Agenda

- Motivation
- Threading Advisor
  - Threading Advisor Workflow
  - Advisor Interface
  - Survey Report
  - Annotations
  - Suitability Analysis
  - Dependencies Analysis
- Vectorization Advisor & Roofline
  - Vectorization Advisor recap
  - Roofline
  - Memory Access Patterns Analysis
  - Dependencies Analysis
- Summary
The “Free Lunch” is over, really
Processor clock rate growth halted around 2005

Software must be parallelized to realize all the potential performance
Moore’s Law Is Going Strong
Hardware performance continues to grow exponentially

“We think we can continue Moore's Law for at least another 10 years.”

Intel Senior Fellow
Mark Bohr
2015
Changing Hardware Impacts Software
More Cores → More Threads → Wider Vectors

High performance software must be both
- Parallel (multi-thread, multi-process)
- Vectorized

*Product specification for launched and shipped products available on ark.intel.com.
Vectorize & Thread or Performance Dies
Threaded + Vectorized can be much faster than either one alone

"Automatic" Vectorization Not Enough
Explicit pragmas and optimization often required

The Difference Is Growing With Each New Generation of Hardware

Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.
Vectorization & Threading Critical on Modern Hardware

- **LIBOR**: 90x
- **Monte Carlo Asian Options**: 90x
- **Black Scholes**: 28x
- **Monte Carlo America Options**: 60x

Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system. For more complete information about performance and benchmark results, visit [www.intel.com/benchmarks](http://www.intel.com/benchmarks).


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, & SSSE3 instruction sets & 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
THREADING ADVISOR
Serial Modeling Has Multiple Benefits
Intel® Advisor

1) Your application can’t fail due to bugs caused by incorrect parallel execution. (It's running serially.)

2) You can easily experiment with several different proposals before committing to the expense of implementation.
   a) Measure performance - focus on where it will pay off.
   b) Predict scalability, load balancing and overheads.
   c) Predict (and avoid) data races

3) All of your test suites should still pass. Validate the correctness of your transformations.

4) You can use Advisor on partially or completely parallelized code.

Design, measure and test before implementation
Threading Advisor Workflow

- Use the Survey to find good potential threading sites.
  - Optionally, follow up with Trip Counts to find information about iteration and call counts.
- Annotate your code.
- Use Suitability to predict how much performance improvement the proposed threading model will create under specific, editable conditions.
- Use Dependencies to determine whether the proposed model is safe, and what needs to be done to correct it.
Survey Report
Threading Advisor

Tip:
Survey sorts by Self Time by default. This is good for Vector Advisor, but for Threading Advisor, you may want to sort by Total Time.

- The Survey Report has lots of information, but most of it is more relevant to Vector Advisor.
- Look for outer loops or functions with high Total Time.
- In this example, setQueen has a high Total Time. It's recursive, but is originally called from a loop in Solve. That makes the loop in Solve a good potential candidate.
Annotating Your Code

• Annotations are notes to Advisor. They are not parallelization commands. They do not affect the way the program itself runs.

• They mark places Advisor should treat as locks or parallel sites.

• To use annotations, you must include the appropriate header/module.

<table>
<thead>
<tr>
<th>C/C++</th>
<th>FORTRAN</th>
<th>C#</th>
</tr>
</thead>
<tbody>
<tr>
<td>In source files where annotations are used, add: include &lt;advisor-annotate.h&gt;</td>
<td>In source files where annotations are used, add: use advisor_annotate</td>
<td>In source files where annotations are used, add: using AdvisorAnnotate;</td>
</tr>
<tr>
<td>Add &lt;install_dir&gt;/include to your include directories.</td>
<td>Add &lt;install_dir&gt;/include to your include directories.</td>
<td>Add the C# annotations definition file to your project.</td>
</tr>
</tbody>
</table>

• The Advisor User's Guide contains a section on Annotations with full documentation, examples, and instructions on the above if you forget.
Common Annotations

• Site Annotations
  • Indicate locations which contain a number of tasks which can run in parallel with each other. Code within the site that is not part of a task is executed only by the entering thread, but can still run in parallel with the tasks.
  • Consist of a Site Begin and a Site End annotation.

• Task Annotations
  • Indicate chunks of code which can run in parallel with other tasks in the same site, including any additional copies of themselves which may be launched.
  • Consist of a Task Begin and a Task End annotation.

• Iteration Task Annotations
  • Indicate that the entirety of a loop’s contents should be considered as a task.
  • Consist only of an Iteration Task annotation. The end is implied.

• Lock Annotations
  • Indicate where Advisor should simulate locking.
  • Consist of a Lock Acquire annotation and a Lock Release annotation.
Suitability Analysis

- Using your annotations, Advisor models how the program would behave in parallel, and predicts performance in specified hypothetical circumstances.

Indicate how many CPUs, what kind of system, and what threading model to make predictions on.

Select a site to view site-specific info in the bottom pane.

Calculate on the assumption you’re using framework constructs that address these issues.

See how things would change if you altered the duration and/or number of iterations/tasks.
Suitability Analysis

- The white dots are the estimated number of times speedup at the given number of CPUs.
- The bullseye is placed on the one corresponding to the number of CPUs you have selected in the controls.
  - If your bullseye is in:
    - **Red:** No benefit, possible slowdown.
    - **Yellow:** Poor scaling, needs improvement.
    - **Green:** Ideal scaling!
- Advisor provides warnings when it predicts that your specifications would lead to issues like load imbalance.
Dependencies Analysis
Threading Advisor

• This is the same analysis as in Vectorization Advisor. It works with annotations as well as selections in the survey report.

• Add lock annotations or reorganize code to resolve reported dependencies, then re-run the analysis to confirm the problem has been resolved.

• Run suitability again to check that you still get good improvement.

• Once you’re happy with Advisor’s predictions, replace the annotations with actual parallelism and locks.
Add Parallel Framework

Here is the list of source locations

Here are templates for popular parallel frameworks

Intel® Advisor

- Contains overhead metrics for popular parallel frameworks
- Quickly prototype and evaluate alternatives
- Detailed help pages for popular parallel frameworks
Intel® Advisor Workflow

Vectorization

Select loops with potential vector dependencies
Dependences
Force vectorization if appropriate

Select loops with poor vector efficiency
Memory Access Patterns

Examine Roofline

Examining Results

Build in Release

Trip Count

Graphical representation of the workflow:

Threaded

Suitability
Select best threading candidate
Dependencies
Correct dependencies and rebuild

Implement Threading

Annotate Source, rebuild, and examine

Back to Start

Optimization Notice
Copyright © 2017, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
VECTORIZATION ADVISOR & ROOFLINE
Vectorization Advisor Workflow

- **Survey** is the bread and butter of Vectorization Advisor! All else builds on it!
- **Trip Counts** adds onto Survey and enables the **Roofline**.
- **Dependencies** determines whether it's safe to force a scalar loop to vectorize.
- **Memory Access Patterns** diagnoses vectorization inefficiency caused by poor memory striding.

Flowchart:
- **Build in Release**
  - **Survey**
    - **Trip Counts**
      - **Examine Roofline**
      - **Examine Results**
      - **Select loops with potential vector dependencies**
        - **Dependencies**
          - **Memory Access Patterns**
            - **Force vectorization if appropriate**
            - **Improve vectorization**
        - **Select loops with poor vector efficiency**
          - **Back to Start**
Survey
Vectorization Advisor

**Tip:**
For vectorization, you generally only care about loops. Set the type dropdown to “Loops”.

**Function/Loop Icons**
- Scalar Function
- Vector Function
- Scalar Loop
- Vector Loop

**Optimization Notice**
For vectorization, you generally only care about loops. Set the type dropdown to “Loops”.

**Vectorizing a loop is usually best done on innermost loops.**
Since it effectively divides duration by vector length, you want to target loops with high self time.

**Expand a vectorized loop to see it split into body, peel, and remainder (if applicable).**

**Advisor advises you on potential vector issues.**
This is often your cue to run MAP or Dependencies. Click the icon to see an explanation in the bottom pane.

**Efficiency is important!**
\[
\text{Efficiency} = 100\% \times \frac{\text{Speedup}}{\text{Vec. Length}}
\]
The black arrow is 1x. Gray means you got less than that. Gold means you got more. You want to get this value as high as possible!

**The Intel Compiler embeds extra information that Advisor can report in addition to its sampled data, such as why loops failed to vectorize.**
What is a Roofline Chart?

A Roofline Chart plots application performance against hardware limitations.

• Where are the bottlenecks?
• How much performance is being left on the table?
• Which bottlenecks can be addressed, and which should be addressed?
• What’s the most likely cause?
• What are the next steps?

Roofline first proposed by University of California at Berkeley:

Cache-aware variant proposed by University of Lisbon:
*Cache-Aware Roofline Model: Upgrading the Loft*, 2013
Roofline Metrics

Roofline is based on Arithmetic Intensity (AI) and FLOPS.

- **Arithmetic Intensity**: FLOP / Byte Accessed
  - This is a characteristic of your algorithm

- **FLOPS**: Floating-Point Operations / Second
  - Is a measure of an implementation (it achieves a certain FLOPS)
  - *And* there is a maximum that a platform can provide
Plotting a Roofline Chart

A Roofline Chart uses AI as its X axis and FLOPS as its Y axis.

The maximum FLOPS as a product of ops/byte (AI) and maximum bytes supplied per second is a diagonal line.

The CPU's maximum FLOPS can be plotted as a horizontal line.

A loop or function can be plotted as a point on the graph.
Classic vs. Cache-Aware Roofline

Intel® Advisor uses the Cache-Aware Roofline model, which has a different definition of Arithmetic Intensity than the original ("Classic") model.

**Classical Roofline**
- Traffic measured from one level of memory (usually DRAM)
- AI may change with data set size
- AI changes as a result of memory optimizations

**Cache-Aware Roofline**
- Traffic measured from all levels of memory
- AI is tied to the algorithm and will not change with data set size
- Optimization does not change AI*, only the performance

*Compiler optimizations may modify the algorithm, which may change the AI.
Ultimate Performance Limits

Performance cannot exceed the machine's capabilities, so each loop is ultimately limited by either compute or memory capacity.
Sub-Roofs and Current Limits

Additional roofs can be plotted for specific computation types or cache levels.

These sub-roofs can be used to help diagnose bottlenecks.
Cache-Aware Roofline Concept

- Prior to collecting data, Advisor runs quick benchmarks to measure hardware limitations.
  - Computational limitations
  - Memory Bandwidth limitations
- These form the performance “roofs”.
- Loops and functions have algorithms and therefore a specific AI.
- Their performance in FLOPS is also measured.
- Optimization changes performance. The goal is to go as far up as possible.

Video Available: Roofline Analysis in Intel® Advisor 2017

Roofline first proposed by University of California at Berkeley: Roofline: An Insightful Visual Performance Model for Multicore Architectures, 2009
Cache aware variant proposed by Technical University of Lisbon: Cache-Aware Roofline Model: Upgrading the Loft, 2013
Cache-Aware Roofline Usage

- Target the bigger, redder dots that are farthest below the top roofs first.
  - Big and red = takes up a lot of time.
  - Far below roofs = lots of room for improvement.

- Roofs below: unlikely bottlenecks, unless you're just above them.

- Roofs above: very likely bottlenecks! The closest roofs above are the most likely causes of bottlenecking.

Note:

Not every algorithm can break every roof! Some algorithms inherently can't avoid certain bottlenecks.
Cache-Aware Roofline

Next Steps

If under or near a memory roof...

- Try a MAP analysis. Make any appropriate cache optimizations.
- If cache optimization is impossible, try reworking the algorithm to have a higher AI.

If under the Vector Add Peak

Check “Traits” in the Survey to see if FMAs are used. If not, try altering your code or compiler flags to induce FMA usage.

If just above the Scalar Add Peak

Check vectorization efficiency in the Survey. Follow the recommendations to improve it if it’s low.

If under the Scalar Add Peak...

Check the Survey Report to see if the loop vectorized. If not, try to get it to vectorize if possible. This may involve running Dependencies to see if it’s safe to force it.

Optimization Notice

Copyright © 2017, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
Memory Access Patterns Analysis
Collecting a MAP

• If you have low vector efficiency, or see that a loop did not vectorize because it was deemed “possible but inefficient”, you may want to run a MAP analysis.

• Advisor will also recommend a MAP analysis if it detects a possible inefficient access pattern.

• Memory access patterns affect vectorization efficiency because they affect how data is loaded into and stored from the vector registers.

• Select the loops you want to run the MAP on using the checkboxes. It may be helpful to reduce the problem size, as MAP only needs to detect patterns, and has high overhead.
  • Note that if changing the problem size requires recompiling, you will need to re-collect the survey before running MAP.
Memory Access Patterns Analysis

Reading a MAP

• MAP is color coded by stride type. From best to worst:
  • **Blue** is unit/uniform (stepping by 1 or 0)
  • **Yellow** is constant (stepping a set distance)
  • **Red** is variable (a changing step distance)

• Click a loop in the top pane to see a detailed report below.
  • The strides that contribute to the loop are broken down in this table.
  • Important information includes the size of the stride, the variable being accessed, and the source.
  • Not all strides will come from your code!
Dependencies Analysis
Vectorization Advisor

- Generally, you don't need to run Dependencies analysis unless Advisor tells you to. It produces recommendations to do so if it detects:
  - Loops that remained unvectorized because the compiler was playing it safe with autovectorization.
  - Outer loop vectorization opportunities
  - Use the survey checkboxes to select which loops to analyze.
  - If no dependencies are found, it's safe to force vectorization.
  - Otherwise, use the reported variable read/write information to see if you can rework the code to eliminate the dependency.
Summary

Survey – Find the most promising sites for threading, see the meat of the vectorization information, and get recommendations from Advisor.

Trip Counts & FLOPS – Add to your Survey report to help fine-tune vector efficiency and capability, as well as unlock the powerful Roofline to visualize your bottlenecks and help direct your efforts.

Suitability – Predict how well your proposed threading model will scale under certain conditions quickly and easily.

Dependencies – Prove or disprove the existence of parallel dependencies and learn how to fix them.

Memory Access Patterns – See how you traverse your data and how it affects your vector efficiency and cache bandwidth usage.
Threading Advisor Demo
## Configurations for 2010-2017 Benchmarks

### Platform Hardware and Software Configuration

<table>
<thead>
<tr>
<th>Platform</th>
<th>Core Frequency</th>
<th>Cores/Sockets</th>
<th>Num Sockets</th>
<th>L1 Cache</th>
<th>L2 Cache</th>
<th>L3 Cache</th>
<th>Memory</th>
<th>Memory Frequency</th>
<th>Memory Access</th>
<th>H/W Prefetchers Enabled</th>
<th>HT Enabled</th>
<th>Turbo Enabled</th>
<th>C States</th>
<th>O/S Name</th>
<th>Operating System</th>
<th>Compiler Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSM</td>
<td>3.33 GHZ</td>
<td>6</td>
<td>2</td>
<td>32K</td>
<td>256K</td>
<td>12 MB</td>
<td>48 MB</td>
<td>1333 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Disabled</td>
<td>Fedora 20</td>
<td>3.11.10-301.fc20</td>
<td>icc version 17.0.2</td>
</tr>
<tr>
<td>SNB</td>
<td>2.9 GHZ</td>
<td>8</td>
<td>2</td>
<td>32K</td>
<td>256K</td>
<td>20 MB</td>
<td>64 GB</td>
<td>1600 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Disabled</td>
<td>Fedora 20</td>
<td>3.11.10-301.fc20</td>
<td>icc version 17.0.2</td>
</tr>
<tr>
<td>IVB</td>
<td>2.7 GHZ</td>
<td>12</td>
<td>2</td>
<td>32K</td>
<td>256K</td>
<td>30 MB</td>
<td>64 GB</td>
<td>1867 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Disabled</td>
<td>RHEL 7.1</td>
<td>3.10.0-229.el7.x86_64</td>
<td>icc version 17.0.2</td>
</tr>
<tr>
<td>HSW</td>
<td>2.2 GHZ</td>
<td>18</td>
<td>2</td>
<td>32K</td>
<td>256K</td>
<td>46 MB</td>
<td>128 GB</td>
<td>2133 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Disabled</td>
<td>Fedora 20</td>
<td>3.15.10-200.fc20.x86_64</td>
<td>icc version 17.0.2</td>
</tr>
<tr>
<td>BDW</td>
<td>2.3 GHZ</td>
<td>18</td>
<td>2</td>
<td>32K</td>
<td>256K</td>
<td>46 MB</td>
<td>256 GB</td>
<td>2400 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Disabled</td>
<td>RHEL 7.0</td>
<td>3.10.0-327.el7.x86_64</td>
<td>icc version 17.0.2</td>
</tr>
<tr>
<td>BDW</td>
<td>2.2 GHZ</td>
<td>22</td>
<td>2</td>
<td>32K</td>
<td>256K</td>
<td>56 MB</td>
<td>128 GB</td>
<td>2133 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Disabled</td>
<td>CentOS 7.2</td>
<td>3.10.0-314.10.2.el7.x86_64</td>
<td>icc version 17.0.2</td>
</tr>
<tr>
<td>SKX</td>
<td>2.5 GHZ</td>
<td>28</td>
<td>2</td>
<td>32K</td>
<td>1024K</td>
<td>40 MB</td>
<td>192 GB</td>
<td>2666 MHz</td>
<td>NUMA</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Disabled</td>
<td>CentOS 7.3</td>
<td>3.10.0-314.10.2.el7.x86_64</td>
<td>icc version 17.0.2</td>
</tr>
</tbody>
</table>

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. For more information go to http://www.intel.com/performance

---

### 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

Performance measured in Intel Labs by Intel employees

Copyright © 2017, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.
Legal Disclaimer & Optimization Notice

Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

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. For more complete information visit www.intel.com/benchmarks.

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.

Copyright © 2018, Intel Corporation. All rights reserved. Intel, Xeon, Xeon Phi, Core, VTune, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.