Performance Tuning for Intel® Xeon Phi™ Coprocessors

Robert Reed
Intel Technical Consulting Engineer
Agenda

Start tuning on host

Overview of Intel® VTune™ Amplifier XE

Efficiency metrics

Problem areas
Start with **host-based profiling to identify vectorization/parallelism/offload candidates**

Start with representative/reasonable workloads!

Use Intel® VTune™ Amplifier XE to gather hot spot data

• Tells what functions account for most of the run time
• Often, this is enough
  - But it does not tell you much about program structure

Alternately, profile functions & loops using Intel® Composer XE

• **Build with options** `-profile-functions -profile-loops=all -profile-loops-report=2`

• Run the code (which may run slower) to collect profile data

• **Look at the resulting dump files, or open the xml file with the data viewer loopprofileviewer.sh located in the compiler ./bin directory**

• **Tells you**
  - which loops and functions account for the most run time
  - how many times each loop executes (min, max and average)
Correctness/Performance Analysis of Parallel code

Intel® Inspector XE and thread-reports in Intel® VTune™ Amplifier XE are not available for the Intel® Xeon Phi™ coprocessor

So...

• Use Intel Inspector XE on your code with **offload disabled** (on host) to identify correctness errors (e.g., deadlocks, races)
  - Once fixed, then enable offload and continue debugging on the coprocessor

• Use Intel VTune Amplifier XE’s parallel performance analysis tools to find issues on the host by running your program with **offload disabled**
  - Fix everything you can
  - Then study scaling on the coprocessor using lessons from host tuning to further optimize parallel performance
    - Be wary of synchronization when the number of threads becomes more than a handful
    - Also pay attention to load balance.
Agenda

Start tuning on host

Overview of Intel® VTune™ Amplifier XE

Efficiency metrics

Problem areas
Intel® VTune™ Amplifier XE offers a rich GUI

- Menu and Tool bars
- Analysis Type
- Viewpoint currently being used
- Tabs within each result
- Grid area
- Current grouping
- Stack Pane
- Timeline area
- Filter area
Intel® VTune™ Amplifier XE on Intel® Xeon Phi™ coprocessors

Adjust Data Grouping
- Function - Call Stack
- Module - Function - Call Stack
- Source File - Function - Call Stack
- Thread - Function - Call Stack

No Call Stacks Yet

Double Click Function to View Source

Filter by Timeline Selection (or by Grid Selection)

Filter by Module & Other Controls

Filter In by Selection

Remove All Filters

No Call Stacks Yet

Double Click Function to View Source

Filter by Timeline Selection (or by Grid Selection)

Filter by Module & Other Controls

Select row(s):

No filters are applied.
Intel® VTune™ Amplifier XE displays event data at function, source & assembly levels

Quick Asm navigation: Select source to highlight Asm

Quickly scroll to hot spots. Scroll Bar “Heat Map” is an overview of hot spots

Time on Source / Asm

Right click for instruction reference manual

Click jump to scroll Asm
Collect events in Intel® VTune™ Amplifier XE on offload and native program executions

Application settings:

- **Application:** `ssh`
- **Parameters:** `mic0 "<app startup>"`
- **Working directory:** Usually does not matter
- Don’t forget to set search directories under “All files” to map source
Coprocessor collections have their own analysis types.
Performance Analysis

Intel® VTune™ Amplifier XE only collects “Lightweight Hotspots” for Intel® Xeon Phi™ coprocessors

• Uses Event-Based Sampling -
• Uses the Performance Monitoring Unit
• No instrumentation (“Locks & Waits”, “Concurrency”, etc.)
• More to come!

Other analysis types need to be configured

• Use “Lightweight Hotspots” and create a copy of it
• Add the desired counters
• Add some useful name and documentation
• Multi-event collections (over 2) can multiplex or use multiple runs
Configuring a User-defined Analysis

NOTE: For analysis purposes, Intel VTune Amplifier XE 2013 may adjust the Sample After values in the table by a multiplier. The multiplier depends on the value of the Duration time estimate option specified in the Properties dialog.

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Sample After</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU_CLK_UNHALTED</td>
<td>10000000</td>
<td>Number of cycles during which the processor is not halted</td>
</tr>
<tr>
<td>INSTRUCTIONS_EXECUTED</td>
<td>10000000</td>
<td>Number of instructions executed (up to two per clock)</td>
</tr>
</tbody>
</table>

Available events:
- BANK_CONFLICTS
- BRANCHES
- BRANCHES_MISPREDICTED
- CODE_CACHE_MISS
- CODE_READ
- DATA_CACHE_LINES_WRITTEN_BACK
- DATA_PAGE_WALK
- DATA_READ
- DATA_READ_MISS
- DATA_READ_MISS_OR_WRITE_MISS
- DATA_READ_OR_WRITE
- DATA_WRITE
- DATA_WRITE_MISS
- EXEC_STAGE_CYCLES

List of MIC cards (e.g. 0.1.2.3): 0

Command line name: shiny-new-analysis

Analysis identifier: shiny-new-analysis
Collecting Hardware Performance Data

Hardware counters and events
• 2 counters in core, most are thread specific
• 4 outside the core (uncore) that get no thread or core details
• See PMU documentation for a full list of events

Collection
• Invoke from Intel® VTune™ Amplifier XE
• If collecting more than 2 core events, select multi-run for more precise results or the default multiplexed collection, all in one run
• Uncore events are limited to 4 at a time in a single run
• Uncore event sampling needs a source of PMU interrupts, e.g. programming cores to CPU_CLK_UNHALTED

Output files
• Intel VTune Amplifier XE performance database
Agenda

Start tuning on host

Overview of Intel® VTune™ Amplifier XE

Efficiency metrics

Problem areas
Efficiency Metric: CPI

- Cycles per Instruction
- Can be calculated per hardware thread or per core
- Is a ratio! So varies widely across applications and should be used carefully.

<table>
<thead>
<tr>
<th>Number of Hardware Threads / Core</th>
<th>Minimum (Best) Theoretical CPI per Core</th>
<th>Minimum (Best) Theoretical CPI per Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>1.5</td>
</tr>
<tr>
<td>4</td>
<td>0.5</td>
<td>2.0</td>
</tr>
</tbody>
</table>
Efficiency Metric: CPI

- Measures how latency affects your application’s execution

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
<th>Investigate if</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPI Per Thread</td>
<td>CPU_CLK_UNHALTED/INSTRUCTIONS_EXECUTED</td>
<td>&gt; 4.0, or increasing</td>
</tr>
<tr>
<td>CPI Per Core</td>
<td>(CPI per Thread) / Number of hardware threads used</td>
<td>&gt; 1.0, or increasing</td>
</tr>
</tbody>
</table>

- Look at how optimizations applied to your application affect CPI
- Address high CPIs through any optimizations that aim to reduce latency
Some CPI Data

- Scaling data from a lab workload:

<table>
<thead>
<tr>
<th>Metric</th>
<th>1 hardware thread / core</th>
<th>2 hardware threads / core</th>
<th>3 hardware threads / core</th>
<th>4 hardware threads / core</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPI per Thread</td>
<td>5.24</td>
<td>8.80</td>
<td>11.18</td>
<td>13.74</td>
</tr>
<tr>
<td>CPI per Core</td>
<td>5.24</td>
<td>4.40</td>
<td>3.73</td>
<td>3.43</td>
</tr>
</tbody>
</table>

- Scatter plot of observed CPIs from lab workloads:
Efficiency Metric: Compute to Data Access Ratio

- Measures an application’s computational density, and its suitability for Intel® MIC Architecture

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
<th>Investigate if</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 Compute to Data Access Ratio</td>
<td>VPU_ELEMENTS_ACTIVE / DATA_READ_OR_WRITE</td>
<td>&lt; Vectorization Intensity</td>
</tr>
<tr>
<td>L2 Compute to Data Access Ratio</td>
<td>VPU_ELEMENTS_ACTIVE / DATA_READ_MISS_OR_WRITE_MISS</td>
<td>&lt; 100x L1 Compute to Data Access Ratio</td>
</tr>
</tbody>
</table>

- Increase computational density through vectorization and reducing data access (see cache issues, also, DATA ALIGNMENT!)
Agenda

Start tuning on host

Overview of Intel® VTune™ Amplifier XE

Efficiency metrics

Problem areas*

*tuning suggestions requiring deeper understanding of architectural tradeoffs and application data handling details are highlighted with this “ninja” notation
Problem Area: L1 Cache Usage

- Significantly affects data access latency and therefore application performance

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
<th>Investigate if</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 Misses</td>
<td>DATA_READ_MISS_OR_WRITE_MISS + L1_DATA_HIT_INFLIGHT_PF1</td>
<td></td>
</tr>
<tr>
<td>L1 Hit Rate</td>
<td>(DATA_READ_OR_WRITE - L1 Misses) / DATA_READ_OR_WRITE</td>
<td>&lt; 95%</td>
</tr>
</tbody>
</table>

- Tuning Suggestions:
  - Software prefetching
  - Tile/block data access for cache size
  - Use streaming stores

  If using 4K access stride, may be experiencing conflict misses

  Examine Compiler prefetching (Compiler-generated L1 prefetches should not miss)
Problem Area: Data Access Latency

- Significantly affects application performance

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
<th>Investigate if</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated Latency Impact</td>
<td>(CPU_CLK_UNHALTED - EXEC_STAGE_CYCLES - DATA_READ_OR_WRITE) / DATA_READ_OR_WRITE_MISS</td>
<td>&gt;145</td>
</tr>
</tbody>
</table>

- Tuning Suggestions:
  - Software prefetching
  - Tile/block data access for cache size
  - Use streaming stores

Check cache locality – turn off prefetching and use CACHE_FILL events - reduce sharing if needed/possible

If using 64K access stride, may be experiencing conflict misses
Problem Area: TLB Usage

- Also affects data access latency and therefore application performance

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
<th>Investigate if</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 TLB miss ratio</td>
<td>DATA_PAGE_WALK/DATA_READ_OR_WRITE</td>
<td>&gt; 1%</td>
</tr>
<tr>
<td>L2 TLB miss ratio</td>
<td>LONG_DATA_PAGE_WALK / DATA_READ_OR_WRITE</td>
<td>&gt; .1%</td>
</tr>
<tr>
<td>L1 TLB misses per L2 TLB miss</td>
<td>DATA_PAGE_WALK / LONG_DATA_PAGE_WALK</td>
<td>&gt; 100x</td>
</tr>
</tbody>
</table>

- Tuning Suggestions:
  - Improve cache usage & data access latency
  - If L1 TLB miss/L2 TLB miss is high, try using large pages
  - For loops with multiple streams, try splitting into multiple loops
  - If data access stride is a large power of 2, consider padding between arrays by one 4 KB page
Problem Area: VPU Usage

• Indicates whether an application is vectorized successfully and efficiently

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
<th>Investigate if</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vectorization Intensity</td>
<td>VPU_ELEMENTS_ACTIVE / VPU_INSTRUCTIONS_EXECUTED</td>
<td>&lt;8 (DP), &lt;16 (SP)</td>
</tr>
</tbody>
</table>

• Tuning Suggestions:
  - Use the Compiler vectorization report!
  - For data dependencies preventing vectorization, try using Intel® Cilk™ Plus #pragma SIMD (if safe!)
  - Align data and tell the Compiler!
  - Re-structure code if possible: Array notations, AOS->SOA
Problem Area: Memory Bandwidth

- Can increase data latency in the system or become a performance bottleneck

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
<th>Investigate if</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Bandwidth</td>
<td>(UNC_F_CH0_NORMAL_READ + UNC_F_CH0_NORMAL_WRITE + UNC_F_CH1_NORMAL_READ + UNC_F_CH1_NORMAL_WRITE) X 64/time</td>
<td>&lt; 80GB/sec (practical peak 140GB/sec) (with 8 memory controllers)</td>
</tr>
</tbody>
</table>

- Tuning Suggestions:
  - Improve locality in caches
  - Use streaming stores
  - Improve software prefetching
Event collections on the coprocessor can generate volumes of data
dgemm: on 60+ cores

Tip: Use cpu-mask to reduce data set, while maintaining the same accuracy.
Summary

• Vectorization, Parallelism, and Data locality are critical to good performance for the Intel® Xeon Phi™ Coprocessor


• Intel® VTune™ Amplifier XE supports collecting all of the above metrics, as well as providing special analysis types like Memory Bandwidth
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, Xeon Phi, 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