

Intel® Parallel Inspector

Product Brief

Intel® Parallel Inspector



"Intel® Parallel Inspector and Intel® Parallel Amplifier greatly simplified the task of finding hotspots and memory leaks. We were pleased with the 2X overall performance improvement and the elimination of several previously unidentified memory leaks."

*Vlad Romashko
Software Development Manager
OpenCascade S.A.S*

Easily Find Threading and Memory Errors before They Happen

Intel® Parallel Inspector is the easiest multithreading error checking tool for Microsoft Visual Studio* C/C++ developers. Intel Parallel Inspector detects challenging threading and memory errors and provides guidance to help ensure application reliability. Unlike other error checkers on the market, Intel Parallel Inspector is the fastest and most comprehensive method to pinpoint latent multithreading and memory errors.

Proactively find latent threading and memory errors to help ensure application reliability with Intel Parallel Inspector.

- Find memory and threading errors with one easy-to-use tool
- Give both experts and novices greater insight into parallel code behavior
- Help ensure that shipped applications run error-free on customer systems
- Find latent bugs within the increasing complexity of parallel programs
- Reduce support costs and increase productivity

Memory and Thread Checking in

One Easy-to-Use Tool

Both memory and thread checking are fully integrated into Microsoft Visual Studio with an easy-to-use interface. Intel Parallel Inspector provides root-cause analysis of crash-causing threading and memory defects. These features, combined with problem set analysis that summarizes related bugs, make this the most comprehensive tool for finding memory and threading errors. Competitive products only support serial applications or do not provide comprehensive memory and thread correctness checking using one tool.

Dynamic Instrumentation That

Works on Standard Builds

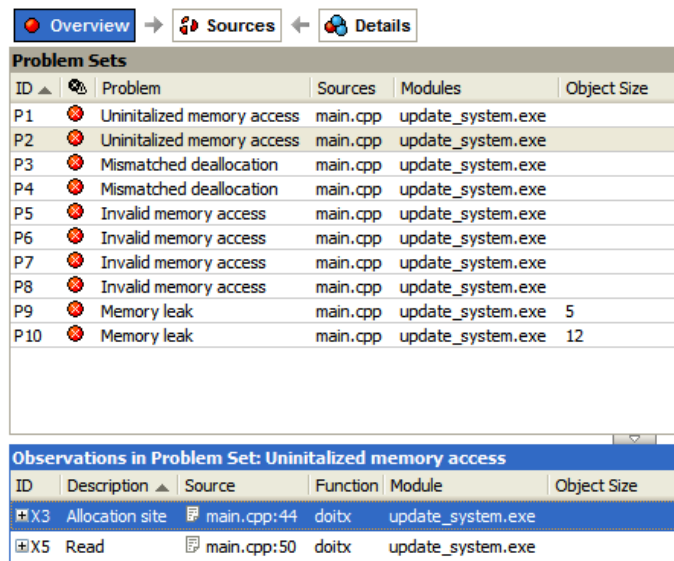
Intel Parallel Inspector doesn't require special builds or compilers, and utilizes dynamic instrumentation to acquire test data. Since it only instruments the code that's executed, analysis can run in less time and work on larger applications.

Thread-Aware Memory Checker

Not all memory checkers are capable of performing analysis of threaded applications. Intel Parallel Inspector performs comprehensive memory checks (e.g., memory leaks, invalid memory read/write, dangling pointer detection, use of uninitialized data) on both single and multithreaded applications.

Excellent Value

Intel Parallel Inspector is aggressively priced for a combined memory and threading correctness tool, so it's an excellent value compared with competitive products. Intel Parallel Inspector is included in Intel® Parallel Studio, which is a comprehensive suite of products for developing, debugging, and tuning parallel C/C++ applications.

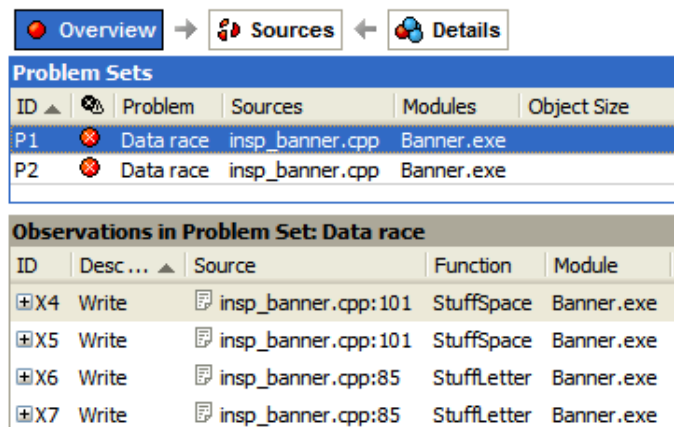


The screenshot shows the Intel Parallel Inspector interface with the 'Overview' tab selected. It displays a table of 'Problem Sets' with columns for ID, Problem, Sources, Modules, and Object Size. Below the table, there is a section for 'Observations in Problem Set: Uninitialized memory access' showing specific instances of the problem.

ID	Problem	Sources	Modules	Object Size
P1	Uninitialized memory access	main.cpp	update_system.exe	
P2	Uninitialized memory access	main.cpp	update_system.exe	
P3	Mismatched deallocation	main.cpp	update_system.exe	
P4	Mismatched deallocation	main.cpp	update_system.exe	
P5	Invalid memory access	main.cpp	update_system.exe	
P6	Invalid memory access	main.cpp	update_system.exe	
P7	Invalid memory access	main.cpp	update_system.exe	
P8	Invalid memory access	main.cpp	update_system.exe	
P9	Memory leak	main.cpp	update_system.exe	5
P10	Memory leak	main.cpp	update_system.exe	12

ID	Description	Source	Function	Module	Object Size
X3	Allocation site	main.cpp:44	doitx	update_system.exe	
X5	Read	main.cpp:50	doitx	update_system.exe	

Quickly finds memory errors, including leaks and corruptions, in single and multithreaded applications. This decreases support costs by finding memory errors before an application ships.

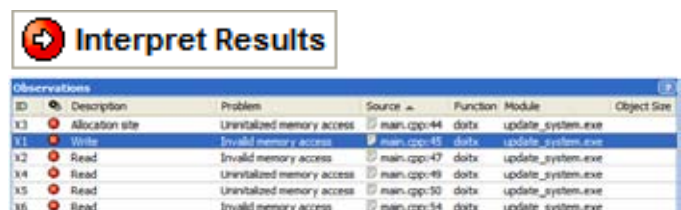


The screenshot shows the Intel Parallel Inspector interface with the 'Overview' tab selected. It displays a table of 'Problem Sets' with columns for ID, Problem, Sources, Modules, and Object Size. Below the table, there is a section for 'Observations in Problem Set: Data race' showing specific instances of the problem.

ID	Problem	Sources	Modules	Object Size
P1	Data race	insp_banner.cpp	Banner.exe	
P2	Data race	insp_banner.cpp	Banner.exe	

ID	Description	Source	Function	Module
X4	Write	insp_banner.cpp:101	StuffSpace	Banner.exe
X5	Write	insp_banner.cpp:101	StuffSpace	Banner.exe
X6	Write	insp_banner.cpp:85	StuffLetter	Banner.exe
X7	Write	insp_banner.cpp:85	StuffLetter	Banner.exe

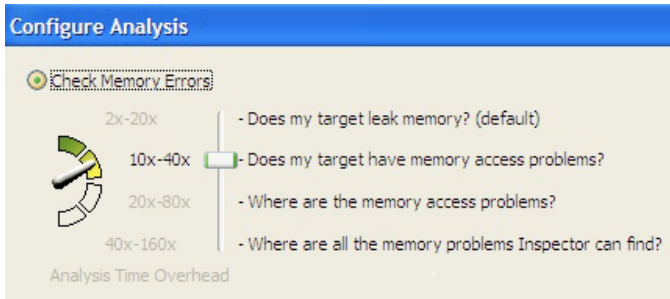
Accurately pinpoints latent threading errors including deadlocks and data races, which helps reduce stalls and crashes due to common errors not found by debuggers and other tools.



The screenshot shows the Intel Parallel Inspector interface with the 'Interpret Results' tab selected. It displays a table of 'Observations' with columns for ID, Description, Problem, Source, Function, Module, and Object Size. The table shows a list of related issues grouped together.

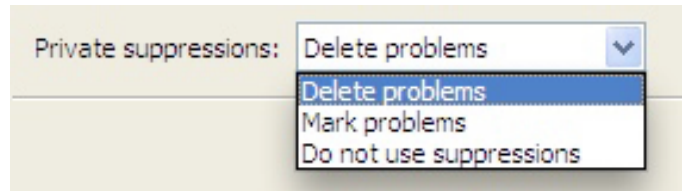
ID	Description	Problem	Source	Function	Module	Object Size
X3	Allocation site	Uninitialized memory access	main.cpp:44	doits	update_system.exe	
X1	Write	Invalid memory access	main.cpp:45	doits	update_system.exe	
X2	Read	Invalid memory access	main.cpp:47	doits	update_system.exe	
X4	Read	Uninitialized memory access	main.cpp:48	doits	update_system.exe	
X5	Read	Uninitialized memory access	main.cpp:50	doits	update_system.exe	
X6	Read	Invalid memory access	main.cpp:54	doits	update_system.exe	

Intuitively guides the developer by grouping related issues together. When you fix one problem, Intel Parallel Inspector shows you all of the related locations where the same fix needs to be applied.



Simple analysis configuration enables developers to control the depth of analysis vs. collection time.

- L1 analysis finds memory leaks and deadlocks.
- L2 analysis identifies the existence of a problem.
- L3 analysis provides root cause information to fix problems.
- L4 provides the most comprehensive level of problem identification and detail.



Result suppression reduces the information that has to be analyzed by suppressing results that are not relevant.

Overview
Sources
Details

Focused observation: main.cpp:50 - Read

```

48
49     c = p[0];
50     c = p[1];
51
52     free(p);
53
54     c = p[2];
55
56     p = (char*)malloc(3);
57

```

Related observation: main.cpp:44 - Allocation site

```

42 void doitx()
43 {
44     char* p = (char*)malloc(4);
45     p[4] = 'a';
46
47     char c = p[5];
48
49     c = p[0];
50     c = p[1];
51

```

Observations in Problem Set: Uninitialized memory access

ID	Description	Source	Function	Module
X3	Allocation site	main.cpp:44	doitx	update_system.exe
X5	Read	main.cpp:50	doitx	update_system.exe

Click on an identified problem to reveal source code to go directly to the offending code to make changes quickly.

Features

- Fully integrated with Microsoft Visual Studio*
- Find memory errors in single and multithreaded applications
 - Memory checking includes uninitialized load detection, use of invalid memory references, mismatched memory allocation and deallocation, memory leaks detection, stack memory checks, and stack trace with controllable stack trace depth
- Find threading errors
 - Data race detection, deadlock detection, depth configurable call stack analysis, diagnostic guidance, built-in knowledge of Intel® Threading Building Blocks, OpenMP*, and Windows threads
- Works with any standard debug build
 - No special test builds or compilers required, so it's easier to test code more often
- Dynamic instrumentation enables testing code without the source; test larger applications because less memory is needed since only executed code is instrumented

System Requirements

- Microsoft Visual Studio
- For the latest system requirements, go to:
www.intel.com/software/products/systemrequirements/

Compatibility

- Compilers: Microsoft Visual C++* Compiler 2005 and 2008 and Intel C++ Compiler
- Threading methodologies: Intel Threading Building Blocks, OpenMP, Windows Threads
- Processors: Designed for and tested on Intel® IA-32 and Intel® 64 processors including Intel® Core™ 2 and Core™ i7 processors. It can be used on compatible processors, although proprietary instructions may cause it to function incorrectly. Please note that Intel® Parallel Composer (compiler and libraries) support Intel IA-32, Intel 64, and all compatible processors.

Support

Intel Parallel Studio products include access to community forums and a knowledge base for all your technical support needs, including technical notes, application notes, documentation, and all product updates.

For more information, go to
<http://software.intel.com/sites/support/>

Beta Versions Available Now

Download and register for the user forums at:
www.intel.com/software/ParallelStudioBeta/

Intel® Parallel Studio

Designed for today's serial applications and tomorrow's software innovators.

Intel brings simplified parallelism to Microsoft Visual Studio* C++ developers with a complete productivity solution designed to optimize serial and new parallel applications for multicore and scale for manycore.

Intel® Parallel Studio: Create optimized serial and parallel applications with the ultimate all-in-one parallelism toolkit

Intel® Parallel Composer: Develop effective applications with a C/C++ compiler and advanced threaded libraries

Intel® Parallel Inspector: Ensure application reliability with proactive parallel memory and threading error checking

Intel® Parallel Amplifier: Quickly find bottlenecks and tune parallel applications for scalable multicore performance

