



INTERMITTENT MULTI-THREADING BUGS: FIND AND SQUASH RACES, DEADLOCKS, AND MEMORY BUGS

Memory & Thread Debugger

Here is What Will Be Covered

Overview

Memory/Thread analysis

New Features

Deep dive into debugger integrations

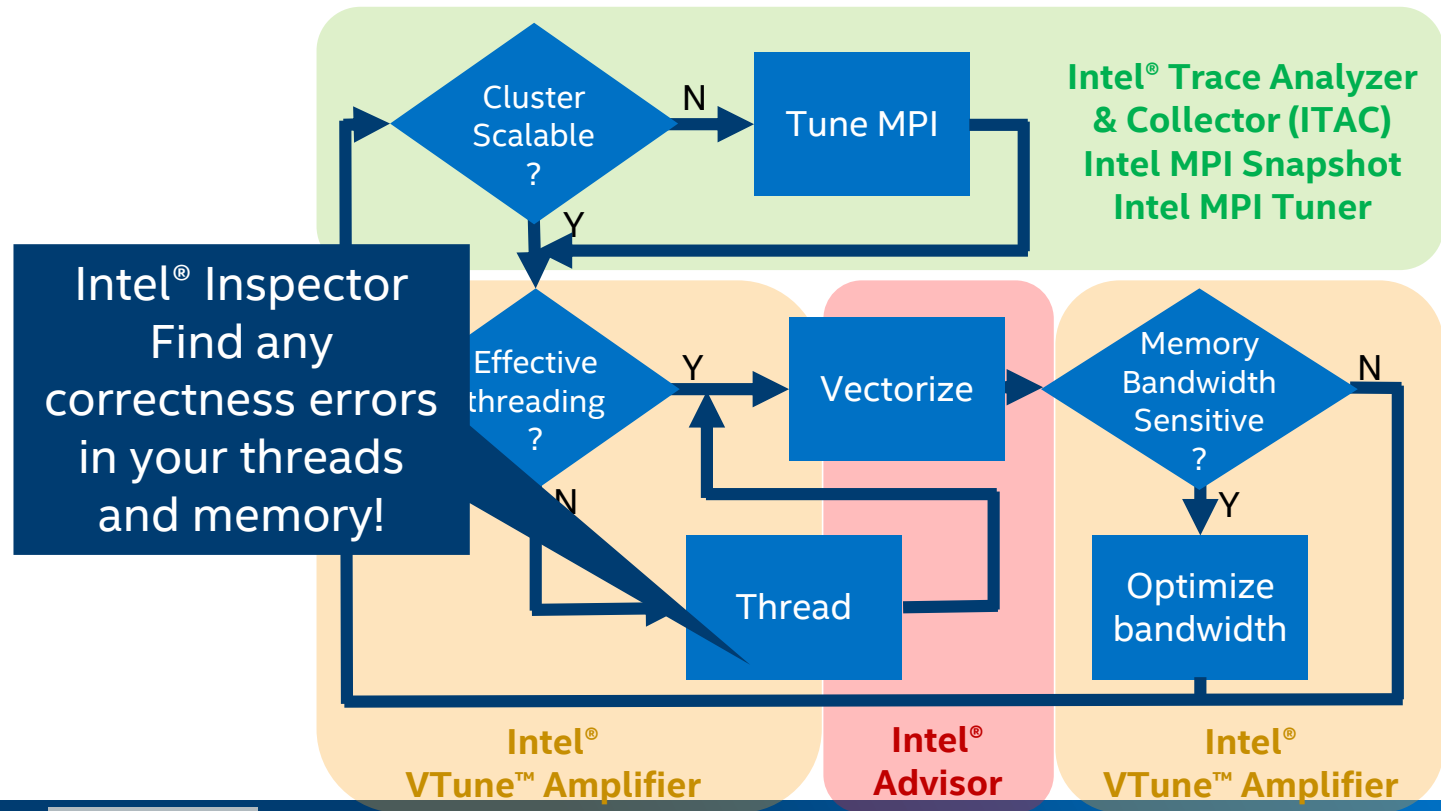
Demo

Call to action

Analysis Tools for Diagnosis

Intel® Parallel Studio XE

Intel® Parallel Studio XE



Optimization Notice

Copyright © 2017, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.



Correctness Tools Increase ROI By 12%-21%

Cost Factors – Square Project Analysis

CERT: U.S. Computer Emergency Readiness Team, and Carnegie Mellon CyLab

NIST: National Institute of Standards & Technology : Square Project Results

Size and complexity of applications is growing



Correctness tools find defects during development prior to shipment

Reworking defects is 40%-50% of total project effort

Reduce time, effort, and cost to repair

Find errors earlier when they are less expensive to fix

Find & Debug Memory & Threading Errors

Intel® Inspector – Memory & Thread Debugger

Correctness Tools Increase ROI By 12%-21%¹

- Errors found earlier are less expensive to fix
- Several studies, ROI% varies, but earlier is cheaper

Diagnosing Some Errors Can Take Months

- Races & deadlocks not easily reproduced
- Memory errors can be hard to find without a tool

Debugger Integration Speeds Diagnosis

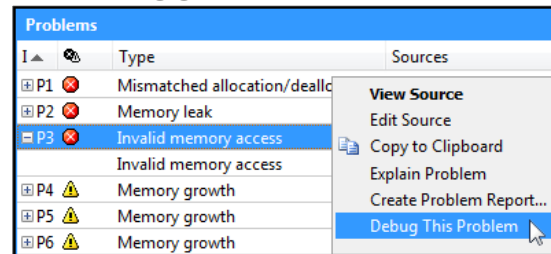
- Breakpoint set just before the problem
- Examine variables & threads with the debugger

Diagnose in hours instead of months

¹ Cost Factors – Square Project Analysis

CERT: U.S. Computer Emergency Readiness Team, and Carnegie Mellon CyLab
NIST: National Institute of Standards & Technology : Square Project Results

Debugger Breakpoints



Part of Intel® Parallel Studio
For Windows* and Linux*

Intel® Inspector dramatically sped up our ability to track down difficult to isolate threading errors before our packages are released to the field.

*Peter von Kaenel, Director,
Software Development,
Harmonic Inc.*

<http://intel.ly/inspector-xe>

Debug Memory & Threading Errors

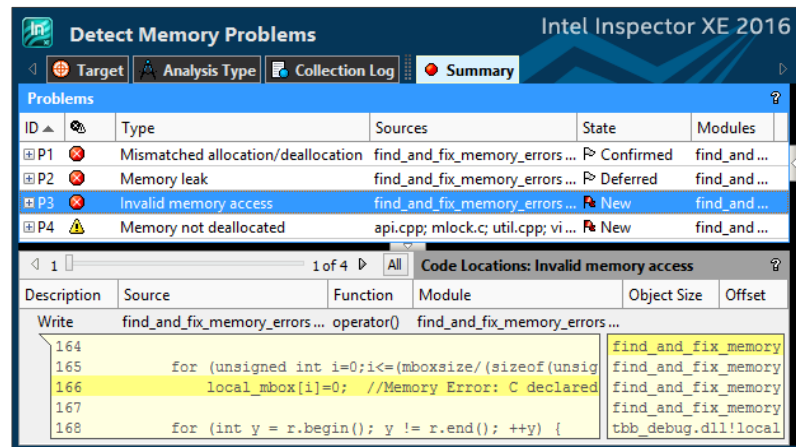
Intel® Inspector

Find and eliminate errors

- Memory leaks, invalid access...
- Races & deadlocks
- C, C++, C#, F# and Fortran (or a mix)

Simple, Reliable, Accurate

- No special recompiles
Use any build, any compiler¹
- Analyzes dynamically generated or linked code
- Inspects 3rd party libraries without source
- Productive user interface + debugger integration
- Command line for automated regression analysis



Clicking an error instantly displays source code snippets and the call stack

Fits your existing process

Optimization Notice ¹That follows common OS standards.

Copyright © 2017, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.

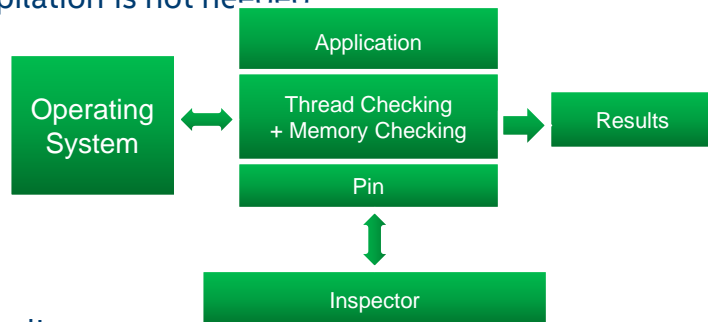


Intel® Inspector dynamic analysis

Data Collection Techniques

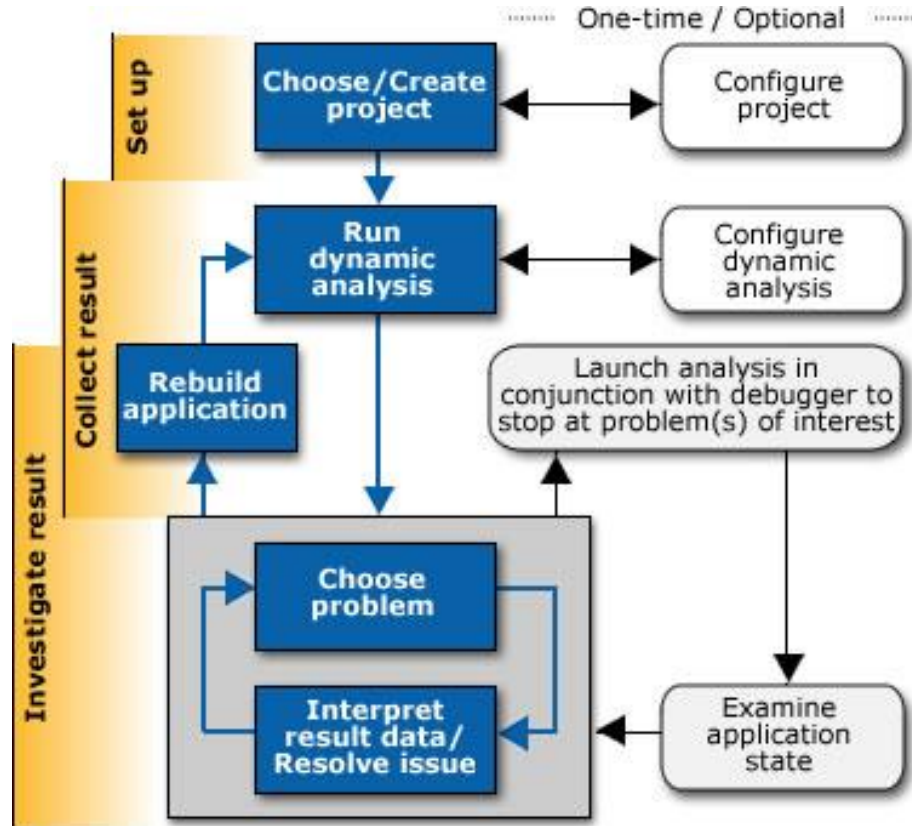
Inspector tracks all memory allocations and threading APIs using a binary instrumentation tool called Pin

- Dynamic instrumentation system provided by Intel (<http://www.pintool.org>)
- Injected code used for observing the behaviour of the program
- Source modification/recompilation is not needed



- OS has to be in the support list
- One process is analysed at a time

Recommended Methodology



Race Conditions Are Difficult to Diagnose

They only occur occasionally and are difficult to reproduce

Correct

Thread 1	Thread 2		Shared Counter
			0
Read count		←	0
Increment			0
Write count		→	1
	Read count	←	1
	Increment		1
	Write count	→	2

Incorrect

Thread 1	Thread 2		Shared Counter
			0
Read count		←	0
	Read count	←	0
Increment			0
	Increment		0
Write count		→	1
	Write count	→	1

Productive User Interface Saves Time

Intel® Inspector

Select a problem set

Code snippets displayed for selected problem

The screenshot displays the Intel Inspector XE 2016 interface. At the top, the title bar reads "Detect Memory Problems" and "Intel Inspector XE 2016". Below the title bar are tabs for "Target", "Analysis Type", "Collection Log", and "Summary". The main area is divided into two panes. The left pane, titled "Problems", contains a table of detected issues. The right pane, titled "Filters", allows for filtering problems by severity, type, and source. Below these panes is a "Code Locations" pane showing the source code for a selected problem.

Problem ID	Type	Sources	State	M
P1	Mismatched allocation/deallocation	find_and_fix_memory_...	Confirmed	
P2	Memory leak	find_and_fix_memory_...	Deferred	fi.
P3	Invalid memory access	find_and_fix_memory_...	New	fi.
P4	Memory not deallocated	api.cpp; mlock.c; util.c...	New	fi.
	Memory not deallocated	video.cpp:82	New	fi.
	Memory not deallocated	util.cpp:163	New	fi.
	Memory not deallocated	api.cpp:218	New	fi.
	Memory not deallocated	mlock.c:347	New	tb.

Description	Source	Function	Module	Object Size	Offset
Mismatched deallocation site	find_and_fix_memory_errors.cpp:175	operator()	find_and_fix_memory_errors.exe		
173	drawing->put_pixel(c);				
174	}				
175	free(drawing); //Memory Error: use delete instead of free				
176	//delete drawing;				
177	}				
Allocation site	find_and_fix_memory_errors.cpp:170	operator()	find_and_fix_memory_errors.exe		
168	for (int y = r.begin(); y != r.end(); ++y) {				
169	{				
170	drawing_area * drawing = new drawing_area(startx, totaly-y, st				
171	for (int x = startx; x < stopx; x++) {				
172	color_t c = render_one_pixel(x, y, local_mbox, serial, st				

Filters let you focus on a module, or error type, or just the new errors or...

Problem States: New, Not Fixed, Fixed, Confirmed, Deferred, Regression

Double Click for Source & Call Stack

Intel® Inspector

Source code locations displayed for selected problem

The screenshot displays the Intel Inspector XE 2016 interface. The main window title is "Mismatched allocation/deallocation". The top navigation bar includes "Target", "Analysis Type", "Collection Log", "Summary", and "Sources". The main content area is divided into two sections: "Mismatched deallocation site" and "Allocation site".

The "Mismatched deallocation site" section shows the following code in `find_and_fix_memory_errors.cpp`:

```
165     for (unsigned int i=0;i<=(mboxsize/(sizeof(unsigned int)));i++)
166         local_mbox[i]=0; //Memory Error: C declared arrays go from 0
167
168     for (int y = r.begin(); y != r.end(); ++y) {
169     {
170         drawing_area * drawing = new drawing_area(startx, totaly
171         for (int x = startx ; x < stopx; x++) {
172             color_t c = render_one_pixel (x, y, local_mbox, serie
173             drawing->put_pixel(c);
174         }
175     }
176     free(drawing); //Memory Error: use delete instead of free
177     //delete drawing;
```

The "Allocation site" section shows the following code in `find_and_fix_memory_errors.cpp`:

```
170     drawing_area * drawing = new drawing_area(startx, totaly
171     for (int x = startx ; x < stopx; x++) {
172         color_t c = render_one_pixel (x, y, local_mbox, serie
173         drawing->put_pixel(c);
174     }
175     free(drawing); //Memory Error: use delete instead of free
176     //delete drawing;
```

Both sections include a "Call Stack" pane on the right. The call stack for the deallocation site includes:

- find_and_fix_memory_errors.exe!operator() - fi
- find_and_fix_memory_errors.exe!run_body - pa
- find_and_fix_memory_errors.exe!execute<class
- find_and_fix_memory_errors.exe!execute<para
- tbb_debug.dll!local_wait_for_all - custom_sc
- tbb_debug.dll!local_spawn_root_and_wait - sc
- tbb_debug.dll!spawn_root_and_wait - schedu
- find_and_fix_memory_errors.exe!spawn_root_a
- find_and_fix_memory_errors.exe!run - parallel_

The call stack for the allocation site includes:

- find_and_fix_memory_errors.exe!operator() - fi
- find_and_fix_memory_errors.exe!run_body - pa
- find_and_fix_memory_errors.exe!execute<class
- find_and_fix_memory_errors.exe!execute<para
- tbb_debug.dll!local_wait_for_all - custom_sche
- tbb_debug.dll!local_spawn_root_and_wait - sc
- tbb_debug.dll!spawn root and wait - schedu

Call Stack

Quickly track down your Fortran issues!

Problems					
ID	Type	Sources	Modules	Object Size	State
P1	Memory leak	nqueens_memory.f90	memory_issues.exe	64	New

Code Locations: Memory leak					
Description	Source	Function	Module	Object Size	Variable
Allocation site nqueens_memory.f90:50 NQUEENS memory_issues.exe 64					
48	!\$ nthreads = omp_get_max_threads()			memory_issues.exe!NQUEENS - nqueens	
49				memory_issues.exe!main	
50	allocate(correct_solution(16))			memory_issues.exe!_tmainCRTStartup	
51	correct_solution = (/ 1,0,0,1,2,10,4,40,92,352,72				
52					

Code Locations: Data race					
Description	Source	Function	Module	Variable	
Read nqueens_threading.f90:117 NQUEENS_ip_SETQUEEN threading_issues.exe					
115	! Recursive routine to set a queen on the board			threading_issues.exe!NQUEENS_ip_SET	
116				threading_issues.exe!NQUEENS_ip_SET	
117	recursive subroutine setQueen (queens, row, col)			threading_issues.exe!NQUEENS_ip_SET	
118	implicit none			threading_issues.exe!NQUEENS_ip_SET	
119	integer, intent(inout) :: queens(:)			threading_issues.exe!NQUEENS_ip_SET	
Write nqueens_threading.f90:117 NQUEENS_ip_SETQUEEN threading_issues.exe					
115	! Recursive routine to set a queen on the board			threading_issues.exe!NQUEENS_ip_SET	
116				threading_issues.exe!NQUEENS_ip_SET	
117	recursive subroutine setQueen (queens, row, col)			threading_issues.exe!NQUEENS_ip_SET	
118	implicit none			threading_issues.exe!NQUEENS_ip_SET	
119	integer, intent(inout) :: queens(:)			threading_issues.exe!NQUEENS_ip_SET	

Locate Deadlocks and Data Races

Target Analysis Type Collection Log Summary

Problems				
ID	Type	Sources	Modules	State
P1	Data race	nqueens_threading.f90	threading_issues.exe	New
P2	Data race	nqueens_threading.f90	threading_issues.exe	New
P3	Data race	nqueens_threading.f90	threading_issues.exe	New

Filtering - Focus on What's Important

Example: See only the errors in one source file

Before – All Errors

(1) Filter – Show only one source file

ID	Type	Sources	State
P1	Mismatched alloc...	find_an ...	New
P2	Mismatched alloc...	api.cpp	New
P3	Memory leak	api.cpp	Confirmed
P4	Mismatched alloc...	video.c ...	Not fixed
P5	Mismatched alloc...	video.c ...	Not fixed
P6			

Severity	Count
Error	55 item(s)
Warning	1 item(s)

Type	Count
Invalid memory access	41 item(s)
Memory leak	1 item(s)
Memory not deallocated	11 item(s)
Mismatched allocation/dealloc...	2 item(s)

Source	Count
api.cpp	21 item(s)
find_and_fix_memory_errors.cpp	3 item(s)
util.cpp	10 item(s)
video.cpp	21 item(s)

After – Only errors from one source file

(2) Error count drops

ID	Type	Sources	State
P1	Mismatche...	find_an ...	New
P2	Memory leak	find_an ...	Confirmed
P3	Invalid me...	find_an ...	Deferred

Severity	Count
Error	3 item(s)

Type	Count
Invalid memory access	1 item(s)
Memory leak	1 item(s)
Mismatched allocation/dealloc...	1 item(s)

Source	Count
find_and_fix_memory_errors.cpp	3 item(s)

State	Count
Confirmed	1 item(s)
Deferred	1 item(s)
New	1 item(s)

Tip: Set the “Investigated” filter to “Not investigated” while investigating problems. This removes from view the problems you are done with, leaving only the ones left to investigate.

Incrementally Diagnose Memory Growth

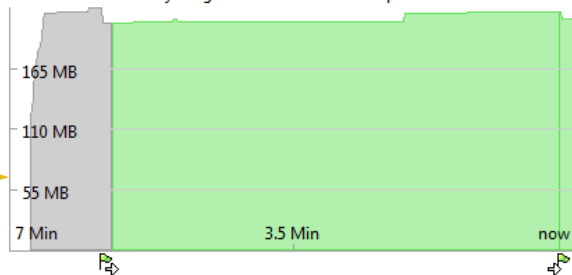
Intel® Inspector

As your app is running...

Memory usage graph plots memory growth

Memory Used by Analysis Tool and Target Application

Last recorded memory usage before collection completed: 211 MB



Select a cause of memory growth

ID	Type	Sources	Modules	Object Size	State
	Memory growth	gdiplus.dll:0x47240	gdiplus.dll	40960	New
	Memory growth	find_and_fix_memory_errors.cpp:163	find_and_fix_memory_errors.exe	90108	Not fixed
	Memory growth	find_and_fix_memory_errors.cpp:163	find_and_fix_memory_errors.exe	1802160	Not fixed
	Memory growth	find_and_fix_memory_errors.cpp:163	find_and_fix_memory_errors.exe	30036	Not fixed
	Memory growth	find_and_fix_memory_errors.cpp:163	find_and_fix_memory_errors.exe	1621944	Not fixed
	Memory growth	find_and_fix_memory_error.cpp:170	find_and_fix_memory_error.exe	40	Not fixed

Description	Source	Function	Module	Object Size	Offset
Allocation site	find_and_fix_memory_errors.cpp:163	operator()	find_and_fix_memory_errors.exe	90108	
161	unsigned int serial=1;				find_and_fix_memory_errors.exe
162	unsigned int mboxsize = sizeof(unsigned int)*(max_objectid() +				find_and_fix_memory_errors.exe
163	unsigned int * local_mbox = (unsigned int *) malloc(mboxsize);				find_and_fix_memory_errors.exe
164					find_and_fix_memory_errors.exe
165	for (unsigned int i=0;i<=(mboxsize/(sizeof(unsigned int)));i++				tbb_debug.dll!local_wait_for_e

See the code snippet & call stack

Speed diagnosis of difficult to find heap errors

Optimization Notice

Copyright © 2017, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.



Automate Regression Analysis

Command Line Interface

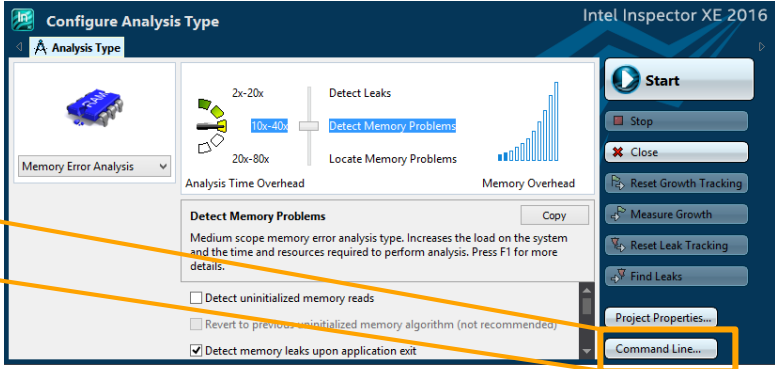
inspxe-cl is the command line:

- **windows:** C:\Program Files\Intel\Inspector XE \bin[32|64]\inspxe-cl.exe
- **Linux:** /opt/intel/inspector_xe/bin[32|64]/inspxe-cl

Help:

```
inspxe-cl -help
```

Set up command line with GUI



Command Line...

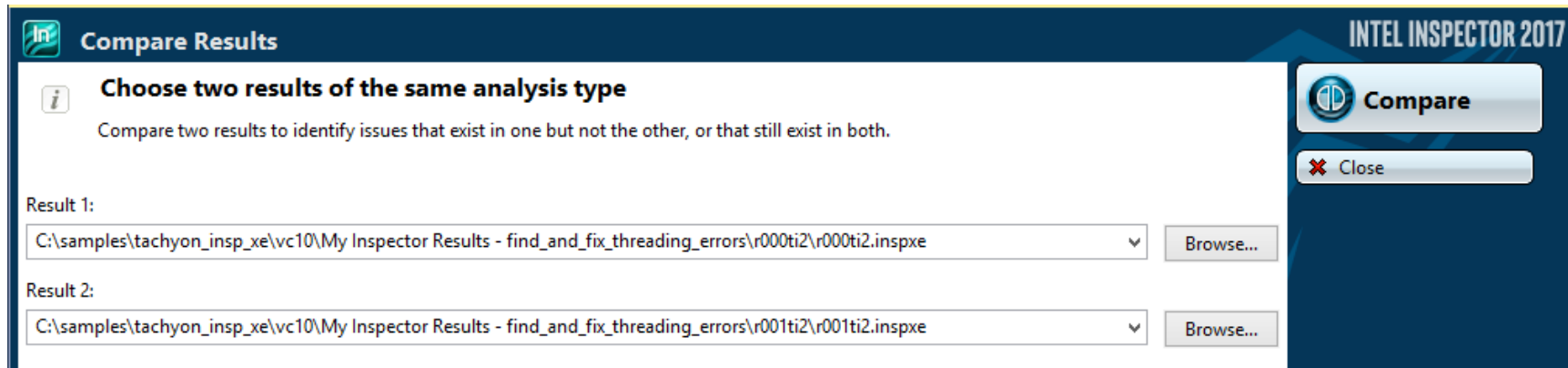
Command examples:

1. `inspxe-cl -collect-list`
2. `inspxe-cl -collect ti2 -- MyApp.exe`
3. `inspxe-cl -report problems`

Send results file to developer to analyze with the UI

Compare results and see what has changed

Ideal for regression testing



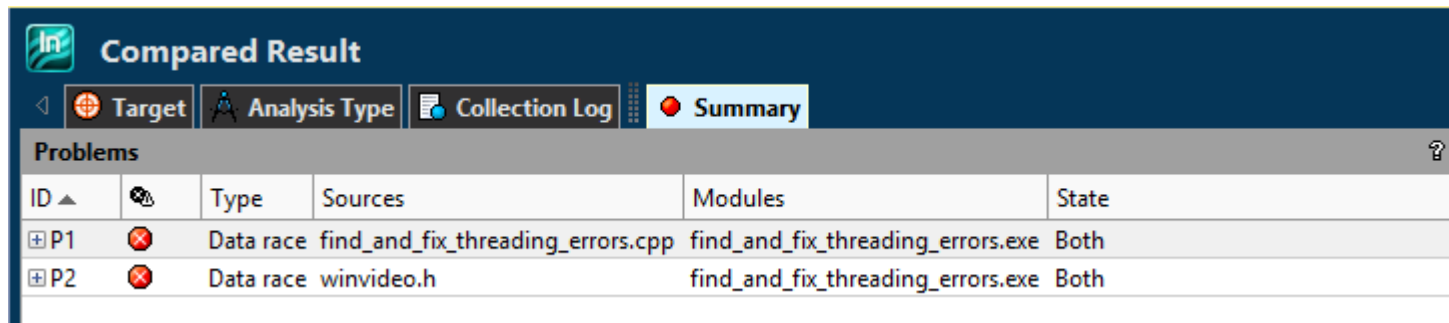
Compare Results

Choose two results of the same analysis type
Compare two results to identify issues that exist in one but not the other, or that still exist in both.

Result 1:
C:\samples\tachyon_insp_xe\vc10\My Inspector Results - find_and_fix_threading_errors\r000ti2\r000ti2.inspxe Browse...

Result 2:
C:\samples\tachyon_insp_xe\vc10\My Inspector Results - find_and_fix_threading_errors\r001ti2\r001ti2.inspxe Browse...

Compare
Close



Compared Result

Target Analysis Type Collection Log Summary

Problems

ID	Type	Sources	Modules	State
P1	Data race	find_and_fix_threading_errors.cpp	find_and_fix_threading_errors.exe	Both
P2	Data race	winvideo.h	find_and_fix_threading_errors.exe	Both

Optimization Notice

Copyright © 2017, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Find problems quicker!

Interactive debugging support

Configure Analysis Type

INTEL INSPECTOR 2017

Analysis Type

Threading Error Analysis

10x-40x Detect Deadlocks

20x-80x Detect Deadlocks and Data Races

40x-160x Locate Deadlocks and Data Races

Analysis Time Overhead Memory Overhead

Detect Deadlocks and Data Races Copy

Medium scope threading error analysis type. Increases the load on the system and the time and resources required to perform analysis. Press F1 for more details.

Terminate on deadlock

Stack frame depth: 1

Analyze without debugger
Run an analysis and report all detected problems. Use to view correctness issues without stopping in the debugger

Enable debugger when problem detected
Run an analysis under the debugger and stop every time a problem is detected. Use to allow investigation of every

Select analysis start location with debugger
Run target application under the debugger with analysis disabled until you choose to turn on analysis. Before starti

Details

Start

Stop

Close

Reset Growth Tracking

Measure Growth

Reset Leak Tracking

Find Leaks

3 debugging modes supported

1. Analyze without debugger
2. Enable debugger when problem detected
3. Start analysis when a debug breakpoint is hit.

Optimization Notice

Copyright © 2017, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.



Intuitive problem solving using debugger integrations

Microsoft Visual Studio* and GNU gdb* or Intel® Debugger (on Linux*)

```
    ///  
    Refresh screen picture  
    bool video::next_frame()  
    {  
        if(!running) return false;  
        g_updates++; // Fast but inaccurate counter. The data race here is benign  
        if(!threaded) while(loop_once(this));  
        else if(g_handles[1]) {  
            SetEvent(g_handles[1]);  
            YIELD_TO_THREAD();  
        }  
        return true;  
    }  
}
```

Problem Details

Source Intel Inspector Disable Breakpoint Re-enable Breakpoints

Data race at data location 0x135dc for threads 16208 and TBB Worker Thread

Description ▲	Source	Function	Module
⊕ Read	winvideo.h:270	next_frame	find_and_fix_threading_errors.exe
⊕ Write	winvideo.h:271	next_frame	find_and_fix_threading_errors.exe

Problem Deta... Compiler Inli... Compiler Opt... Call Stack Breakpoints Output

Optimization Notice

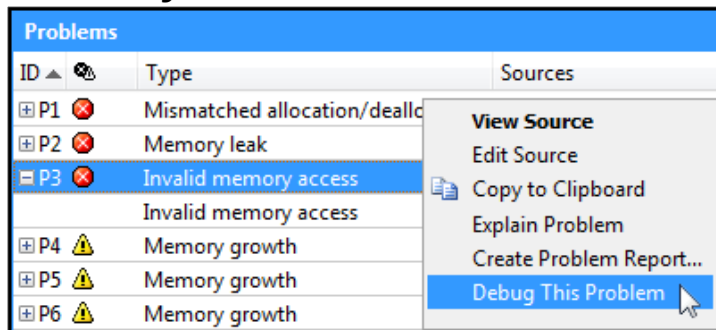
Copyright © 2017, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.



Break At Just The Right Time

Intel® Inspector - Memory & Thread Debugger

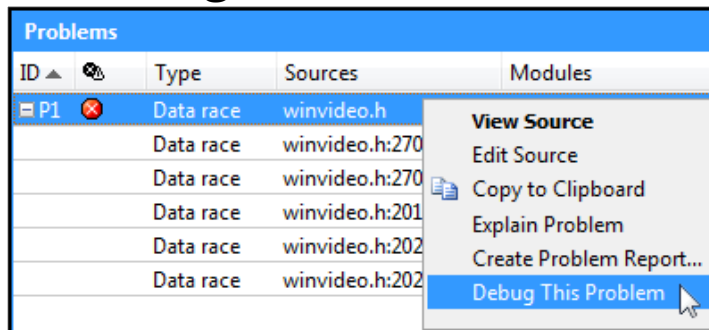
Memory Errors



Problems		
ID	Type	Sources
P1	Mismatched allocation/deallocation	
P2	Memory leak	
P3	Invalid memory access	
P4	Memory growth	
P5	Memory growth	
P6	Memory growth	

- View Source
- Edit Source
- Copy to Clipboard
- Explain Problem
- Create Problem Report...
- Debug This Problem

Threading Errors



Problems			
ID	Type	Sources	Modules
P1	Data race	winvideo.h	
	Data race	winvideo.h:270	
	Data race	winvideo.h:270	
	Data race	winvideo.h:201	
	Data race	winvideo.h:202	
	Data race	winvideo.h:202	

- View Source
- Edit Source
- Copy to Clipboard
- Explain Problem
- Create Problem Report...
- Debug This Problem

Break into the debugger just before the error occurs.

Examine the variables and threads.

Diagnose the problem.

Save time. Find and diagnose errors with less effort.

Work Smarter & Faster

Intel® Inspector - Memory & Thread Debugger

Precise Error Suppression

```
Suppression = {  
    Name = "Example";  
    Type = { uninitialized_memory_access }  
    Stacks = {  
        {  
            mod=a.out, func=update_x;  
            func=main;  
        }  
    }  
}
```

Precise, easy to edit, team shareable.

Choose which stack frame to suppress.

Eliminate the false, not the real errors.

Pause/Resume Collection

```
__itt_suppress_push(__itt_suppress_threading_errors);  
/* Any threading errors here are ignored */  
__itt_suppress_pop();  
/* Any threading errors here are seen */
```

Speed-up analysis by limiting its scope.

Analyze only during the execution of the suspected problem.

Find and diagnose errors with less effort.

Productive Memory & Threading Debugger

Intel® Inspector

	Memory Analysis	Threading Analysis
View Context of Problem		
Stack	✓	✓
Multiple Contributing Source Locations	✓	✓
Collapse multiple “sightings” to one error (e.g., memory allocated in a loop, then leaked is 1 error)	✓	✓
Suppression, Filtering, and Workflow Management	✓	✓
Visual Studio* Integration (Windows*)	✓	✓
Command line for automated tests	✓	✓
Time Line visualization	✓	✓
Memory Growth during a transaction	✓	
Trigger Debugger Breakpoint	✓	✓

Easier & Faster Debugging of Memory & Threading Errors

Optimization Notice

Copyright © 2017, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.



WHAT'S NEW

New Features

Support for Intel® Xeon Phi™ processor (codename: Knights Landing)

Support for C++11 synchronization primitives during threading analysis

Variable name detection for threading analysis

Variable name detection for threading analysis

Detect Deadlocks and Data Races INTEL INSPECTOR 2017

Target Analysis Type Collection Log Summary

Problems

ID	Type	Sources	Modules	State
P1	Data race	find_and_fix_threading_errors.cpp	find_and_fix_threading_errors.exe	New
	Data race	find_and_fix_threading_errors.cp ...	find_and_fix_threading_errors.exe	New
	Data race	find_and_fix_threading_errors.cp ...	find_and_fix_threading_errors.exe	New
	Data race	find_and_fix_threading_errors.cp ...	find_and_fix_threading_errors.exe	New
	Data race	find_and_fix_threading_errors.cp ...	find_and_fix_threading_errors.exe	New
P2	Data race	winvideo.h	find_and_fix_threading_errors.exe	New

Filters Sort

Severity	Count
Error	2 item(s)

Type	Count
Data race	2 item(s)

Source	Count
find_and_fix_threading_errors.cpp	1 item(s)
winvideo.h	1 item(s)

Module	Count
find_and_fix_threading_errors.exe	2 item(s)

State	Count
New	2 item(s)

Suppressed	Count
Not suppressed	2 item(s)

Investigated	Count
Not investigated	2 item(s)

Code Locations: Data race

Description	Source	Function	Module	Variable
Write	find_and_fix_threading_errors.cpp:105	render_one_pixel	find_and_fix_threading_errors.exe	col:r
	103	primary.scene = sscene;		find_and_fix_threading_errors.exe!render_one_pix
	104			
	105	col=trace(sprimary); //Threading Error: col is a global variable		
	106	//2 ways to fix this threading error		
	107	// 1) Make col a local variable		
Write	find_and_fix_threading_errors.cpp:105	render_one_pixel	find_and_fix_threading_errors.exe	col:r
	103	primary.scene = sscene;		find_and_fix_threading_errors.exe!render_one_pix
	104			find_and_fix_threading_errors.exe!render_one_pix
	105	col=trace(sprimary); //Threading Error: col is a global variable		
	106	//2 ways to fix this threading error		
	107	// 1) Make col a local variable		

Timeline

- read_video (14232)
- BB Worker Thread (12516)

Optimization Notice

Memory & Threading Debugger Saves Time

Intel® Inspector

“We struggled for a week with a crash situation, the corruption was identified but the source was really hard to find. Then we ran **Intel® Inspector** and immediately found the array out of bounds that occurred long before the actual crash. We could have saved a week!”

*Mikael Le Guerroué,
Senior Codec Architecture Engineer,
Envivio*

“**Intel® Inspector** is quite fast and intuitive compared to products we have used in the past. We can now run our entire batch of test cases (~750) which was not feasible previously. **Intel® Inspector** easily completed tests that failed due to lack of virtual memory on another product.”

*Gerald Mattauch
Senior Software Developer
Siemens AG, Healthcare Sector*

Intel® Inspector has dramatically sped up our ability to find/fix memory problems and track down difficult to isolate threading errors before our packages are released to the field.

*Peter von Kaenel, Director,
Software Development,
Harmonic Inc.*

[More Case Studies](#)

Call to Action

Modernize your Code

- To get the most out of your hardware, you need to modernize your code with vectorization and threading.
- Taking a methodical approach such as the one outlined in this presentation, and taking advantage of the powerful tools in Intel® Parallel Studio XE, can make the modernization task dramatically easier.

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © 2015, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

