Using Intel® Inspector XE 2011
with Fortran Applications

Jackson Marusarz
Intel Corporation
Legal Disclaimer

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY
ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN
INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS
ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES
RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER
INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION
IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or
characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have
no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to
change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate
from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product
order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by
calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Inside, Cilk, Core Inside, i960, Intel, the Intel logo, Intel AppUp, Intel Atom,
SpeedStep, Intel Sponsors of Tomorrow., the Intel Sponsors of Tomorrow. logo, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale,
InTru, the InTru logo, InTru soundmark, Itanium, Itanium Inside, MCS, MMX, Moblin, Pentium, Pentium Inside, skool, the skool logo,
Sound Mark, The Journey Inside, vPro Inside, VTune, Xeon, and Xeon Inside are trademarks of Intel Corporation in the U.S. and other
countries.

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

Microsoft, Windows, Visual Studio, Visual C++, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation
in the United States and/or other countries.

Copyright (C) 2011, Intel Corporation. All rights reserved.
Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the “Intel® Compiler User and Reference Guides” under “Compiler Options." Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools 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 Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® 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.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101
## Intel® Parallel Studio XE 2011
Powerful tools to create fast, reliable and secure code

<table>
<thead>
<tr>
<th>Phase</th>
<th>Productivity Tool</th>
<th>Feature</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Build &amp; Debug</td>
<td><strong>Intel® Composer XE</strong></td>
<td>C/C++ and Fortran compilers, performance libraries, and parallel models</td>
<td>Application performance, scalability and quality for current multicore and future many-core systems.</td>
</tr>
<tr>
<td>Advanced Verify</td>
<td><strong>Intel® Inspector XE</strong></td>
<td>Memory &amp; threading error checking tool for higher code reliability &amp; quality</td>
<td>Increases productivity and lowers cost, by catching memory and threading defects early</td>
</tr>
<tr>
<td>Advanced Tune</td>
<td><strong>Intel® VTune™ Amplifier XE</strong></td>
<td>Performance Profiler to optimize performance and scalability</td>
<td>Removes guesswork, saves time, makes it easier to find performance and scalability bottlenecks Combines ease of use with deeper insights.</td>
</tr>
</tbody>
</table>

**+ Extra Feature in Parallel Studio XE suites:**
Static Security Analyzer
## Intel® Parallel Studio XE 2011
Powerful tools to create fast, reliable and secure code

<table>
<thead>
<tr>
<th>Phase</th>
<th>Productivity Tool</th>
<th>Feature</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Build &amp; Debug</td>
<td><strong>Intel® Composer XE</strong></td>
<td>C/C++ and Fortran compilers, performance libraries, and parallel models</td>
<td>Application performance, scalability and quality for current multicore and future many-core systems.</td>
</tr>
<tr>
<td>Advanced Verify</td>
<td><strong>Intel® Inspector XE</strong></td>
<td>Memory &amp; threading error checking tool for higher code reliability &amp; quality</td>
<td>Increases productivity and lowers cost, by catching memory and threading defects early</td>
</tr>
<tr>
<td>Advanced Tune</td>
<td><strong>Intel® VTune™ Amplifier XE</strong></td>
<td>Performance Profiler to optimize performance and scalability</td>
<td>Removes guesswork, saves time, makes it easier to find performance and scalability bottlenecks Combines ease of use with deeper insights.</td>
</tr>
</tbody>
</table>

**Extra Feature in Parallel Studio XE suites:**
Static Security Analyzer
Intel® Inspector XE 2011
Advancing Application Reliability, Code Quality and Security

• Powerful Robust Dynamic Analysis
  – Memory errors
    – Invalid Memory Accesses
    – Memory Leaks
    – Uninitialized Memory Accesses
    – Improper usage of Memory API(s)
    – Resource Leaks (Windows only)
  – Threading Errors
    – Data Races
    – Deadlock/Lock Hierarchy Violation
    – Cross Thread Stack Memory Accesses

• Static Security Analysis (requires suite-level license)
  – 250 Different Coding Issues
  – Buffer overflows and boundary violations
  – Use of uninitialized variables and objects

• Available for Windows* and Linux* OS
Creating a new project

Intel Inspector XE uses your project properties if you use Microsoft Visual Studio* software.
Starting a new analysis

1. Select Analysis Type

2. Fine-tune Data Collected*

3. Click Start

*Collection time and space tradeoffs for each analysis type
Memory Analysis

Code Locations grouped into Problems

Code Locations show problematic source lines

Filters make problem management easy

Detect Memory Problems

<table>
<thead>
<tr>
<th>Problems</th>
<th>Sources</th>
<th>Modules</th>
<th>Object Size</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>nqueens_memory.f90</td>
<td>memory_issues.exe</td>
<td>New</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>nqueens_memory.f90</td>
<td>memory_issues.exe</td>
<td>71</td>
<td>New</td>
</tr>
</tbody>
</table>

Filters

- Severity: Error
- Problem: Memory leak
- Source: nqueens_memory.f90
- Module: memory_issues.exe
- State: New
- Suppressed: Not suppressed
- Investigated: Not investigated
Memory Analysis – Memory Leak

Call stack data for each Code Location

Sources view provides complete source code

To fix this leak we need to add a deallocation

```c
!deallocate the pointer to avoid a memory leak
deallocate (correct_solution)
```
Memory Analysis – Uninitialized memory

To fix this we need to initialize the memory

```fortran
allocate (queens(size))
!initialize the queens array to 0 to avoid "uninitialized memory access" issue
queens = 0
```
Threading Analysis – Data Race #1

Choose a new analysis type

Consistent Summary view for all analysis types

This data race needs synchronization

```c
!$OMP ATOMIC
nrOfSolutions = nrOfSolutions + 1
```
Data race caused by memory reuse. Each thread needs a local copy.

! Make copy of queens array
lcl_queens = queens
Static Security Analysis
Feature Overview

• Static Security Analysis (SSA) is a suite-level feature that uses:
  – Intel® Composer XE (Compiler) for the analysis
  – Intel Inspector XE to display and manage results

• Analyzes source code to detect problems

• What does SSA detect?
  - 250 Different Coding Issues
  - Buffer overflows and boundary violations
  - Use of uninitialized variables and objects
  - Memory and other resource leaks
  - Incorrect usage of pointers and dynamically allocated memory
  - Incorrect use of OpenMP* and Intel® Cilk™ Plus directives
  - Error-prone C++ and Fortran language usage
  - Misuse of string, memory, and formatting library routines
  - Etc…
Static Security Analysis – Summary View

- Ensure that the array is allocated before use
- Always check for optional argument before use
- A warning about unused code
Command Line Interface

- **inspxe-cl** is the command line.
  - **Windows:** `C:\Program Files\Intel\Inspector XE 2011\bin[32|64]\inspxe-cl.exe`
  - **Linux:** `/opt/intel/inspector_xe_2011/bin[32|64]/inspxe-cl`

- To get detailed help:
  `inspxe-cl -help`

- Get Command Line from GUI

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

More Help is available with the Online Documentation
Regression Testing

• **Create Baseline Suppression File:**
  
  $> \text{inspxe-cl -collect ti2 -r BaseLineResults -- App.exe}
  
  $> \text{inspxe-cl -create-suppression-file myThread.sup -result-dir BaseLineResults}

• **Nightly Correctness Regression Testing:**
  
  $> \text{inspxe-cl -collect ti2 -suppression-file MyThread.sup -r NightlyTestResults -- App.exe}

  [...Stuff Deleted...]

  0 new problem(s) found

**CLI is very useful for running Intel Inspector XE as part of Correctness regression testing**
More Information:

Free Evaluation Copy:
www.intel.com/software/products/eval/

Additional Webinars: