

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.*

Increase Reliability by Finding Threading and Memory Errors Before They Happen

Intel® Parallel Inspector combines threading and memory checking into one powerful error checking tool. It helps increase the reliability, security, and accuracy of C/C++ applications from within Microsoft Visual Studio*. Intel Parallel Inspector uses dynamic instrumentation that requires no special test builds or compilers, so it's easier to test code more often.

- Find memory and threading errors with one easy-to-use tool
- Help ensure that shipped applications run error-free on customer systems
- Give both experts and novices greater insight into parallel code behavior
- 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 a 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 and Binaries

Intel Parallel Inspector utilizes dynamic instrumentation to acquire test data and doesn't require special builds, add-ins, or compilers. Since it only instruments the code that's executed, analysis can run in less time and work on larger applications. It can even find errors in binaries where you don't have the source code.

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.

| ID | Problem | Sources | Modules | Object Size | State |
|-----|------------------------------------|----------|-------------------|-------------|-----------|
| P1 | Uninitialized memory access | main.cpp | worstcodeever.exe | | Not fixed |
| P2 | Uninitialized memory access | main.cpp | worstcodeever.exe | | Not fixed |
| P3 | Mismatched allocation/deallocation | main.cpp | worstcodeever.exe | | Not fixed |
| P4 | Mismatched allocation/deallocation | main.cpp | worstcodeever.exe | | Not fixed |
| P5 | Invalid memory access | main.cpp | worstcodeever.exe | | Fixed |
| P6 | Invalid memory access | main.cpp | worstcodeever.exe | | Not fixed |
| P7 | Invalid memory access | main.cpp | worstcodeever.exe | | Not fixed |
| P8 | Invalid memory access | main.cpp | worstcodeever.exe | | Not fixed |
| P9 | Memory leak | main.cpp | worstcodeever.exe | 5 | Not fixed |
| P10 | Memory leak | main.cpp | worstcodeever.exe | 12 | Not fixed |

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.

| ID | Problem | Sources | Modules | Object Size | State |
|----|--------------------------|----------|-------------------|-------------|-----------|
| P1 | Data race | main.cpp | worstcodeever.exe | | Not fixed |
| P2 | Lock hierarchy violation | main.cpp | worstcodeever.exe | | Not fixed |

| ID | Description | Source | Function | Module | Object Size | State |
|-----|-----------------|--------------|----------|-------------------|-------------|-------------|
| X10 | Allocation site | main.cpp:176 | DeadLock | worstcodeever.exe | | Information |
| X13 | Allocation site | main.cpp:174 | DeadLock | worstcodeever.exe | | Information |

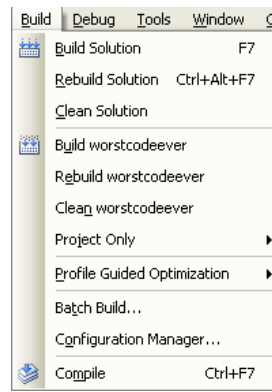
Accurately pinpoints latent threading errors including deadlocks and data races. This helps reduce stalls and crashes due to threading errors not found by debuggers and other tools.

Analysis completed successfully **Interpret Result**

Event Log → Sources

| Time | Description | Modules | Sources |
|----------|---------------------------------------|-------------------|--------------------------|
| 11:06:05 | Error:Invalid memory access | worstcodeever.exe | main.cpp:52 |
| 11:06:06 | Error:Invalid memory access | worstcodeever.exe | main.cpp:54 |
| 11:06:06 | Error:Uninitialized memory access | worstcodeever.exe | main.cpp:51; main.cpp:56 |
| 11:06:06 | Error:Uninitialized memory access | worstcodeever.exe | main.cpp:51; main.cpp:57 |
| 11:06:06 | Error:Invalid memory access | worstcodeever.exe | main.cpp:61 |
| 11:06:06 | Error:Mismatched allocation/deallo... | worstcodeever.exe | main.cpp:64; main.cpp:66 |
| 11:06:06 | Error:Mismatched allocation/deallo... | worstcodeever.exe | main.cpp:63; main.cpp:67 |
| 11:06:06 | Error:Invalid memory access | worstcodeever.exe | main.cpp:79 |
| 11:06:06 | Error:Memory leak | worstcodeever.exe | main.cpp:76 |
| 11:06:06 | Error:Memory leak | worstcodeever.exe | main.cpp:192 |

Clicking the **Interpret Result** button intuitively guides the developer by grouping related issues together. When you fix one problem, Parallel Inspector shows you all of the related locations where the same fix needs to be applied.



Parallel Inspector works on standard debug builds and even binaries. No special compilers, add-ins or special builds needed.

Configure Analysis

Intel Parallel Inspector

Analysis type: Memory Errors Always show this dialog box before running memory error analyses

2x-20x | Does my target leak memory?
 10x-40x | Does my target have memory access problems?
 20x-80x | Where are the memory access problems?
 40x-160x | Where are all the memory problems Inspector can find?

Analysis Time Overhead

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.

Overview → Sources ← Details

Focus Observation: main.cpp:86 - Write

```
85 for (int i=0; i<TotalBlueTroops; i++)
86     BantamBridge+=Blue;
87 return NULL;
88 }
```

Related Observation: main.cpp:92 - Write

```
91 for (int i=0; i<TotalGreyTroops; i++)
92     BantamBridge+=Grey;
93 return NULL;
94 }
```

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

Private suppressions:

- Delete problems
- Delete problems
- Mark problems
- Do not use suppressions

Result suppression allows you to mark or delete identified issues that are not relevant.

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* 2005 or 2008 (except the Express Edition)
- 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) supports 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/>

Download a Trial Version Today

Evaluation copy available at:

www.intel.com/software/products/ParallelStudio/

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

