Agenda

Introduction

• The Advisor XE Workflow
  - Survey
  - Add Annotations
  - Model Performance Suitability
  - Check Correctness
  - Add Parallel Framework

Summary
What is Intel® Advisor XE?
Threading Assistant

• Productivity boosting parallelism modeling tool
• A methodology and set of tools to help you easily add correct and effective **parallelism** to your program
• Helps parallelize software with confidence
• For C/C++/Fortran on Windows*/Linux* and C# .NET on Windows*
Simplify and Speed Threading Design
Intel® Advisor XE – Threading Assistant

The Challenge of Parallel Design:
• Need to implement to measure performance
• Implementation is time consuming
• Disrupts regular product development
• Testing difficult without tools

Intel Advisor XE Separates Design & Implementation
• Fast exploration of multiple options
• Find errors before implementation
• Design without disrupting development

---

Add Parallelism with Less Effort, Less Risk and More Impact
Amdahl's Law

(paraphrased) “The benefit from parallelism is limited by the computation which remains serial”

If you perfectly execute ½ of your application in parallel you will achieve < 2x speedup

The implication of this is that you must focus your attention where your application spends its time.
Agenda

Introduction

• The Advisor XE Workflow
  - Survey
  - Add Annotations
  - Model Performance Suitability
  - Check Correctness
  - Add Parallel Framework

Summary
Design Then Implement
Intel® Advisor XE 2013 – Threading Assistant

Design Parallelism
1) Analyze it.
2) Design it. (Compiler ignores these annotations.)
3) Tune it.
4) Check it.

Implement Parallelism
5) Do it!

Design Parallelism
- No disruption to regular development
- All test cases continue to work
- Tune and debug the design before you implement it

Implement Parallelism
Less Effort, Less Risk, More Impact
Agenda

Introduction

• The Advisor XE Workflow
  - Survey
  - Add Annotations
  - Model Performance Suitability
  - Check Correctness
  - Add Parallel Framework

Summary
Intel® Advisor Survey

Where should I add parallelism?

Find the places that are important to your application
Check different loop metrics and start adding annotations
Two Candidate loops

- **56%**: POTENTIAL::start (loop)

```
for (int i = 0; i < constants.POT_ITERATION; i++)
{
    potentialTotal = 0.0;
    computePot_st();
    if (i % 10 == 0)
        Console.WriteLine("{0} - (Potential - {1:F5})", i, pot);
    updatePositions();
}
```

- **41.8%**: NBODIES::start (loop)

```
public void start()
{
    for (int i = 0; i < constants.NB_NUM_BODIES; i++)
        body[i] = new body();

    // Loop over various sizes of the problem
    for (int n = 2; n <= constants.NB_NUM_BODIES; n *= 2)
    {
        startBodies(n);
        runBodies(n);
    }
}
```
Agenda

Introduction

• The Advisor XE Workflow
  - Survey
  - Add Annotations
  - Model Performance Suitability
  - Check Correctness
  - Add Parallel Framework

Summary
Intel® Advisor XE Annotations

3 primary concepts to create a parallel program model

• SITE (where should I focus)
  - A region of code in your application you want to transform into parallel code

• TASK (what should I do in parallel)
  - The region of code in a SITE you want to execute in parallel with the rest of the code in the SITE

• LOCK (what should be serial)
  - Mark regions of code in a TASK which must be serialized

NOTE
• All of these regions may be nested
• You may create more than one SITE
• For C/C++: simple macros - work with any compiler
for(int i=0;i<sourceFiles.size();i++){

drawing_area drawing(startx, totaly-y, stopx-startx)

video->next_frame()

...}

Propose how you would like to partition your algorithm
Propose how you would like to partition your algorithm
Add Annotations

```cpp
ANNOTATE_SITE_BEGIN(allRows);
for(int i=0;i<sourceFiles.size();i++){
    ANNOTATE_ITERATION_TASK (eachRow);
    drawing_area drawing(startx, totaly-y, stopx-startx);
    video->next_frame()
    ...
}
ANNOTATE_SITE_END(allRows);
```

Propose how you would like to partition your algorithm
Add Annotations

```c
ANNOTATE_SITE_BEGIN(allRows);
for(int i=0;i<sourceFiles.size();i++){
    ANNOTATE_ITERATION_TASK (eachRow);
    drawing_area drawing(startx, totaly-y, stopx-startx);

    ANNOTATE_LOCK_ACQUIRE(0);
    video->next_frame()
    ANNOTATE_LOCK_RELEASE(0);
...
}
ANNOTATE_SITE_END(allRows);
```

Propose how you would like to partition your algorithm
Add Annotations: Visual-Studio Integrated

```c
for (int y = starty; y < stopy; y++)
{
    drawing_area drawing(startx, totaly-y, stopx-startx, 1);
    //...
    video->next_frame();
}
```

```
ANNOTATE_SITE_BEGIN( allRows );
for (int y = starty; y < stopy; y++)
{
    ANNOTATE_ITERATION_TASK( eachRow );
    drawing_area drawing(startx, totaly-y, stopx-startx, 1);
    //...
    video->next_frame();
}
ANNOTATE_SITE_END();
```
Add Annotations:
**Standalone, Windows or Linux**

```c
for (int x = startx; x < stopx; x++) {
    color_t c = render_one_pixel(x, y, m_storage.lock);
    drawing.put_pixel(c);
}
```

Iterative Loop Annotations Example

```
// To copy compiler options, select Build Settings topic from the drop-down list.

#include "advisor-annotate.h"
// Add to each module that contains Intel Advisor XE annotations

// Begin a parallel code region (parallel site)
ANNOTATE_SITE_BEGIN( MySite1 ):// Place before the loop control statement
// loop control statement
// If the entire loop body is not a single task, select a different topic from the list
ANNOTATE_ITERATION_TASK( MyTask? ):// Place at the start of loop body. This iterative-task annotation
// loop body
ANNOTATE_SITE_END();:// End the parallel code region, after task execution completes
```

```c
ANNOTATE_SITE_BEGIN( allRows );
for (int y = starty; y < stopy; y++)
{
    ANNOTATE_ITERATION_TASK( eachRow );
    drawing_area drawing(startx, totaly-y, stopx-startx, 1);
    //...
    video->next_frame();
}
ANNOTATE_SITE_END();
```
Agenda

Introduction

• The Advisor XE Workflow
  - Survey
  - Add Annotations
  - Model Performance Suitability
  - Check Correctness
  - Add Parallel Framework

Summary
Suitability – How Fast Will It Be?

Ideal scalability: linear
- The speedup increases linear to the number of cores

Scalability can be limited by:
- Serial execution (Amdahl’s law)
- Load balancing
- Dataset size (Gustafson’s law)
- Task granularity and scheduler overhead
- Lock contention
Suitability – Compare Multiple Sites

Analyze the performance of your proposal

Estimated Overall Speed-up

Recommended Improvement

Scalability Graph
Agenda

Introduction

• The Advisor XE Workflow
  - Survey
  - Add Annotations
  - Model Performance Suitability
  - Check Correctness
  - Add Parallel Framework

Summary
Correctness – Any Data Sharing Bugs?

4 Memory reuse conditions found!

Observations help identify problem

Analyze your design for errors
Summary – Make informed decisions

Intel Advisor XE helps you choose where to add parallelism to your program

Potential program gain**: 2.41x (8 CPUs, Microsoft TPL Threading Model)

The most time-consuming (hot) functions found during Survey analysis appear below. Consider adding parallel site and task annotations around these functions so Suitability and Correctness can predict their parallel behavior.

<table>
<thead>
<tr>
<th>Function</th>
<th>Source Location</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisplayClassFlt&lt;ForWorkers&gt; b c</td>
<td></td>
<td>10.9345s</td>
</tr>
<tr>
<td>VTuneAmplifierXE:Examples:NBODIESstart</td>
<td>nbodies.cs:105</td>
<td>7.3358s</td>
</tr>
<tr>
<td>VTuneAmplifierXE:Examples:POTENTIAL:computePot st</td>
<td>potential.cs:41</td>
<td>2.9775s</td>
</tr>
</tbody>
</table>

Collection Details.

Survey
- Collection started: 21 March 2012, 6:01:26 PM
- Collection finished: 21 March 2012, 6:01:39 PM
- Elapsed time: 00 min 13 sec
- Collector Log: See log
- Application Output: See output
- Collector Command Line: See command line
and then Repeat...

You do not have to choose the perfect answer the first time, so you can go back and modify your choices

Iterative refinement will either

• Create a suitable and correct annotation proposal
• Conclude no viable sites are possible

Efficiently arriving at either answer is valuable
Agenda

Introduction

• The Advisor XE Workflow
  - Survey
  - Add Annotations
  - Model Performance Suitability
  - Check Correctness
  - Add Parallel Framework

Summary
## Add Parallel Framework

### OpenMP example

<table>
<thead>
<tr>
<th>Serial C/C++ and Fortran Code with Intel Advisor XE Annotations</th>
<th>Parallel C/C++ and Fortran Code using OpenMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>// Parallelize data - one task within a C/C++ counted loop</td>
<td>// Parallelize data - one task, C/C++ counted loops</td>
</tr>
<tr>
<td>ANNOTATE_SITE_BEGIN(site);</td>
<td>#pragma omp parallel for</td>
</tr>
<tr>
<td>for (i = lo; i &lt; n; ++i) {</td>
<td>for (int i = lo; i &lt; n; ++i) {</td>
</tr>
<tr>
<td>ANNOTATE_ITERATION_TASK(task);</td>
<td>statement;</td>
</tr>
<tr>
<td>statement;</td>
<td>}</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>

| ! Parallelize data - one task within a Fortran counted loop    | ! Parallelize data - one task with a Fortran counted loop |
| call annotate_site_begin("site1")                           | !$omp parallel do                           |
| do i = 1, N                                                  | do i = 1, N                                 |
|     call annotate_iteration_task("task1")                    |     statement                               |
|     statement                                                 | end do                                      |
| end do                                                       | !$omp end parallel do                       |
| call annotate_site_end                                        |                                             |
Agenda

Introduction

• The Advisor XE Workflow
  - Survey
  - Add Annotations
  - Model Performance Suitability
  - Check Correctness
  - Add Parallel Framework

Summary
**Summary**

The Intel Advisor XE is a unique tool that:

- helps you work smarter though detailed modeling
- guides you through the necessary steps
- leaves you in control of code and architectural choices
- lets you transform serial algorithms into parallel form faster

**The parallel modeling methodology:**

- maintains your original application’s semantics and behavior
- helps find the natural opportunities to exploit parallel execution
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 © , Intel Corporation. All rights reserved. Intel, the Intel logo, Xeon, Core, VTune, and Cilk 
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