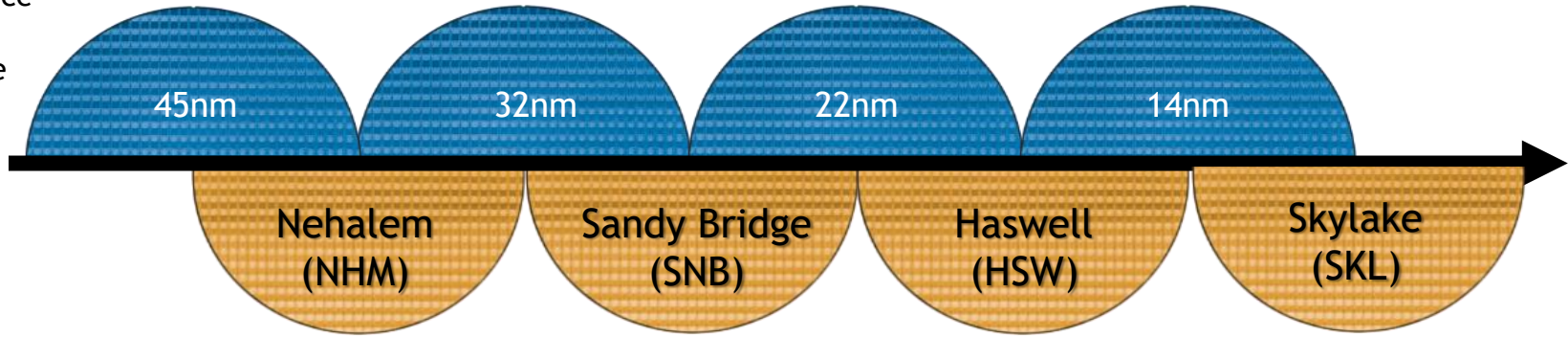
Intel® Software Tools for KNL

甘驰
Developer Products Division/SSG

IA Cores build on a Common Architecture

Scalable Performance
Energy Efficient
Microarchitecture



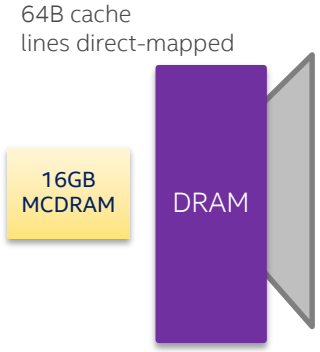
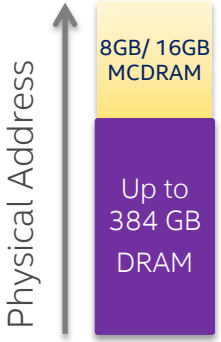
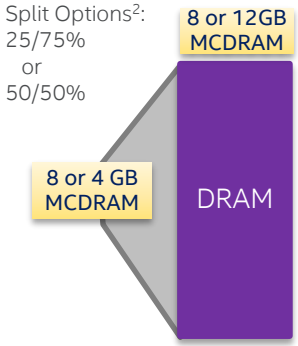
Highly Parallel
Energy Efficient
Architecture

All dates, product descriptions, availability, and plans are forecasts and subject to change without notice.

Integrated On-Package Memory Usage Models

Model configurable at boot time and software exposed through NUMA[1]

Platform Memory (DDR4) only available for bootable KNL host processor

	Cache Model	Flat Model	Hybrid Model
			
Description	Hardware automatically manages the MCDRAM as a "L3 cache" between CPU and ext DDR memory	Manually manage how the app uses the integrated on-package memory and external DDR for peak perf	Harness the benefits of both Cache and Flat models by segmenting the integrated on-package memory
Usage Model	▪ App and/or data set is very large and will not fit into MCDRAM ▪ Unknown or unstructured memory access behavior	▪ App or portion of an app or data set that can be, or is needed to be "locked" into MCDRAM so it doesn't get flushed out	▪ Need to "lock" in a relatively small portion of an app or data set via the Flat model ▪ Remaining MCDRAM can then be configured as Cache

1. NUMA = non-uniform memory access
2. As projected based on early product definition

Intel® Parallel Studio XE – Components

	Composer Edition	Professional Edition	Cluster Edition
Intel® C++ compiler	☑	☑	☑
Intel® Fortran compiler	☑	☑	☑
Intel® Math Kernel Library	☑	☑	☑
Intel® Threading Building Blocks library	☑	☑	☑
Intel® Integrated Performance Primitives	☑	☑	☑
Intel® Cilk™ Plus parallel model	☑	☑	☑
OpenMP*	☑	☑	☑
Intel® Advisor XE		☑	☑
Intel® Inspector XE		☑	☑
Intel® VTune™ Amplifier XE		☑	☑
Intel® Data Analytic Acceleration Library (Intel® DAAL)	☑	☑	☑
Intel® MPI library			☑
Intel® Trace Analyzer and Collector			☑
Rogue Wave IMSL* Library	**Bundled & Add-on**	**Add-on**	**Add-on**

Generate Code for KNL with Intel Compiler 17.0

Linux* OS:

-xMIC-AVX512

Windows* OS:

/QxMIC-AVX512

May generate Intel® Advanced Vector Extensions 512 (Intel® AVX-512) Foundation instructions, Intel® AVX-512 Conflict Detection instructions, Intel® AVX-512 Exponential and Reciprocal instructions, Intel® AVX-512 Prefetch instructions for Intel® processors, and the instructions enabled with CORE-AVX2. Optimizes for Intel® processors that support Intel® AVX-512 instructions.

Libraries Optimized for KNL

- Intel® Math Kernel Library 2017
  - Optimized math functions to enable neural networks (CNN and DNN) for deep learning
- Intel® Distribution for Python
  - NumPy/SciPy/Scikit-Learn/Pandas accelerated with Intel® MKL, Intel® MPI, Intel® TBB, Intel® DAAL
- Intel® Data Analytics Acceleration Library 2017
- Intel® MPI Library
  - Usage of specially optimized memcpy for KNL in MPI Library
  - Tuning of shared memory collectives on single KNL Nodes
- Intel® Integrated Performance Primitives (Intel® IPP)
- Intel® TBB Library

KNL Profiling Available Today

VTune™ Amplifier XE 2016 Update 2

Memory Access analysis

- Shows performance problems by memory hierarchy
- Measure DRAM and MCDRAM bandwidth
- Helps define data structures to allocate to MCDRAM

General Exploration analysis

- Efficiency of code passing through the core pipeline

Scalability analysis with Advanced Hotspots

- Serial vs Parallel time
- MPI and OpenMP imbalance, overhead cost, parallel loop parameters

Custom PMU event collection

Grouping: Bandwidth Domain / Bandwidth Utilization Type / Memory Object / Allocation Stack

Bandwidth Domain / Bandwidth Utilization Type / Memory Object / Allocation Stack	Memory Bound	Loads	Stores	LLC Miss Count	Average Latency (cycles)
DRAM, GB/sec	0.657	125,874,377,622	16,061,040 ...	130,507,830	40
High	0.750	28,236,084,708	5,014,875, ...	75,304,518	91
stream.c:180 (76 MB)		900,002,700	654,009,810	18,301,098	495
stream.c:179 (76 MB)		1,050,003,150	667,210,008	33,301,998	487
stream.c:181 (76 MB)		1,434,004,302	907,213,608	20,101,206	412
Selected 1 row(s):	1.000	126,000,378	21,600,324	300,018	61

OpenMP Analysis. Collection Time: 28.061

Serial Time (outside any parallel region): 12.203s (43.5%)

Serial Time of your application is high. It directly impacts application Elapsed Time and scalability. Explore options for parallelization, algorithm or microarchitecture tuning of the serial part of the application.

Parallel Region Time: **15.858s (56.5%)**

Estimated Ideal Time: 5.005s (17.8%)

OpenMP Potential Gain: 10.853s (38.7%)

The time wasted on load imbalance or parallel work arrangement is significant and negatively impacts the application performance and scalability. Explore OpenMP regions with the highest metric values. Make sure the workload of the regions is enough and the loop schedule is optimal.

Running VTune Amplifier from the command-line

On self-boot KNL machines ensure the amplxe-cl command is installed. See the "amplxe-cl –help" command for complete details.

To collect:

- Hotspots:
  - amplxe-cl –collect advanced-hotspots –- myapp.out
- General Exploration:
  - amplxe-cl –collect general-exploration –- myapp.out
- Memory Access:
  - amplxe-cl –collect memory-access –- myapp.out

Advanced Hotspots Analysis

Supports OpenMP* analysis

Stack-sampling is enabled. However, call counts and trip counts are not supported.

Advanced Hotspots Hotspots viewpoint (change)

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-do

Elapsed Time: 4.506s

CPU Time:	566.499s
Effective Time:	195.181s
Spin Time:	352.865s

A significant portion of CPU time is spent waiting. Use this metric to discover which synchronizations are parameters, changing the lock implementation (for example, by backing off then descheduling), or adjust

Overhead Time:	18.453s
Instructions Retired:	320,824,000,000
CPI Rate:	2.642

The CPI may be too high. This could be caused by issues such as memory stalls, instruction starvation, instructions. Explore the other hardware-related metrics to identify what is causing high CPI.

CPU Frequency Ratio:	1.071
Total Thread Count:	130
Paused Time:	0s

Total Thread Count much higher on Phi. Threading is vital for performance.

Advanced Hotspots Hotspots viewpoint (change) Intel VTune Amplifier XE 2016

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

OpenMP Analysis. Collection Time: 4.506

Serial Time (outside any parallel region): 1.107s (24.6%)

Serial Time of your application is high. It directly impacts application Elapsed Time and scalability. Explore options for parallelization, algorithm or microarchitecture tuning of the serial part of the application.

Parallel Region Time:	**3.399s (75.4%)**
Estimated Ideal Time:	1.217s (27.0%)
OpenMP Potential Gain:	2.182s (48.4%)

The time wasted on load imbalance or parallel work arrangement is significant and negatively impacts the application performance and scalability. Explore OpenMP regions with the highest metric values. Make sure the workload of the regions is enough and the loop schedule is optimal.

Top OpenMP Regions by Potential Gain

This section lists OpenMP regions with the highest potential for performance improvement. The Potential Gain metric shows the elapsed time that could be saved if the region was optimized to have no load imbalance assuming no runtime overhead.

OpenMP Region	OpenMP Potential Gain	(%)	OpenMP Region Time
mainompparallel:130@unknown:19:25	2.182s	48.4%	3.399s

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
__kmp_wait_template<kmp_flag_64>	libiomp5.so	282.763s
mainompparallel_for@19	test.out	186.721s
kmp_basic_flag<unsigned long long>::notdone_check	libiomp5.so	13.521s
__kmp_hierarchical_barrier_release	libiomp5.so	8.720s
__kmp_yield	libiomp5.so	7.838s
[Others]	N/A*	66.935s

*N/A is applied to non-summable metrics.

Advanced Hotspots Analysis

Adjust grouping to change granularity

Top Functions

See 100s of OpenMP Threads

Overhead is very important on KNL

Poor/OK/Good classification can be adjusted manually.

VTune Amplifier Tips

VTune Finalization:

- Finalization might be slow on KNL. Finalize on Xeon.
- Disable auto finalization with: -no-auto-finalize

Large amount of raw data collected:

- Appropriately select the app run duration using: -target-duration-type=<veryshort/short/medium/long>
- Change the default data limit as required.

Power throttling:

- Keep an eye on the CPU frequency ratio. If this ratio changes significantly during the run then you might be seeing throttling or turbo effects.

VTune Amplifier Tips (cont.)

Event multiplexing:

- Similar to KNC, KNL has only 2 general purpose counters. Hence, when collecting a large number of events the data might be statistically invalid.
- Try changing the target duration type or allow multiple runs.

High Bandwidth Memory Analysis on KNL

Memory Bandwidth often is a limiting factor of compute intensive applications on multi-core systems

MCDRAM – High Bandwidth Memory with much greater bandwidth speedup to alleviate this problem

Limited MCDRAM size might require selective data object placement to HBM (for flat and hybrid MCDRAM modes)

Memory Access analysis helps to identify memory objects for HBM placement to benefit the most

HBW Analysis Steps – Case Study

Use Case:

- miniFE* benchmark from Mantevo Suite

Platform:

- KNL with MCDRAM in flat mode

VTune Amplifier XE 2017 Beta with NDA package

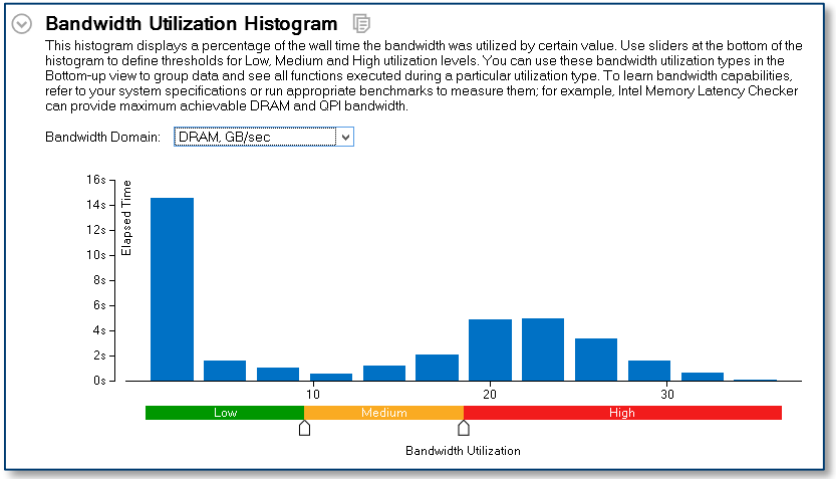
HBW Analysis Steps – Case Study. Step 1

Run the original application under Memory Access analysis

Explore DRAM Bandwidth histogram to see if the app is bandwidth bound

Significant portion of application time spent in high memory bandwidth utilization

The app may benefit from MCDRAM



HBW Analysis Steps – Case Study. Step 2

Investigate the memory allocations inducing bandwidth

- “Bandwidth Domain/Bandwidth Utilization Type/Memory Object/Allocation Stack” grouping with expansion by “DRAM/High” and sorting by L2 Miss Count

Memory Access Memory Usage viewpoint (change)

INTEL VTUNE AMPLIFIER XE 2017

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Platform | CSRMatrix.h...

package_0 23.4 15.6 7.8

DRAM Bandwidth, GB/sec

DRAM Bandwidth, ... Total, GB/sec Read, GB/sec Write, GB/sec MCDRAM Flat Mod... MCDRAM Cache M... CPU Time

L2 Miss Count (Memory Allocation)

Viewing 1 of 1 selected stack(s)

100.0% (20400612 of 20400612)

miniFE.x!__gnu_cxx::new_allocator<double>::allocate - new_allocator.h
miniFE.x!std::vector<double, std::allocator<double>>::_M_allocate_and_copy<double*> ...
miniFE.x!miniFE::CSRMatrix<double, int, int>::reserve_space+0x130 - CSRMatrix.hpp:93
miniFE.x!miniFE::generate_matrix_structure<miniFE::CSRMatrix<double, int, int>>+0x1...
miniFE.x!miniFE::driver<double, int, int>+0x186 - driver.hpp:181
miniFE.x!main+0x7f4 - main.cpp:178
miniFE.x!_start+0x28 - [unknown source file]

Grouping: Bandwidth Domain / Bandwidth Utilization Type / Memory Object / Allocation Stack

Bandwidth Domain / Bandwidth Utilization Type / Memory Object / Allocation Stack	CPU Time	L2 Miss Count
DRAM, GB/sec	4517.651s	311,009,330
High	3053.636s	259,007,770
new_allocator.h:104 (840 MB)		102,003,060
[Unknown]		71,002,130
new_allocator.h:104 (31 MB)		53,001,590
new_allocator.h:104 (420 MB)		22,000,660
[Stack]		5,000,150
new_allocator.h:104 (31 MB)		3,000,090
new_allocator.h:104 (31 MB)		2,000,060
new_allocator.h:104 (15 MB)		1,000,030
Medium	754.149s	50,001,500
Low	709.866s	2,000,060
DRAM Read, GB/sec	4517.651s	311,009,330
DRAM Write, GB/sec	4517.651s	311,009,330
Selected 1 row(s):		102,003,060

Memory Access Memory Usage viewpoint (change)

Collection Log | Analysis Target | Analysis Type | Summary | Botton

Source | Assembly | Assembly grouping:

```
So.  Source
88     void reserve_space(unsigned nrows, unsigned ncols_per_row)
89     {
90       rows.resize(nrows);
91       row_offsets.resize(nrows+1);
92       packed_cols.reserve(nrows * ncols_per_row);
93       packed_coefs.reserve(nrows * ncols_per_row);
94
95       #pragma omp parallel for
96       for(MINIFE_GLOBAL_ORDINAL i = 0; i < nrows; ++i) {
97       rows[i] = 0;
98       row_offsets[i] = 0;
99       }
```

Focus on allocations inducing L2 misses

Allocation stack shows the allocation place in user’s code

HBW Analysis Steps – Case Study. Step 3

Allocate object using High Bandwidth Memory

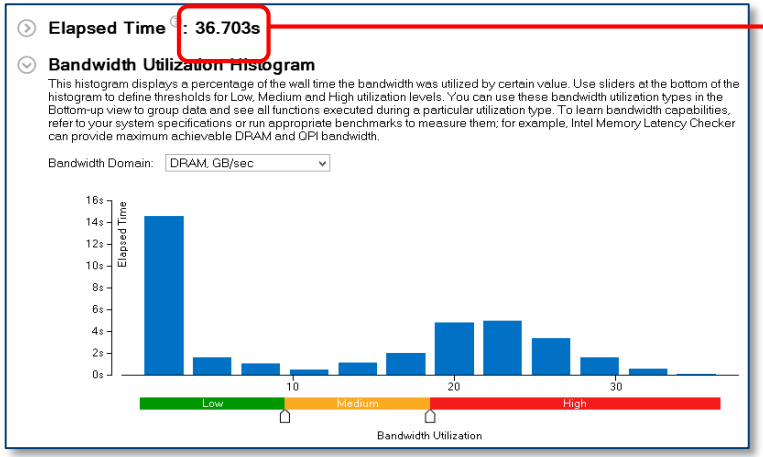
- Specifying a custom memory allocator class for stored vector elements

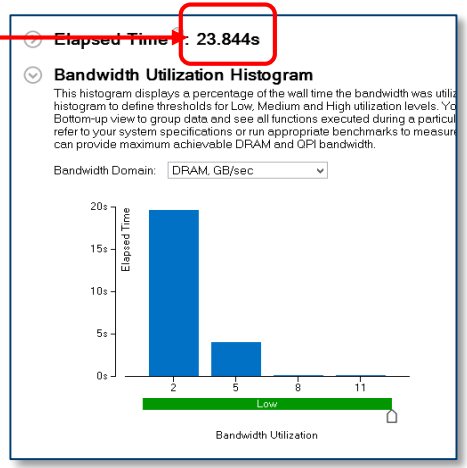
```
template<typename Scalar,
        typename LocalOrdinal,
        typename GlobalOrdinal,
        typename ComputeNode>
struct CSRMatrix {
...
 typedef Scalar        ScalarType;
 typedef LocalOrdinal  LocalOrdinalType;
 typedef GlobalOrdinal GlobalOrdinalType;
 typedef ComputeNode   ComputeNodeType;

 bool                  has_local_indices;
 std::vector<GlobalOrdinal> rows;
 std::vector<LocalOrdinal>  row_offsets;
 std::vector<LocalOrdinal>  row_offsets_external;
 std::vector<GlobalOrdinal> packed_cols;
 std::vector<Scalar>        packed_coefs;
 LocalOrdinal           num_cols;
 ComputeNode&           compute_node;

...
}
```

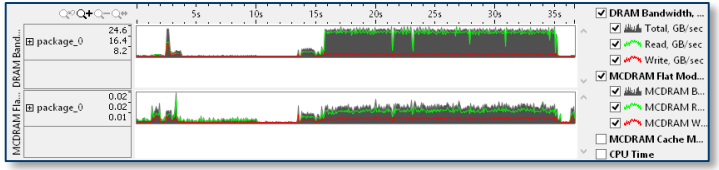
```
        typename LocalOrdinal,
        typename GlobalOrdinal,
        typename ComputeNode>
struct CSRMatrix {
...
 typedef Scalar        ScalarType;
 typedef LocalOrdinal  LocalOrdinalType;
 typedef GlobalOrdinal GlobalOrdinalType;
 typedef ComputeNode   ComputeNodeType;

 bool                  has_local_indices;
 std::vector<GlobalOrdinal, hbwmalloc::hbwmalloc_allocator<GlobalOrdinal> > rows;
 std::vector<LocalOrdinal, hbwmalloc::hbwmalloc_allocator<GlobalOrdinal> > row_offsets;
 std::vector<LocalOrdinal, hbwmalloc::hbwmalloc_allocator<GlobalOrdinal> > row_offsets_external;
 std::vector<GlobalOrdinal, hbwmalloc::hbwmalloc_allocator<GlobalOrdinal> > packed_cols;
 std::vector<Scalar, hbwmalloc::hbwmalloc_allocator<GlobalOrdinal> >        packed_coefs;
 LocalOrdinal           num_cols;
 ComputeNode&           compute_node;

...
}
```

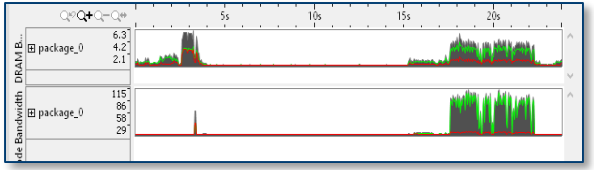
HBW Analysis Steps – Case Study. Step 4

Rerun the benchmark









DRAM bandwidth significantly decreased

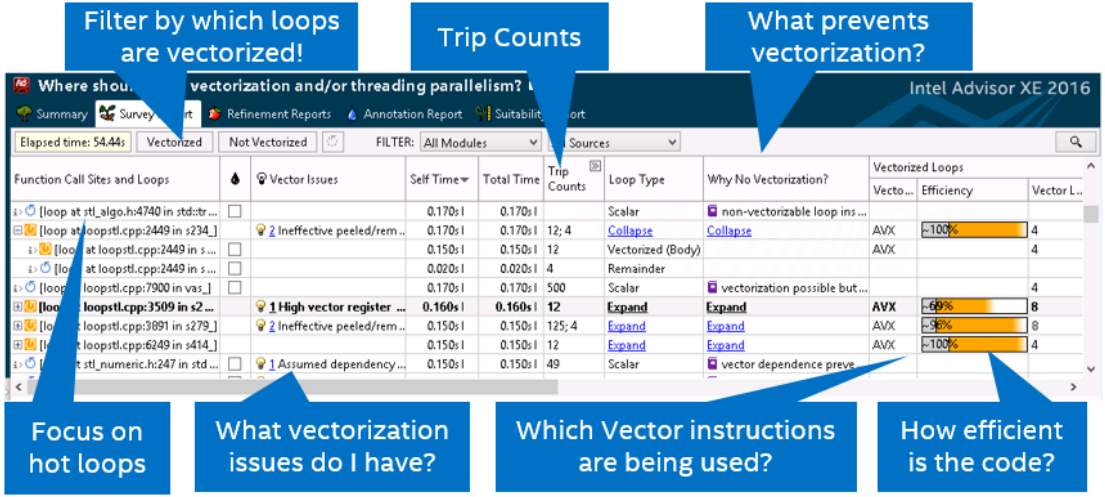
reducing DRAM memory access stalls

Easier Vectorization – Faster Code Faster

Intel® Advisor - Vectorization Advisor – New for 2016

The Right Data At Your Fingertips

- Sorts loops by potential performance gain
- Easy to read compiler reports on your source
- Vectorization tips
- Trip counts
- Dependencies
- Memory access pattern data

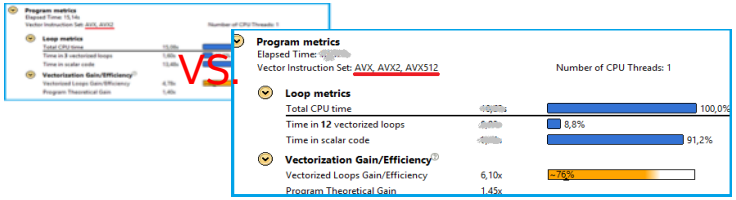


Optimize for AVX-512 Even Without Hardware

Intel® Advisor 2016

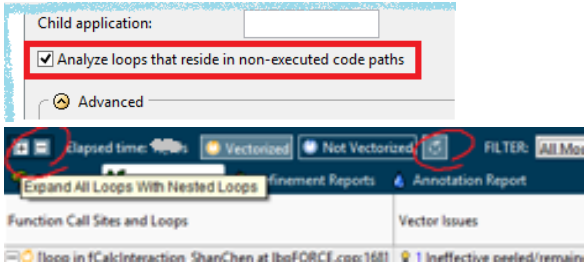
Native Advisor profiling for KNL and future platforms

- Vectorization Summary quickly compares AVX-512 vs. other architecture



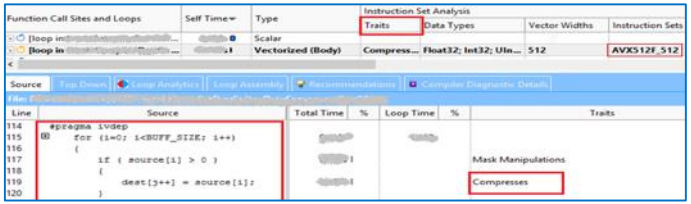
Optimize for AVX-512 even without AVX-512 hardware

- Explores AVX-512 code paths characteristics
- Create speed-up estimates for AVX512 (traits for different ISA, optimizations applied by compiler, vector length and etc.)



Survey Report provides AVX-512 Traits

- Advisor highlights AVX-512 instructions which could significantly affect vector code performance (e.g. Compress / Expand, Scatter, Conflict Detection)



Legal Disclaimer & Optimization Notice

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.

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

intel®