec clang++`clang::Pars	int 1.1 ser::ParseCastExpression(this=0x000055 TypeCastState isTypeCast, bool isVectorLiteral.
<pre>* thread #1, name = 'clang++', stop reason = breakp - frame #0: 0x000055556128f73e clang++`clang::Pa - 10<u>52</u> - 10<u>53</u> - 10<u>54</u></pre>	<pre>pint 1.1 tser::ParseCastExpression(this=0x00005 TypeCastState isTypeCast,     bool isVectorLiteral,     bool *NotPrimaryExpression) {</pre>	<pre>+ process 0782 scopped  * thread #1, name = + frame #0: 0x0000 + 1068 + 1069 + 1070</pre>	'clang++', stop reason = breakpo: 0555561a4f8ec clang++`clang::Pars	int 1.1 ser::ParseCastExpression(this=0x000055 TypeCastState isTypeCast, bool isVectorLiteral, bool *NotPrimaryExpression) {
<pre>* thread #1, name = 'clang++', stop reason = breakp - frame #0: 0x000055556128f73e clang++`clang::Pa - 10<u>52</u> - 10<u>53</u> - 10<u>54</u> &gt; 10<u>55</u> ExprResult Res;</pre>	<pre>bint 1.1 rser::ParseCastExpression(this=0x00005 TypeCastState isTypeCast,     bool isVectorLiteral,     bool *NotPrimaryExpression) {    </pre>	<pre>* thread #1, name = + frame #0: 0x0000 + 1068 + 1069 + 1070 + -&gt; 1071 Expri </pre>	'clang++', stop reason = breakpo: 0555561a4f8ec clang++`clang::Pars Result Res;	int 1.1 ser::ParseCastExpression(this=0x000055 TypeCastState isTypeCast, bool isVectorLiteral, bool *NotPrimaryExpression) {
<pre>* thread #1, name = 'clang++', stop reason = breakp - frame #0: 0x000055556128f73e clang++`clang::Pa - 1052 - 1053 - 1054 &gt; 1055 ExprResult Res; ^ - 1056 tok::TokenKind SavedKind = Tok getK</pre>	<pre>bint 1.1 rser::ParseCastExpression(this=0x00005 TypeCastState isTypeCast,     bool isVectorLiteral,     bool *NotPrimaryExpression) {     ind():</pre>	+ process 0.82 scopped * thread #1, name = + frame #0: 0x0000 + 1068 + 1069 + 1070 + -> 1071 Expri + 1072 tok:	'clang++', stop reason = breakpo 0555561a4f8ec clang++`clang::Pars Result Res; :TokenKind SavedKind = Tok.getKir	int 1.1 ser::ParseCastExpression(this=0x000055 TypeCastState isTypeCast, bool isVectorLiteral, bool *NotPrimaryExpression) {
<pre>* thread #1, name = 'clang++', stop reason = breakp - frame #0: 0x000055556128f73e clang++`clang::Pa - 1052 - 1053 - 1054 &gt; 1055 ExprResult Res; ^ - 1056 tok::TokenKind SavedKind = Tok.getK - 1057 auto SavedType = PreferredType;</pre>	<pre>bint 1.1 rser::ParseCastExpression(this=0x00005 TypeCastState isTypeCast,     bool isVectorLiteral,     bool *NotPrimaryExpression) { ind();</pre>	<pre>+ process 0.82 scopped  * thread #1, name = + frame #0: 0x0000 + 1068 + 1069 + 1070 + -&gt; 1071 Exprf</pre>	'clang++', stop reason = breakpo: 0555561a4f8ec clang++`clang::Pars Result Res; :TokenKind SavedKind = Tok.getKin SavedType = PreferredType;	int 1.1 ser::ParseCastExpression(this=0x000055 TypeCastState isTypeCast, bool isVectorLiteral, bool *NotPrimaryExpression) {
<pre>* thread #1, name = 'clang++', stop reason = breakp - frame #0: 0x000055556128f73e clang++`clang::Pa - 1052 - 1053 - 1054 &gt; 1055 ExprResult Res; ^ - 1056 tok::TokenKind SavedKind = Tok.getK - 1057 auto SavedType = PreferredType; - 1058 NotCastExpr = false;</pre>	<pre>oint 1.1 rser::ParseCastExpression(this=0x00005 TypeCastState isTypeCast,    bool isVectorLiteral,    bool *NotPrimaryExpression) { ind();</pre>	<pre>+ process 0782 scopped  * thread #1, name = + frame #0: 0x0000 + 1068 + 1069 + 1070 + -&gt; 1071 Exprf  + 1072 tok: + 1073 auto + 1074 NotCa</pre>	'clang++', stop reason = breakpo 2555561a4f8ec clang++`clang::Pars Result Res; :TokenKind SavedKind = Tok.getKin SavedType = PreferredType; astExpr = false;	int 1.1 ser::ParseCastExpression(this=0x000055 TypeCastState isTypeCast, bool isVectorLiteral, bool *NotPrimaryExpression) { nd();
<pre>* thread #1, name = 'clang++', stop reason = breakp</pre>	<pre>oint 1.1 rser::ParseCastExpression(this=0x00005 TypeCastState isTypeCast,    bool isVectorLiteral,    bool *NotPrimaryExpression) { ind();</pre>	<pre>* thread #1, name = + frame #0: 0x0000 + 1068 + 1069 + 1070 + -&gt; 1071 Exprf ^ + 1072 tok:: + 1073 auto + 1074 NotCa</pre>	'clang++', stop reason = breakpo: 2555561a4f8ec clang++`clang::Pars Result Res; :TokenKind SavedKind = Tok.getKin SavedType = PreferredType; astExpr = false;	int 1.1 ser::ParseCastExpression(this=0x000055 TypeCastState isTypeCast, bool isVectorLiteral, bool *NotPrimaryExpression) {

o DiffDebug			
<ul> <li>frame #0: 0x0055556128f73e clang++`clang::Parser::ParseCastExpression(clang::Parser:: frame #1: 0x0055556128d88e clang++`clang::Parser::ParseCastExpression(clang::Parser:: frame #2: 0x0055556128ae84 clang++`clang::Parser::ParseAssignmentExpression(clang::Parser: frame #3: 0x00555561278c44 clang++`clang::Parser::ParseInitializer() at Parser.h:2119</li> <li>frame #4: 0x0055556125642564 clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAt</li> <li>frame #5: 0x0055556125644 clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpe- frame #6: 0x00555561259ed clang++`clang::Parser::ParseSimpleDeclaration(clang::DeclaratorAndAt</li> </ul>	<pre>+ 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::ParsingDeclSpec8 + frame #6: 0x00555561a1a03 e clang++`clang::Parser::ParseSimpleDeclaration(clang::Declarat</pre>		
<pre>(clang::ExprResult)Res=None (clang::tok::TokenKind)SavedKind=unknown (clang::PreferredTypeBuilder)SavedType=Non (bool)AllowSuffix=false</pre> - (clang::Parser::CastParseKind)ParseKind=A (bool)isAddressOfOperand=false - (bool &)NotCastExpr=0x00007f (clang::Parser::TypeCastSta (bool)isVectorLiteral=false (bool *)NotPrimaryExpressio Stack	<pre>(clang::ExprR.sult)Res=None (clang::tok::TokenKind)SavedKind=unknown (clang::TreferredTypeBuilder)SavedType=Non ) ix=false (bool &amp;)NotCastExpr=0x00007ffffff03f7 (clang::Parser::TypeCastState)isTypeCast=No (bool)isVectorLiteral=false (bool *)NotPrimaryExpression=0x00000000000</pre>		
Enter your base command here	Enter your regression command here		
Base Diff  > 712 ExprResult Res = ParseCastExpression(ParseKind,	Regression Diff + -> 729 ExprResult Res = ParseCastExpression(ParseKind,		
<ul> <li>713 isAddressOfOperand,</li> <li>714 NotCastExpr,</li> <li>715 isTypeCast,</li> <li>8</li> <li>Process 6783 stopped</li> </ul>	+ 730 isAddressOfOperand, + 731 NotCastExpr, + 732 isTypeCast, s + Process 6782 stopped		
<pre>* thread #1, name = 'clang++', stop reason = breakpoint 1.1 - frame #0: 0x000055556128f73e clang++`clang::Parser::ParseCastExpression(this=0x0000 - 1052 TypeCastState isTypeCast, - 1053 bool isVectorLiteral, - 1054 bool *NotPrimaryExpression) {&gt; 1055 ExprResult Res;</pre>	<pre>* 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) { + -&gt; 1071 ExprResult Res; ^</pre>		
<ul> <li>1056 tok::TokenKind SavedKind = Tok.getKind();</li> <li>1057 auto SavedType = PreferredType;</li> <li>1058 NotCastExpr = false;</li> </ul>	<pre>+ 1072 tok::TokenKind SavedKind = Tok.getKind(); + 1073 auto SavedType = PreferredType; + 1074 NotCastExpr = false;</pre>		

0		DiffDebug	
<pre>Base Stackframe frame #0: 0x0055556128f73e clang++`clang frame #1: 0x0055556128d88e clang++`clang frame #2: 0x0055556128ae84 clang++`clang: frame #3: 0x00555561278c44 clang++`clang frame #4: 0x0055556125c8bb clang++`clang: frame #5: 0x0055556125b4f4 clang++`clang: frame #6: 0x00555561259ede clang++`clang:</pre>	::Parser::ParseCastExpression(clang:: :Parser::ParseCastExpression(clang:: :Parser::ParseAssignmentExpression(cl :Parser::ParseInitializer() at Parse :Parser::ParseDeclarationAfterDeclara :Parser::ParseDeclGroup(clang::Parsin :Parser::ParseSimpleDeclaration(clang	<pre>Regression Stackframe Parser::C Parser::C ang::Pars er.h:2119 torAndAtt gDeclSpec g::Declara Regression Stackframe + frame #0: 0x00555561a468ec clang++`cla + frame #1: 0x00555561a46062 clang++`cla + frame #3: 0x00555561a38f5c clang++`cla + frame #4: 0x00555561a1ca26 clang++`cla + frame #5: 0x00555561a1b64b clang++`cla + frame #6: 0x00555561a1a03 e clang++`cla </pre>	<pre>ang::Parser::ParseCastExpression(clang::Parser::Ca ang::Parser::ParseCastExpression(clang::Parser::Ca ing::Parser::ParseAssignmentExpression(clang::Parse ang::Parser::ParseInitializer() at Parser.h:2125 ( ing::Parser::ParseDeclarationAfterDeclaratorAndAttr ing::Parser::ParseDeclGroup(clang::ParsingDeclSpec&amp; ing::Parser::ParseSimpleDeclaration(clang::Declarat</pre>
<pre>Base Locals (clang::ExprResult)Res=None (clang::tok::TokenKind)SavedKind=unknown (clang::PreferredTypeBuilder)SavedType=Non (bool)AllowSuffix=false</pre>	<ul> <li>- (clang::Parser *)this=0x000055556 (clang::Parser::CastParseKind)Par (bool)isAddressOfOperand=false</li> <li>- (bool &amp;)NotCastExpr=0x00007ffffff (clang::Parser::TypeCastState)isT (bool)isVectorLiteral=false</li> <li>(bool *)NotPrimaryExpression=0x00</li> </ul>	64748600       (clang::ExprResult)Res=None         (clang::tok::TokenKind)SavedKind=unknown         (clang::PreferredTypeBuilder)SavedType=N         (cfool)AllowSuffix=false         0000000000	<pre>+ (clang::Parser *)this=0x00005555648901a0 (clang::Parser::CastParseKind)ParseKind=Any (bool)isAddressOfOperand=false + (bool &amp;)NotCastExpr=0x00007fffffff03f7 (clang::Parser::TypeCastState)isTypeCast=No (bool)isVectorLiteral=false (bool *)NotPrimaryExpression=0x00000000000</pre>
Enter your base command here		Enter vour decression command here	
> 712 ExprResult Res = ParseCastE	xpression(ParseKind,	prResult Res = ParseCa	stExpression(ParseKind,
- 713 - 714 - 7 <u>15</u> S	<pre>isAddressOfOperand, NotCastExpr, isTypeCast,</pre>	Arguments	<pre> isAddressOfOperand, NotCastExpr, isTypeCast,</pre>
<pre>* thread #1, name = 'clang++', stop reason = - frame #0: 0x000055556128f73e clang++`c' - 1052 - 1053 - 1054&gt; 1055 ExprResult Res; - 1056 tok::TokenKind SavedKind = ' - 1057 auto SavedType = PreferredTy - 1058 NotCastExpr = false;</pre>	<pre>= breakpoint 1.1 lang::Parser::ParseCastExpression(thi</pre>	<pre>&gt;</pre>	<pre>ion = breakpoint 1.1 +`clang::Parser::ParseCastExpression(this=0x000055</pre>



<pre>from #1: 00000000000000000000000000000000000</pre>	o D1+fDebug		
Part Curdit         Clang::Expression(basel)         Clang::Expression(basel)         Clang::Expression(basel)         Clang::Expression(basel)         Clang::PreferredTypebulter)         SwedKind=unknown (clang::PreferredTypebulter)         Clang::PreferredTypebulter)         PreferredTypebulter)         StaddressOfOperand, * Tria <t< th=""><th>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::ParseCastExpression(clang::Parser::C frame #2: 0x0055556128ae84 clang++`clang::Parser::ParseCastExpression(clang::Parser::C frame #3: 0x00555561278c44 clang++`clang::Parser::ParseInitializer() at Parser.h:2119 frame #4: 0x0055556125612564 clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAtt frame #5: 0x005555612564f4 clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec frame #6: 0x00555561259ede clang++`clang::Parser::ParseSimpleDeclaration(clang::Declarat</th><th colspan="2"><pre>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&amp; + frame #6: 0x00555561a1a03e clang++`clang::Parser::ParseSimpleDeclaration(clang::Declarat</pre></th></t<>	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::ParseCastExpression(clang::Parser::C frame #2: 0x0055556128ae84 clang++`clang::Parser::ParseCastExpression(clang::Parser::C frame #3: 0x00555561278c44 clang++`clang::Parser::ParseInitializer() at Parser.h:2119 frame #4: 0x0055556125612564 clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAtt frame #5: 0x005555612564f4 clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec frame #6: 0x00555561259ede clang++`clang::Parser::ParseSimpleDeclaration(clang::Declarat	<pre>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&amp; + frame #6: 0x00555561a1a03e clang++`clang::Parser::ParseSimpleDeclaration(clang::Declarat</pre>	
Enter your base command here       Enter your gression command here        > 712       ExprResult Res = ParseCastExpression(ParseKind,         - 713       isAddressOfOperand,         - 714       NotCastExpr,         - 715       isTypeCast,         * -> 715       isTypeCast,         * -> 715       isTypeCast,         * -> rocess 6783 stopped       * thread #1, name = 'clang++', stop reason = breakpoint 1.1         - frame #0: 0x000055556128f73e clang++'clang::ParseciatExpression(this=0x00005         - 1052       TypeCastState isTypeCast,         - 1053       bool *NotPrimaryExpression) {        > 1055       ExprResult Res;        > 1056       tok::TokenKind SavedKind = Tok.getKind();         - 1056       tok::TokenKind SavedKind = Tok.getKind();         - 1057       auto SavedType = PreferredType;         - 1058       NotCastExpr = false;	Base Locals	<pre>None SavedKind=unknown ilder)SavedType=Non + (clang::Parser *)this=0x00005555648g01a0 (clang::Parser::CastParseKind)ParseKind=Any (bool)isAddressOfOperand=false + (bool &amp;)NotCastExpr=0x00007fffffff03f7 (clang::Parser::TypeCastState)isTypeCast=No (bool)isVectorLiteral=false (bool *)NotPrimaryExpression=0x00000000000</pre>	
<pre>&gt; 712 ExprResult Res = ParseCastExpression(ParseKind, - 713 isAddressOfOperand, - 714 NotCastExpr, - 715 isTypeCast, - 716 isTypeCast, - 717 isTypeCast, - 718 isTypeCast, - 718 isTypeCast, - 719 ExprResult Res = ParseCastExpression(ParseKind, - 718 isAddressOfOperand, - 719 istore - 729 ExprResult Res = ParseCastExpression(ParseKind, - 731 NotCastExpr, - 732 isTypeCast, - 732 isT</pre>	Enter your base command here Enter your regression of		
	<pre>-&gt; 712 ExprResult Res = ParseCastExpression(ParseKind, - 713 isAddressOfOperand, - 714 NotCastExpr, - 715 isTypeCast, - 715 isTypeCast, - 715 - 729 ExprRes + 730 + 731 + 731 + 731 + 731 + 731 + 731 + 731 + 731 + 732 +</pre>	ult Res = ParseCastExpression(ParseKind, isAddressOfOperand, NotCastExpr, isTypeCast, ang++', stop reason = breakpoint 1.1 5561a4f8ec clang++`clang::Parser::ParseCastExpression(this=0x000055 TypeCastState isTypeCast, bool isVectorLiteral, bool *NotPrimaryExpression) { ult Res;	

fram #1: def00000012472be change- change: heaves: intrace intervention (change: heaves: intervention (change: heaves: intrace intervention (change: heaves: intervention (change:	0	Ditt	Debug	
<pre>class:fbprResult)Res=None (class::tbprResult]Res=None (class::tbprResult]Res=None (class::tbprResult]Res=None (class::tbprResult]Res=None (bool *)NotPrimaryExpression=Dx0000000000 = NotPrimaryExpression=Dx000000000000000000000000000000000000</pre>	<pre>Base Stackframe frame #0: 0x0055556128f73e clang++`clang frame #1: 0x0055556128d88e clang++`clang frame #2: 0x0055556128ae84 clang++`clang: frame #3: 0x00555561278c44 clang++`clang frame #4: 0x005555612568bb clang++`clang: frame #5: 0x00555561259ede clang++`clang: frame #6: 0x00555561259ede clang++`clang:</pre>	::Parser::ParseCastExpression(clang::Parser::C ::Parser::ParseCastExpression(clang::Parser::C :Parser::ParseAssignmentExpression(clang::Pars ::Parser::ParseInitializer() at Parser.h:2119 :Parser::ParseDeclarationAfterDeclaratorAndAtt :Parser::ParseDeclGroup(clang::ParsingDeclSpec :Parser::ParseSimpleDeclaration(clang::Declara	<pre>Regression Stackframe + frame #0: 0x00555561a4f8ec clang++`clang + frame #1: 0x00555561a4da94 clang++`clang + frame #2: 0x00555561a4b062 clang++`clang: + frame #3: 0x00555561a38f5c clang++`clang + frame #4: 0x00555561a1ca26 clang++`clang: + frame #5: 0x00555561a1b64b clang++`clang: + frame #6: 0x00555561a1a03e clang++`clang:</pre>	g::Parser::ParseCastExpression(clang::Parser::Ca g::Parser::ParseCastExpression(clang::Parser::Ca :Parser::ParseAssignmentExpression(clang::Parse g::Parser::ParseInitializer() at Parser.h:2125 ( :Parser::ParseDeclarationAfterDeclaratorAndAttr :Parser::ParseDeclGroup(clang::ParsingDeclSpec& :Parser::ParseSimpleDeclaration(clang::Declarat
Enter your base command here Enter your base command here Enter your regression command here Enter your regression command here Process 6783 stopped * thread #1, name = 'clang++', stop reason * thread #1, name = 'clang++', stop reason = breakpoint 1.1 * frame #0: 0x00005555128f73e clang++ * 1052 * 1055 * 1055 * process 6782 stopped * thread #1, name = 'clang++', stop reason = breakpoint 1.1 * frame #0: 0x00005555126f73e clang++ * 1068 * 1068 * 1069 * 1070 * 1072 * 1073 * 1073 * 1073 * 1074 * 1075 * 1074 * 1074 * 1075 * 1074 * 1075 * 1074 * 1075 * 10	<pre>Pase Locals (clang::ExprResult)Res=None (clang::tok::TokenKind)SavedKind=unknown (clang::PreferredTypeBuilder)SavedType=Non (bool)AllowSuffix=false</pre>	<ul> <li>Base Args</li> <li>(clang::Parser *)this=0x0000555564748<u>6</u>00 (clang::Parser::CastParseKind)ParseKind=An (bool)isAddressOfOperand=false</li> <li>(bool &amp;)NotCastExpr=0x00007fffffff72b7 (clang::Parser::TypeCastState)isTypeCast=N (bool)isVectorLiteral=false (bool *)NotPrimaryExpression=0x00000000000</li> </ul>	<pre>Regression Locals (clang::ExprResult)Res=None (clang::tok::TokenKind)SavedKind=unknown (clang::PreferredTypeBuilder)SavedType=Non (bool)AllowSuffix=false</pre>	<pre>Progression Args + (clang::Parser *)this=0x00005555648901a0 (clang::Parser::CastParseKind)ParseKind=Any (bool)isAddressOfOperand=false + (bool &amp;)NotCastExpr=0x00007fffffff03f7 (clang::Parser::TypeCastState)isTypeCast=No (bool)isVectorLiteral=false (bool *)NotPrimaryExpression=0x00000000000</pre>
Enter your base command here Enter your base command here Enter your regression command here Enter your regression command here Enter your regression command here Repression Diff + -> 729 ExprResult Res = ParseCastExpression(ParseKind, - 713 - 713 - 714 - 715 - 7				
<ul> <li>-&gt; 712 ExprResult Res = ParseCastExpression(ParseKind,</li> <li>-&gt; 712 isAddressOfOperand,</li> <li>-&gt; 713 isAddressOfOperand,</li> <li>-&gt; 714 isAddressOfOperand,</li> <li>-&gt; 715 isTypeCast,</li> <li>-&gt; 715 isTypeCast,</li> <li>-&gt; 716 isAddressOfOperand,</li> <li>-&gt; 717 isTypeCast,</li> <li>-&gt; 718 isAddressOfOperand,</li> <li>-&gt; 719 isAddressOfOperand,</li> <li>-&gt; 710 isAddressOfOperand,</li> <li>-&gt; 1052 isAddressOfOperand,</li> <li>-&gt; 1055 isAddressOfOperand,</li> <li>-&gt; 1056 isVectoriteral,</li> <li>-&gt; 1057 auto SavedKind = Preferred</li> <li>-&gt; 1058 NotCastExpr = false;</li> </ul>				
<pre>- 713 isAddressOfOperand, NotCastExpr, - 715 isTypeCast, - 716 isAddressOfOperand, NotCastExpr, - 717 isTypeCast, - 718 isAddressOfOperand, NotCastExpr, - 718 isTypeCast, - 720 isTypeCast, - 721 isTypeCast, - 721 isTypeCast, - 721 isTypeCast, - 722 isTypeCast, - 720 isTypeCast</pre>	Enter your base command here		Enter your regression command here	
<pre>* Process or ag stopped * thread #1, name = 'clang++', stop reason - frame #0: 0x000055556128f73e clang++ - 1052 - 1053 - 1054 - 1055 ExprResult Res; - 1056 tok::TokenKind SavedKind = - 1057 auto SavedType = Preferred - 1058 NotCastExpr = false;</pre>	Enter your base command here Base Diff > 712 ExprResult Res = ParseCastE	xpression(ParseKind,	Enter your regression command here Regression Diff + -> 729 ExprResult Res = ParseCastE	xpression(ParseKind,
	Enter your base command here Base Diff > 712 ExprResult Res = ParseCastE - 713 - 714 - 715 s	xpression(ParseKind, ^ isAddressOfOperand, NotCastExpr, isTypeCast,	Enter your regression command here Regression Diff + -> 729 ExprResult Res = ParseCastE + 730 + 731 + 732 s	Expression(ParseKind, ^ isAddressOfOperand, NotCastExpr, isTypeCast,
	Enter your base command here Base Diff > 712 ExprResult Res = ParseCastE - 713 - 714 - 715 S Process 6783 stopped * thread #1, name = 'clang++', stop reason - frame #0: 0x000055556128f73e clang++` - 1052 - 1053 - 1054 > 1055 ExprResult Res; - 1056 tok::TokenKind SavedKind = - 1057 auto SavedType = Preferred - 1058 NotCastExpr = false;	<pre>xpression(ParseKind,</pre>	Enter your regression command here Regression Diff + -> 729 ExprResult Res = ParseCastE + 730 + 731 + 732 s + Process 6782 stopped * thread #1, name = 'clang++', stop reason + frame #0: 0x0000555561a4f8ec clang++'c + 1068 + 1069 + 1070 + -> 1071 ExprResult Res; ^ + 1072 tok::TokenKind SavedKind = + 1073 auto SavedType = PreferredT + 1074 NotCastExpr = false;	<pre>Expression(ParseKind,</pre>

n

0	Dift	fDebug	
<pre>Base Stackframe frame #0: 0x0055556128f73e clang++`clang: frame #1: 0x0055556128d88e clang++`clang:: frame #2: 0x0055556128ae84 clang++`clang:: frame #3: 0x00555561278c44 clang++`clang:: frame #4: 0x0055556125c8bb clang++`clang:: frame #5: 0x0055556125b4f4 clang++`clang:: frame #6: 0x00555561259ed clang++`clang::</pre>	:Parser::ParseCastExpression(clang::Parser::C :Parser::ParseCastExpression(clang::Parser::C Parser::ParseAssignmentExpression(clang::Pars :Parser::ParseInitializer() at Parser.h:2119 Parser::ParseDeclarationAfterDeclaratorAndAtt Parser::ParseDeclGroup(clang::ParsingDeclSpec Parser::ParseSimpleDeclaration(clang::Declara	<pre>Regression Stackframe + frame #0: 0x00555561a4f8ec clang++`clan + frame #1: 0x00555561a4da94 clang++`clan + frame #2: 0x00555561a4b062 clang++`clan + frame #3: 0x00555561a38f5c clang++`clan + frame #4: 0x00555561a1ca26 clang++`clang + frame #5: 0x00555561a1b64b clang++`clang + frame #6: 0x00555561a1a03e clang++`clang</pre>	ng::Parser::ParseCastExpression(clang::Parser::Ca ng::Parser::ParseCastExpression(clang::Parser::Ca g::Parser::ParseAssignmentExpression(clang::Parse ng::Parser::ParseInitializer() at Parser.h:2125 ( g::Parser::ParseDeclarationAfterDeclaratorAndAttr g::Parser::ParseDeclGroup(clang::ParsingDeclSpec& g::Parser::ParseSimpleDeclaration(clang::Declarat
Base Locals (clang::ExprResult)Res=None (clang::tok::TokenKind)SavedKind=unknown (clang::PreferredTypeBuilder)SavedType=Non (bool)AllowSuffix=false	<ul> <li>Base Args</li> <li>(clang::Parser *)this=0x0000555564748600 (clang::Parser::CastParseKind)ParseKind=An (bool)isAddressOfOperand=false</li> <li>(bool &amp;)NotCastExpr=0x00007fffffff72b7 (clang::Parser::TypeCastState)isTypeCast=N (bool)isVectorLiteral=false (bool *)NotPrimaryExpression=0x0000000000</li> </ul>	Regression Locals (clang::ExprResult)Res=None (clang::tok::TokenKind)SavedKind=unknown (clang::PreferredTypeBuilder)SavedType=Non (bool)AllowSuffix=false	<pre>Regression Args + (clang::Parser *)this=0x00005555648901a0  (clang::Parser::CastParseKind)ParseKind=Any  (bool)isAddressOfOperand=false + (bool &amp;)NotCastExpr=0x00007fffffff03f7  (clang::Parser::TypeCastState)isTypeCast=No  (bool)isVectorLiteral=false  (bool *)NotPrimaryExpression=0x00000000000</pre>
Enter your base command here		Enter your regression command here	
<pre>-&gt; 712 ExprResult Res = ParseCastEx - 713 - 714 - 715 s - Process 6783 stopped * thread #1, name = 'clang++', stop reason = - frame #0: 0x000055556128f73e clang++`cl - 1052 - 1053 - 1054&gt; 1055 ExprResult Res; ^</pre>	pression(ParseKind, isAddressOfOperand, NotCastExpr, isTypeCast, breakpoint 1.1 ang::Parser::ParseCastExpressi TypeCastState isTypeC bool isVectorLiteral, bool *NotPrimaryExpre	<pre>http://www.comments.comme</pre>	<pre>tExpression(ParseKind,</pre>
<ul> <li>10<u>56</u> tok::TokenKind SavedKind = T</li> <li>1057 auto SavedType = PreferredTy</li> <li>10<u>58</u> NotCastExpr = false;</li> </ul>	ok.getKind(); pe;	+ 10 <u>72</u> tok::TokenKind SavedKind = + 1073 auto SavedType = Preferred + 10 <u>74</u> NotCastExpr = false;	= Tok.getKind(); dType;

#### Advantages

Differential Debugging is not restricted to finding regressions in the codebase.

- Bug Localization in Regression Analysis
- Migration and Third–Party Library Updates
- Debugging Across Compiler Optimizations

	llvm / <b>llvm-pro</b> j	ject
<> Code	• Issues 5k+	រា Pull reque



## Time for Demonstration

### Future Work

- Improved semantic diff. E.g. Address Space Randomization (ALSR)
- Automatically halt execution at diverging stack frames
- Watchpoints for diverging variables of interest

These enhancements would reduce manual effort, accelerate bug localization, and improve the overall user experience of IDD.

# Thank You

Any Questions?

- GitHub: github.com/compiler-research/idd
   PyPI: pypi.org/project/idd
- ✤ Install via: pip install idd