Roofing a house
roofline model in Intel® Advisor

Roman Belenov, Zakhar Matveev, Igor Kaleturin, Julia Fedorova
Problem

What limits my code?

How can I do better?

- Peak performance
- SIMD vectorization
- Multithreading
- Blocking

Peak bandwidth
Roofline model


**Cache Aware(CARM):** A. Ilic et.al. “Cache-aware Roofline model: Upgrading the loft” IEEE CAL (2014)

AI = Compute (FLOP) / Bandwidth (bytes)

Traditional roofline – only DRAM bytes count
CARM – count all loads and stores
## Intel® Advisor

**Optimization workflow**

1. **Survey Target**
   - Collect
   - 1.1 Find Trip Counts and FLOPS
     - Collect

**Profile**

- Function Call Sites and Loops
- Loop characteristics

**Issues**

- Why No Vectorization?
  - Unrolled by 8
  - FMA
  - Float32

**Code properties**

- Instructions
- Data Types

**Recommendations**

### Issue: Potential underutilization of FMA instructions

Your current hardware supports the AVX2 instruction set architecture (ISA), which enables the use of fused multiply-add (FMA) instructions. Improve performance by utilizing FMA instructions.

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Force vectorization if possible</td>
<td>Low</td>
</tr>
</tbody>
</table>

Loop contains FMA instructions (so vectorization could be beneficial) but is not vectorized. To fix: Review corresponding compiler diagnostics to check if vectorization enforcement is possible and profitable.

---

**Read More:**

- Vectorization Resources for Intel® Advisor Users
Roofline integrated

Switch between the roofline and the grid

Hotspots marked by size and color

Source for the selected loop

Loop data hint

CARM-based AI
Under the hood

**FLOPs**
- Binary Instrumentation
- Does not rely on CPU counters

**Bytes**
- Binary Instrumentation
- Counts operands size (not cachelines)

**Seconds**
- User-mode sampling
- Root access not needed

**Roofs**
- Microbenchmarks
- Actual peak for the current configuration

Performance = Flops/seconds

Al = Flop/byte
Case study - Minighost

Models heat diffusion via stencil computations
Part of NERSC-8/Trinity benchmarks
http://www.nersc.gov/users/computational-systems/cori/nersc-8-procurement/trinity-nersc-8-rfp/nersc-8-trinity-benchmarks/minighost/

3D grid with 27-point stencil used

\[
\text{SLICE\_BACK} = \text{GRID}(I-1,J-1,K-1) + \text{GRID}(I-1,J,K-1) + \text{GRID}(I-1,J+1,K-1) + \&
\text{GRID}(I,J-1,K-1) + \text{GRID}(I,J,K-1) + \text{GRID}(I,J+1,K-1) + \&
\text{GRID}(I+1,J-1,K-1) + \text{GRID}(I+1,J,K-1) + \text{GRID}(I+1,J+1,K-1)
\]

\[
\text{SLICE\_MINE} = \text{GRID}(I-1,J-1,K) + \text{GRID}(I-1,J,K) + \text{GRID}(I-1,J+1,K) + \&
\text{GRID}(I,J-1,K) + \text{GRID}(I,J,K) + \text{GRID}(I,J+1,K) + \&
\text{GRID}(I+1,J-1,K) + \text{GRID}(I+1,J,K) + \text{GRID}(I+1,J+1,K)
\]

\[
\text{SLICE\_FRONT} = \text{GRID}(I-1,J-1,K+1) + \text{GRID}(I-1,J,K+1) + \text{GRID}(I-1,J+1,K+1) + \&
\text{GRID}(I,J-1,K+1) + \text{GRID}(I,J,K+1) + \text{GRID}(I,J+1,K+1) + \&
\text{GRID}(I+1,J-1,K+1) + \text{GRID}(I+1,J,K+1) + \text{GRID}(I+1,J+1,K+1)
\]

\[
\text{GRID\_NEXT}(I,J,K) = (\text{SLICE\_BACK} + \text{SLICE\_MINE} + \text{SLICE\_FRONT}) / 27.0
\]

Let’s focus on single-node optimizations
Easy task first

Suspicious memory copy hotspot

The caller in application code
Chip away at the surplus

Before – 7.65 GFLOP/s
After – 10.65 GFLOP/S
≈40% improvement

The result is copied back

Swap indices instead
Here comes the roofline chart

Two major hotspots - Almost on the roofs

Really DRAM bound
Nothing to do
Why no vectorization?

Compiler does not use SIMD

But we can make a hint…
Still not enough

We want this

Only ≈ 6% speedup

No vectorization issues

But what about DRAM?
Cache usage improved

Almost 2x!

Loop blocking

Use non-temporal stores
What we achieved

About 2.5x speed improvement!

Measured on 4C/8T Intel® Core™ i7 4770K 3.5GHZ
OpenMP* parallelization (8 threads)
Without MPI
Intel® Compiler 15.0, “/O3 /Qopenmp /QxCORE-AVX2”
Can we do better?

- We have 27 reads + 1 write per grid element
- One read and one write have to go to DRAM
- DRAM AI = CARM AI * 28 / 2 = CARM AI * 14 = 0.1205 * 14 = 1.687
- Now we are bound by DRAM, but probably can get ≈ (38.76/28.13-1) = 38% (for this loop, not entire app)

- Using DRAM-based arithmetic intensity – yet to be implemented
Summary

• Roofline model is a useful analysis and optimization technique
• Even more useful when integrated into the tool
  • Source, code properties etc. just a few clicks away
• We have an early build, working on improving it
  • Feedback is needed!

Follow the links
Windows: https://registrationcenter.intel.com/en/forms/?productid=2784
Linux: https://registrationcenter.intel.com/en/forms/?productid=2785
or
Contact vector_advisor@intel.com to get a copy!
Questions?
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 © 2014, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

<table>
<thead>
<tr>
<th>Optimization Notice</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

Notice revision #20110804