Faster Code.... Faster

Intel® Parallel Studio XE 2017

Dr.-Ing. Michael Klemm
Software and Services Group
michael.klemm@intel.com

Unleash the Beast...

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.

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® Parallel Studio XE

Profiling, Analysis & Architecture

Intel® Inspector
Memory & Threading Checking

Intel® VTune™ Amplifier
Performance Profiler

Intel® Advisor
Vectorization Optimization & Thread Prototyping

Cluster Tools

Intel® Cluster Checker
Cluster Diagnostic Expert System

Intel® Trace Analyzer & Collector
MPI Profiler

Intel® MPI Library

Performance Libraries

Intel® Data Analytics Acceleration Library
Optimized for Data Analytics & Machine Learning

Intel® Math Kernel Library
Optimized Routines for Science, Engineering & Financial

Intel® Integrated Performance Primitives
Image, Signal & Compression Routines

Intel® Threading Building Blocks
Task Based Parallel C++ Template Library

Intel® C/C++ & Fortran Compilers

Intel® Distribution for Python
Performance Scripting - Coming Soon – Q3'16

INTEL® COMPILERS

Intel® Compilers for Parallel Studio XE 2017

What's new in Intel® C++ 17.0 and Intel® Fortran 17.0

Productive language-level vectorization & parallelism models for advanced developers driving application performance

Common updates

- Enhanced support for the newest AVX2 and AVX512 instruction sets for the latest Intel® processors (including Intel® Xeon Phi)
- Enhanced optimization/vectorization reports – register allocation
- Tight integration with Intel® Advisor
- Initial support for OpenMP* 4.5, offering improved vectorization control, new SIMD instructions, and much more

Intel® C++ Compiler

- SIMD Data Layout Template to facilitate vectorization for your C++ code
- Virtual function vectorization capability
- Improved compiler loop and function alignment
- Full support for the latest C11 and C++14 standards

Intel® Fortran Compiler

- Substantial coarray performance improvement – up to **twice as fast** as previous versions on non-trivial coarray Fortran programs
- Almost complete Fortran 2008 support
- Further interoperability with C (part of draft Fortran 2015)

Impressive Performance Improvement

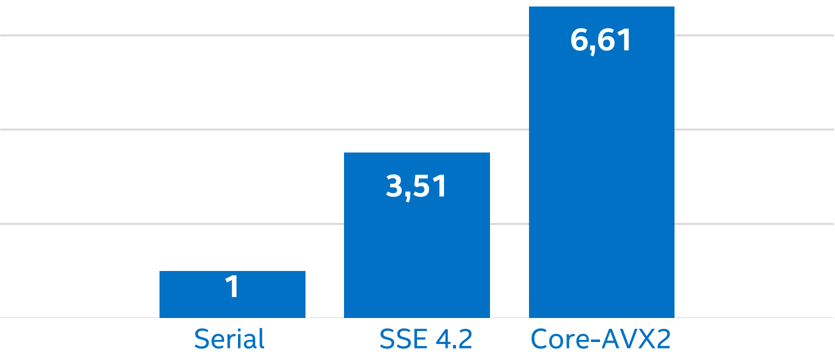
Intel® Compiler OpenMP* Explicit Vectorization

- Three lines added that take full advantage of both SSE or AVX
- Pragma's ignored by other compilers so code is portable

```
#pragma omp declare simd linear(z:40) uniform(L, N, Nmat) linear(k)
float path_calc(float *z, float L[][VLEN], int k, int N, int Nmat)

#pragma omp declare simd uniform(L, N, Nopt, Nmat) linear(k)
float portfolio(float L[][VLEN], int k, int N, int Nopt, int Nmat)
… … …
for (path=0; path<NPATH; path+=VLEN) {
   /* Initialise forward rates */
   z = z0 + path * Nmat;
#pragma omp simd linear(z:Nmat)
   for(int k=0; k < VLEN; k++) {
     for(i=0;i<N;i++) {
        L[i][k] = L0[i];
     }

     /* LIBOR path calculation */
     float temp = path_calc(z, L, k, N, Nmat);
     v[k+path]  = portfolio(L, k, N, Nopt, Nmat);

     /* move pointer to start of next block */
     z += Nmat;
   }
 }
```

Libor calculation speedup

Normalized performance data – higher is better



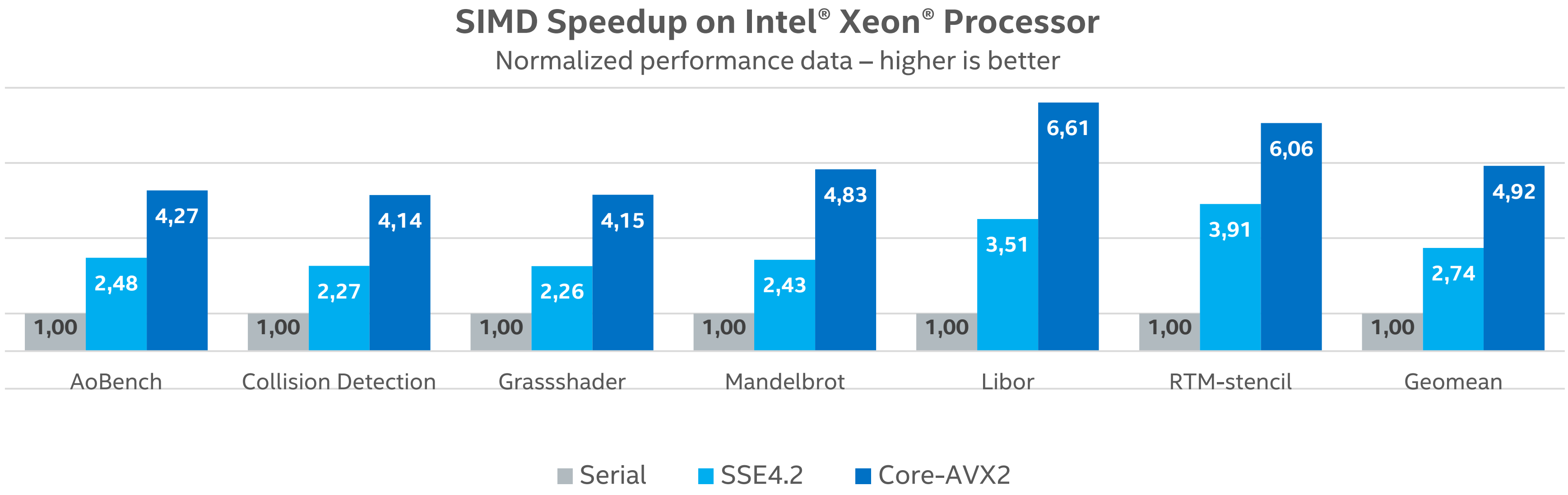
Configuration: Intel® Xeon® CPU E3-1270 @ 3.50 GHz Haswell system (4 cores with Hyper-Threading On), running at 3.50GHz, with 32.0GB RAM, L1 Cache 256KB, L2 Cache 1.0MB, L3 Cache 8.0MB, 64-bit Windows* Server 2012 R2 Datacenter. Compiler options:, SSE4.2: –O3 –Qopenmp -simd –QxSSE4.2 or AVX2: -O3 –Qopenmp –simd -QxCORE-AVX2. For more information go to http://www.intel.com/performance

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. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation

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.

Impressive performance improvement

Intel C++ Explicit Vectorization using OpenMP* SIMD



Configuration: Intel® Xeon® CPU E3-1270 @ 3.50 GHz Haswell system (4 cores with Hyper-Threading On), running at 3.50GHz, with 32.0GB RAM, L1 Cache 256KB, L2 Cache 1.0MB, L3 Cache 8.0MB, 64-bit Windows* Server 2012 R2 Datacenter. Compiler options:, SSE4.2: –O3 –Qopenmp -simd –QxSSE4.2 or AVX2: -O3 –Qopenmp –simd -QxCORE-AVX2. For more information go to http://www.intel.com/performance

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. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation

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 SOFTWARE ANALYSIS TOOLS

Intel® VTune™ Amplifier XE Performance Profiler

Intel® Advisor XE Vectorization Optimization and Thread Prototyping

INTEL® VTUNE™ AMPLIFIER XE
PERFORMANCE PROFILER

Intel® VTune™ Amplifier

Faster, Scaleable Code, Faster

Get the Data You Need

- Hotspot (Statistical call tree), Call counts (Statistical)
- Thread Profiling – Concurrency and Lock & Waits Analysis
- Cache miss, Bandwidth analysis…[1]
- GPU Offload and OpenCL™ Kernel Tracing

Find Answers Fast

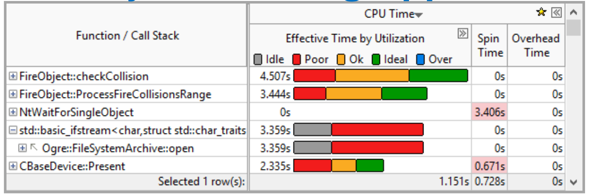
- View Results on the Source / Assembly
- OpenMP Scalability Analysis, Graphical Frame Analysis
- Filter Out Extraneous Data – Organize Data with Viewpoints
- Visualize Thread & Task Activity on the Timeline

Easy to Use

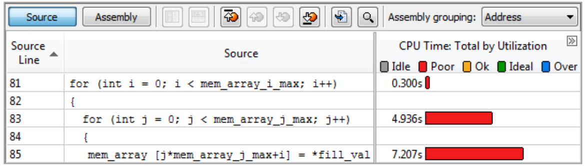
- No Special Compiles – C, C++, C#, Fortran, Java, ASM
- Visual Studio* Integration or Stand Alone
- Graphical Interface & Command Line
- Local & Remote Data Collection
- Analyze Windows* & Linux* data on OS X*[2]

[1] Events vary by processor. [2] No data collection on OS X*

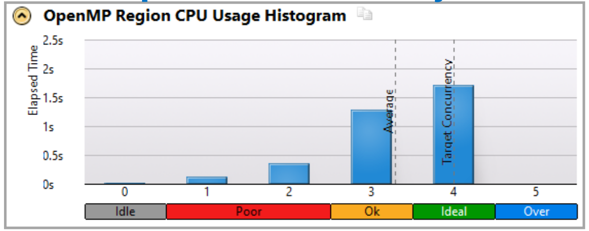
Quickly Find Tuning Opportunities



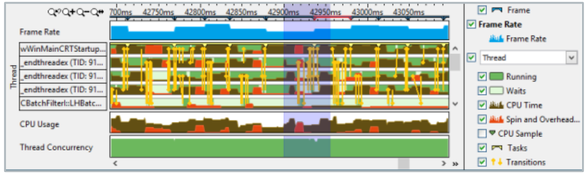
See Results On The Source Code



Tune OpenMP Scalability



Visualize & Filter Data



New!

Profile Python & Go!

And Mixed Python / C++ / Fortran

Low Overhead Sampling

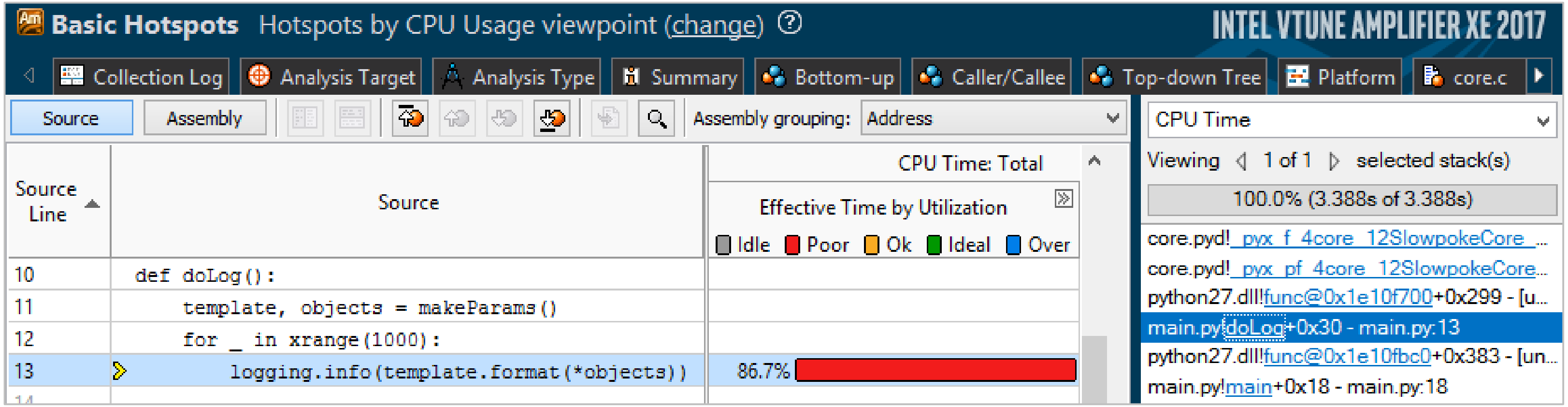
- Accurate performance data without high overhead instrumentation
- Launch application or attach to a running process

Precise Line Level Details

- No guessing, see source line level detail

Mixed Python / native C, C++, Fortran…

- Optimize native code driven by Python



Three Keys to HPC Performance:

Threading, Memory Access, Vectorization – Intel VTune™ Amplifier

New!

Threading: CPU Utilization

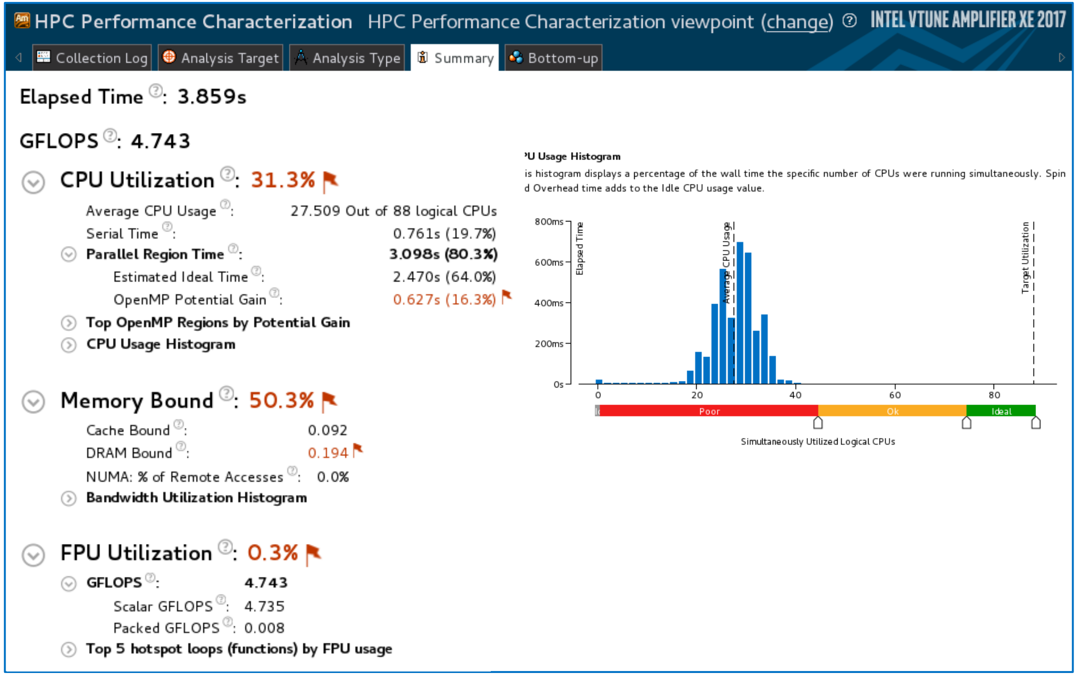
- Serial vs. Parallel time
- Top OpenMP regions by potential gain
- Tip: Use hotspot OpenMP region analysis for more detail

Memory Access Efficiency

- Stalls by memory hierarchy
- Bandwidth utilization
- Tip: Use Memory Access analysis

Vectorization: FPU Utilization

- FLOPS† estimates from sampling
- Tip: Use Intel Advisor for precise metrics and vectorization optimization



† For 3rd, 5th, 6th Generation Intel® Core™ processors and second generation Intel® Xeon Phi™ processor code named Knights Landing.

Optimization Notice

Improved!

Optimize Memory Access

Memory Access Analysis - Intel® VTune™ Amplifier 2017

Tune data structures for performance

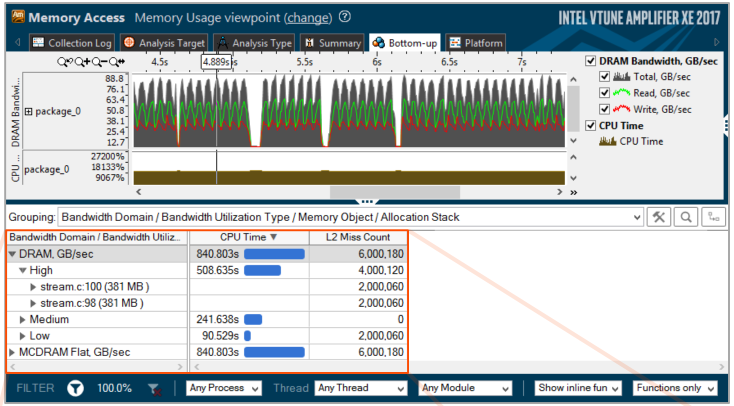
- Attribute cache misses to data structures (not just the code causing the miss)
- Support for custom memory allocators

Optimize NUMA latency & scalability

- True & false sharing optimization
- Auto detect max system bandwidth
- Easier tuning of inter-socket bandwidth

Easier install, latest processors

- No special drivers required on Linux*
- Intel® Xeon Phi™ processor MCDRAM (high bandwidth memory) analysis



Bandwidth Domain / Bandwidth Utiliz...	CPU Time ▼	L2 Miss Count
▼ DRAM, GB/sec	840.803s	6,000,180
▼ High	508.635s	4,000,120
▶ stream.c:100 (381 MB)		2,000,060
▶ stream.c:98 (381 MB)		2,000,060
▶ Medium	241.638s	0
▶ Low	90.529s	2,000,060
▶ MCDRAM Flat, GB/sec	840.803s	6,000,180

Storage Device Analysis (HDD, SATA or NVMe SSD)

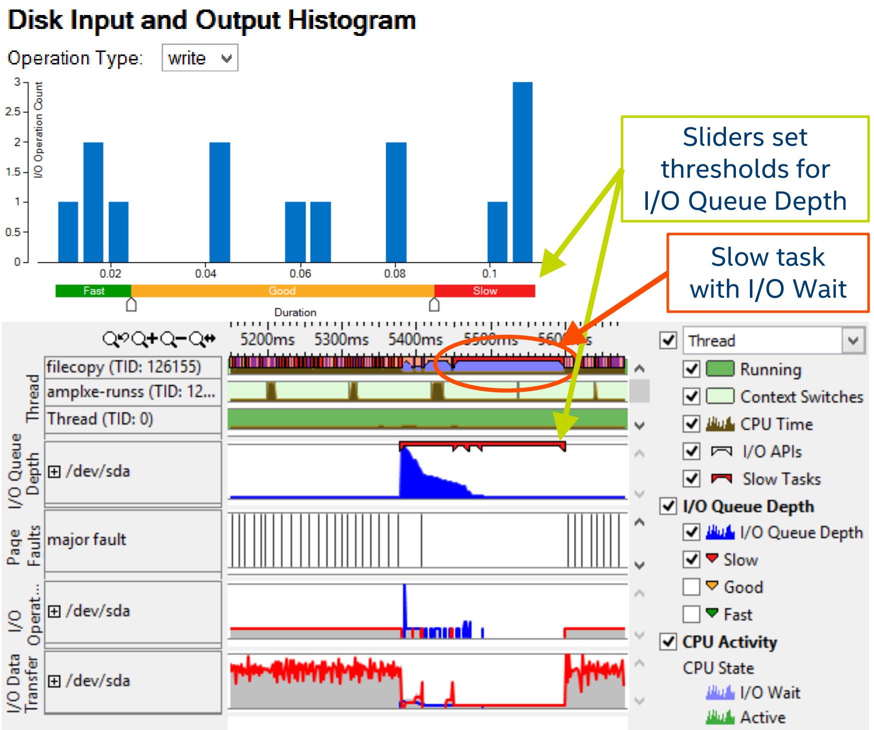
Intel® VTune™ Amplifier

Are You I/O Bound or CPU Bound?

- Explore imbalance between I/O operations (async & sync) and compute
- Storage accesses mapped to the source code
- See when CPU is waiting for I/O
- Measure bus bandwidth to storage

Latency analysis

- Tune storage accesses with latency histogram
- Distribution of I/O over multiple devices



Intel® Performance Snapshots

Three Fast Ways to Discover Untapped Performance

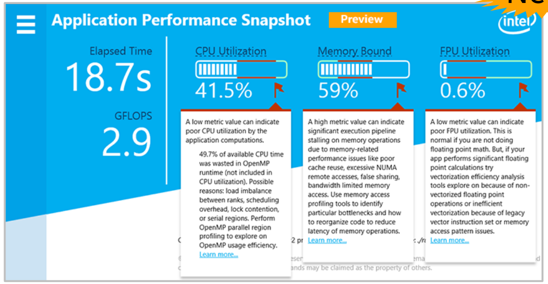
Is your application making good use of modern computer hardware?

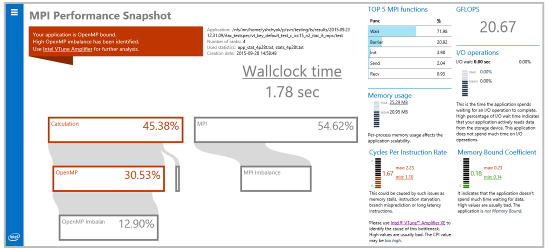
- Run a test case during your coffee break.
- High level summary shows which apps can benefit most from code modernization and faster storage.

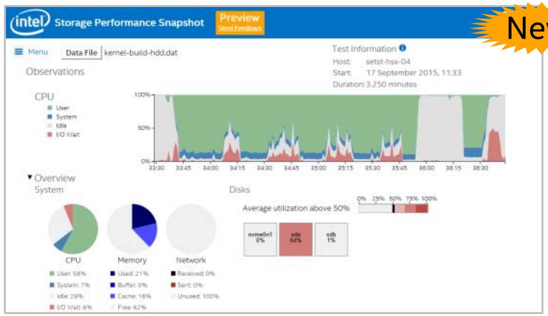
Pick a Performance Snapshot:

- **Application** – for non-MPI apps
- **MPI** – for MPI apps
- **Storage** – for systems. Servers and workstations with directly attached storage.

Free download: http://www.intel.com/performance-snapshot
Also included with Intel® Parallel Studio and Intel® VTune™ Amplifier products.







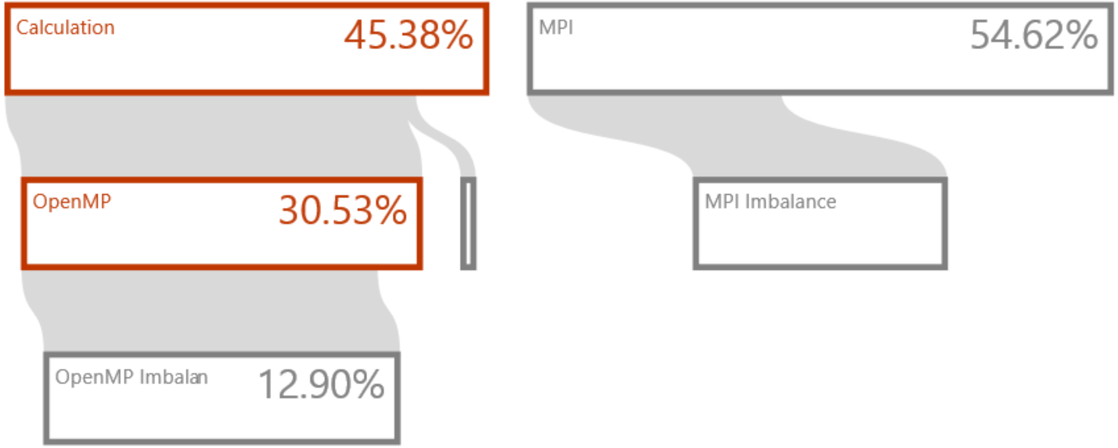
MPI Performance Snapshot

Your application is OpenMP bound.
High OpenMP imbalance has been identified.
Use Intel VTune Amplifier for further analysis.

Application: /nfs/inn/home/yshchyok/p/svn/testing/ts/results/2015.09.23 12.31.09/itac_testspec/vt_key_default_test_c_icc15_n2_itac_it_mps/test
Number of ranks: 4
Used statistics: app_stat_4p28t.txt, stats_4p28t.txt
Creation date: 2015-09-28 14:58:48

Wallclock time

1.78 sec



TOP 5 MPI functions

Func	%
Wait	71.98
Barrier	20.92
Init	3.98
Send	2.04
Recv	0.93

GFLOPS

20.67

I/O operations

I/O wait: **0.00 sec** 0.00%

PEAK 0.00%
MEAN 0.00%

This is the time the application spends waiting for an I/O operation to complete. High percentage of I/O wait time indicates that your application actively reads data from the storage device. This application does not spend much time on I/O operations.

Memory usage

PEAK 25.29 MB
MEAN 20.95 MB

Per-process memory usage affects the application scalability.

Cycles Per Instruction Rate

1.67 max: 2.23 min: 1.10

This could be caused by such issues as memory stalls, instruction starvation, branch misprediction or long latency instructions.

Please use Intel® VTune™ Amplifier XE to identify the cause of this bottleneck. High values are usually bad. The CPI value may be *too high*.

Memory Bound Coefficient

0.18 max: 0.23 min: 0.14

It indicates that the application doesn't spend much time waiting for data. High values are usually bad. The application is *not Memory Bound*.

Free download: http://www.intel.com/performance-snapshot. Also included with Intel® Parallel Studio Cluster Edition.

INTEL® ADVISOR XE

VECTORIZATION OPTIMIZATION AND THREAD PROTOTYPING FOR SOFTWARE ARCHITECTS

Get Faster Code Faster! Intel® Advisor

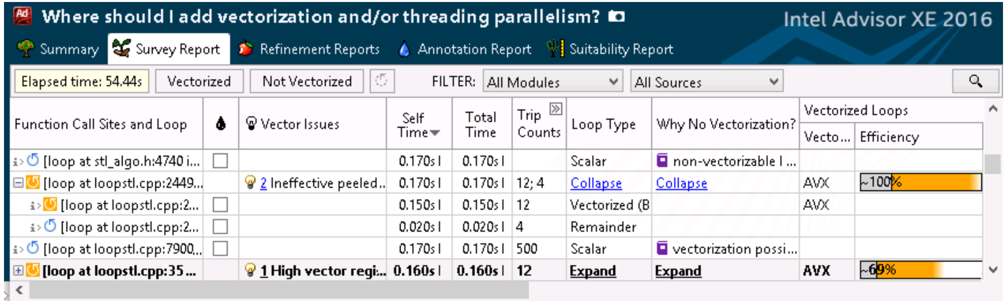
Vectorization Optimization

Have you:

- Recompiled for AVX2 with little gain
- Wondered where to vectorize?
- Recoded intrinsics for new arch.?
- Struggled with compiler reports?

Data Driven Vectorization: New!

- What vectorization will pay off most?
- What's blocking vectorization? Why?
- Are my loops vector friendly?
- Will reorganizing data increase performance?
- Is it safe to just use pragma simd?



"**Intel® Advisor's** Vectorization Advisor permitted me to focus my work where it really mattered. When you have only a limited amount of time to spend on optimization, it is invaluable."

Gilles Civario
Senior Software Architect
Irish Centre for High-End Computing

Faster Code Faster with Data Driven Design

Intel® Advisor – Vectorization Optimization and Thread Prototyping

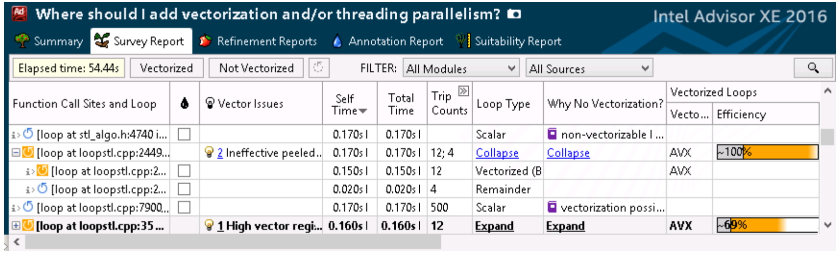
Faster Vectorization Optimization:

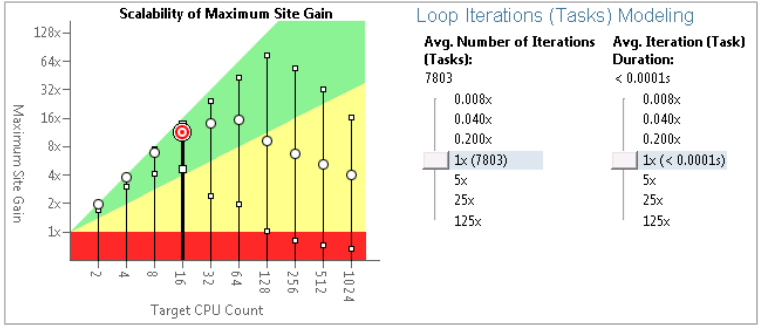
- Vectorize where it will pay off most
- Quickly ID what is blocking vectorization
- Tips for effective vectorization
- Safely force compiler vectorization
- Optimize memory stride

Breakthrough for Threading Design:

- Quickly prototype multiple options
- Project scaling on larger systems
- Find synchronization errors before implementing threading
- Design without disrupting development

Less Effort, Less Risk and More Impact





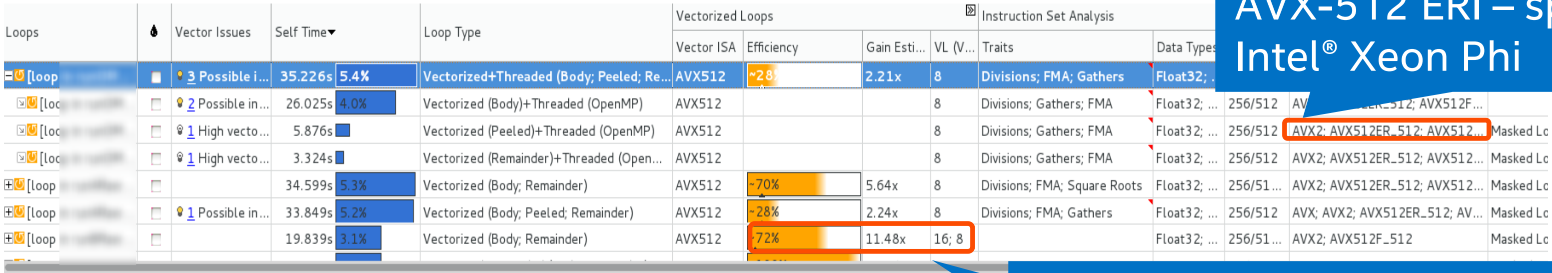
Part of Intel® Parallel Studio for Windows* and Linux*

http://intel.ly/advisor-xe

New!

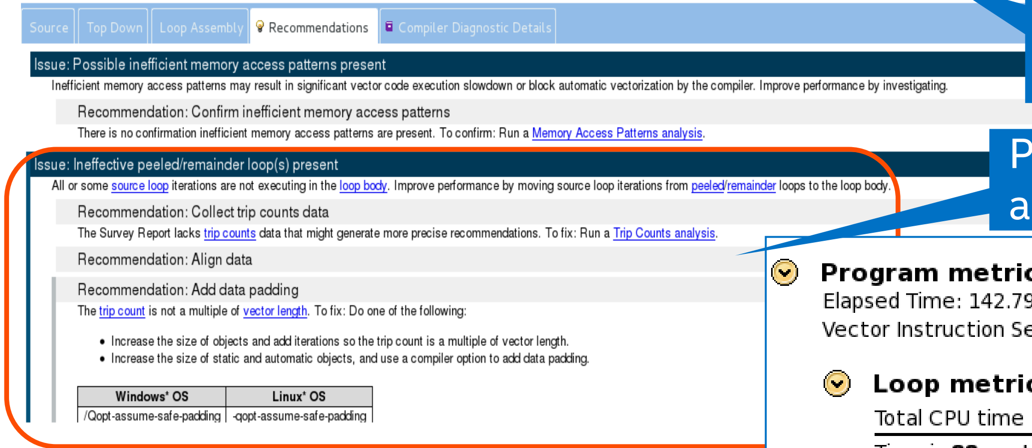
Next Gen Intel® Xeon Phi™ Support

Vectorization Advisor runs on and optimizes for Intel® Xeon Phi

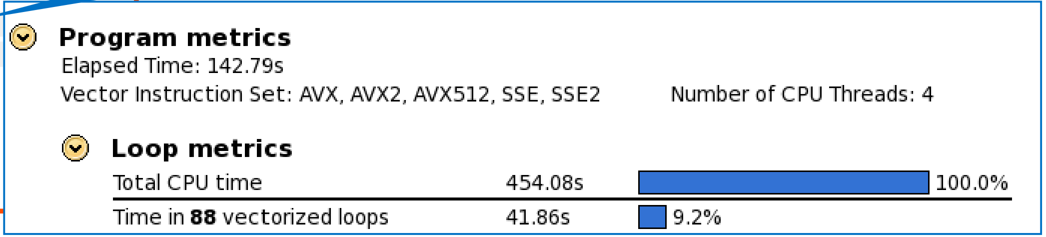


AVX-512 ERI – specific to Intel® Xeon Phi

Efficiency (72%), Speed-up (11.5x), Vector Length (16)



Performance optimization problem and advice how to fix it



New!

Start Tuning for AVX-512 without AVX-512 hardware

Intel® Advisor - Vectorization Advisor

Use –axCOMMON-AVX512 –xAVX compiler flags to generate both code-paths

- AVX(2) code path (executed on Haswell and earlier processors)
- AVX-512 code path for newer hardware

Compare AVX and AVX-512 code with Intel Advisor

Loops	🌢	Self Time	Loop Type	Vectorized Loops: Vect...	Efficiency	Gain...	VL (...	Compiler Es...	Instruction Set Analysis: Traits	Data T...	Vector W...	Instruction Sets	Advanced: Vectorization De
[loop in s352_ at loopstl.cpp:5939]		**0,641s**	**Vectorized (Body)**	**AVX2**	~54%	**2,15x**	**4**	**2,15x**	**FMA; Inserts**	**Float32**	**128**	**AVX; FMA**	
[loop in s352_ at loopstl.cpp:5939]		n/a	Remainder [Not Executed]				4		FMA				
[loop in s352_ at loopstl.cpp:5939]		0,641s	Vectorized (Body)	AVX2			4	2,15x	Inserts; FMA				
[loop in s352_ at loopstl.cpp:5939]		n/a	Vectorized (Body) [Not Executed]	AVX512			16	3,20x	Gathers; FMA				
[loop in s352_ at loopstl.cpp:5939]		n/a	Vectorized (Remainder) [Not Executed]	AVX512			16	2,70x	Gathers; FMA				
[loop in s125_.Aompparallel_for@ ...		0,496s	Vectorized Versions	AVX2	~100%	13,54x	8	<13,54x	FMA; NT-stores				
[loop in s125_.Aompparallel_for...		n/a	Peeled [Not Executed]				8		FMA				
[loop in s125_.Aompparallel_for...		n/a	Remainder [Not Executed]				8		FMA				
[loop in s125_.Aompparallel_for...		0,465s	Vectorized (Body)	AVX2			8	13,54x	[illegible]				
[loop in s125_.Zompparallel_for...		n/a	Vectorized (Peeled) [Not Executed]	AVX512			16	6,77x	FMA				
[loop in s125_.Zompparallel_for...		n/a	Vectorized (Body) [Not Executed]	AVX512			32	30,61x	[illegible]				
[loop in s125_.Zompparallel_for...		n/a	Vectorized (Remainder) [Not Executed]	AVX512			16	9,78x	FMA				

Inserts (AVX2) vs. Gathers (AVX-512)

Speed-up estimate: 13.5x (AVX2) vs. 30.6x (AVX-512)

New!

Precise Repeatable FLOPS Metrics

Intel® Advisor – Vectorization Optimization

- FLOPS by loop and function
- All recent Intel processors (not co-processors)
- Instrumentation (count FLOP) plus sampling (time with low overhead)
- Adjusted for masking with AVX-512 processors

INTEL ADVISOR 2017

Function Call Sites and Loops	FLOPS						
	GFLOPS	AI	L1 GB/s	GFLOP	FLOP Per Iteration	L1 GB	L1 Bytes Per Iteration
[loop in matvec at Multiply.c:69]	0.826	0.1633	5.0586	3.0720	32	18.8160	196
[loop in matvec at Multiply.c:60]	0.912	0.1633	5.5853	3.0720	32	18.8160	196
[loop in matvec at Multiply.c:69]	1.248	0.2500	4.9920	1.3440	4	5.3760	16
[loop in matvec at Multiply.c:60]	1.592	0.2500	6.3699	1.3440	4	5.3760	16
[loop in matvec at Multiply.c:69]	3.055	0.2500	12.2205	0.0960	16	0.3840	64
[loop in matvec at Multiply.c:60]	6.282	0.2500	25.1279	0.0960	16	0.3840	64

New!

Enhanced Memory Access Analysis

Are you bandwidth or compute limited?

Measure Footprint

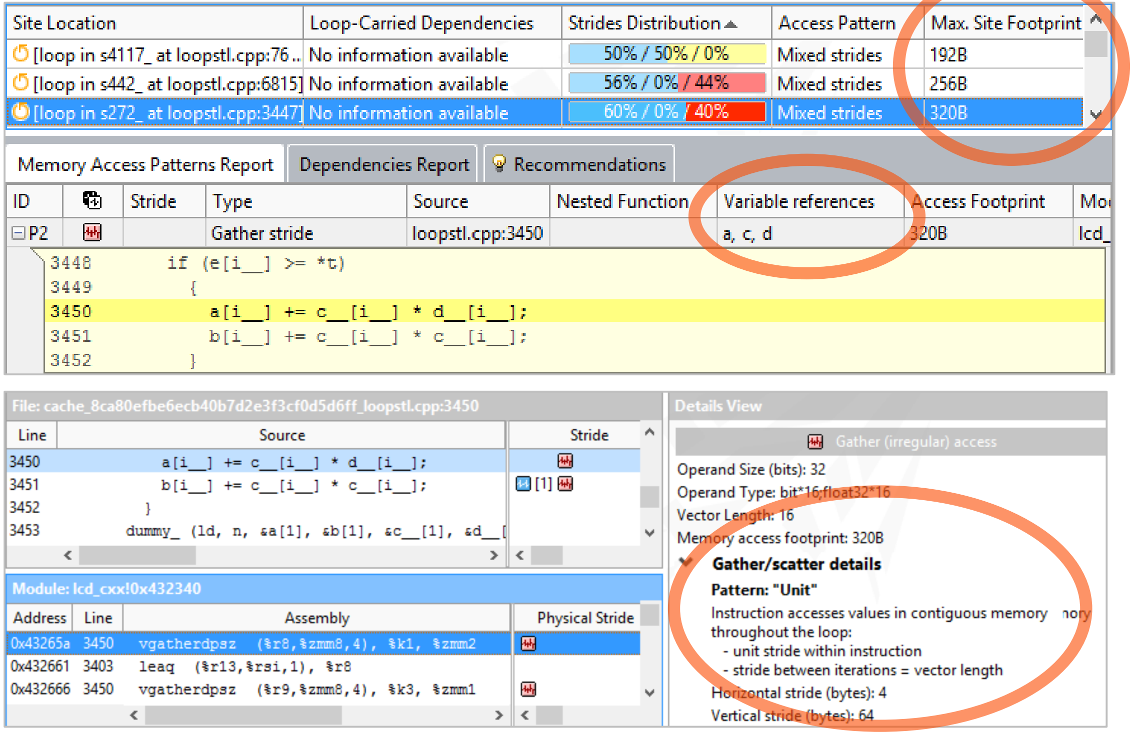
- Compare to cache size
  Does it fit in cache?

Variable References

- Map data to variable names for easier analysis

Gather/Scatter

- Detect unneeded gather/scatters that reduce performance

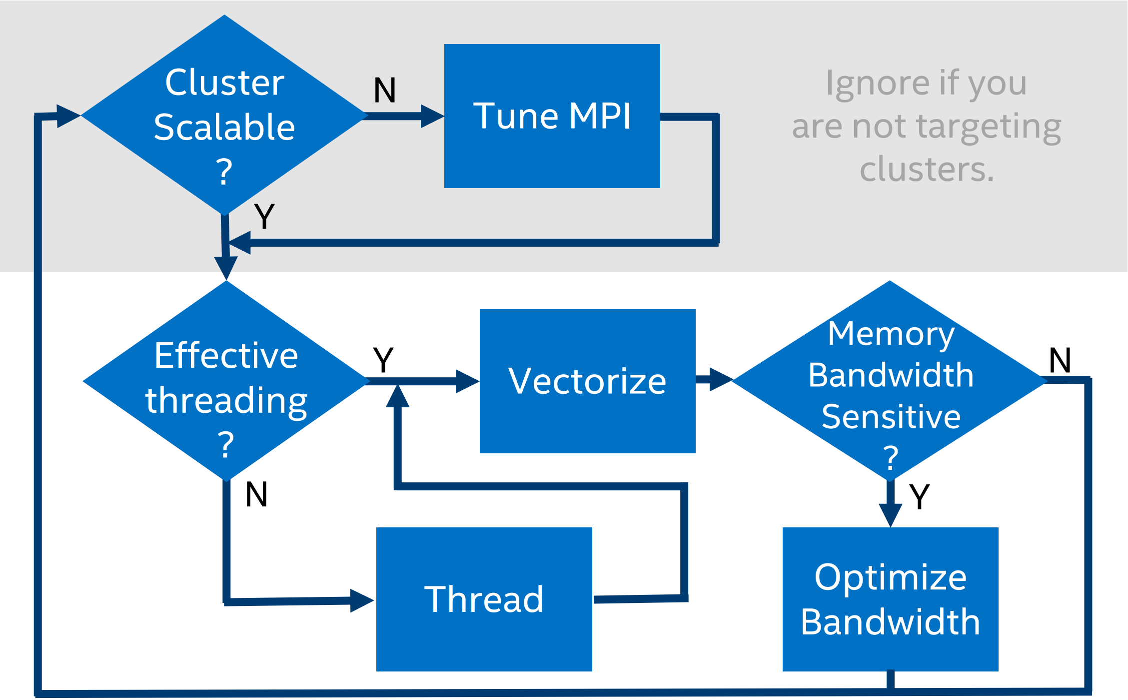


Speaker – the speaker notes are important for this presentation. Be sure to read them.

WHICH TOOL SHOULD I BE USING?

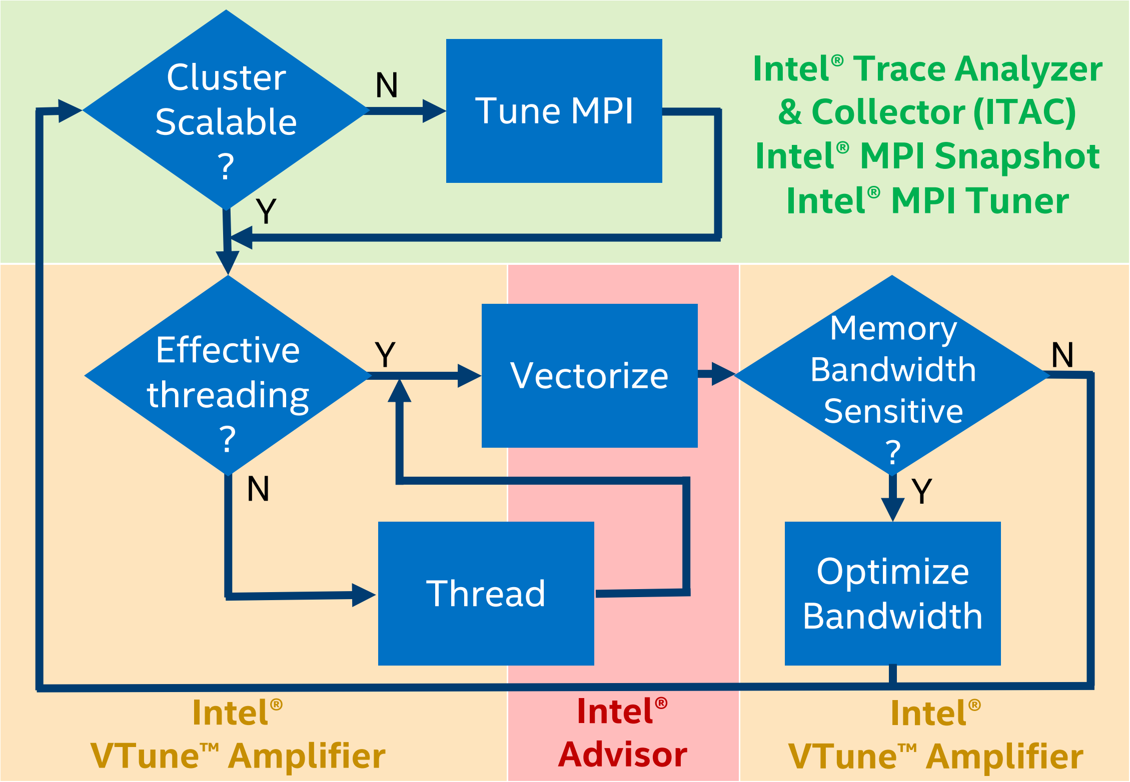
Optimizing Performance On Parallel Hardware

It's an iterative process…



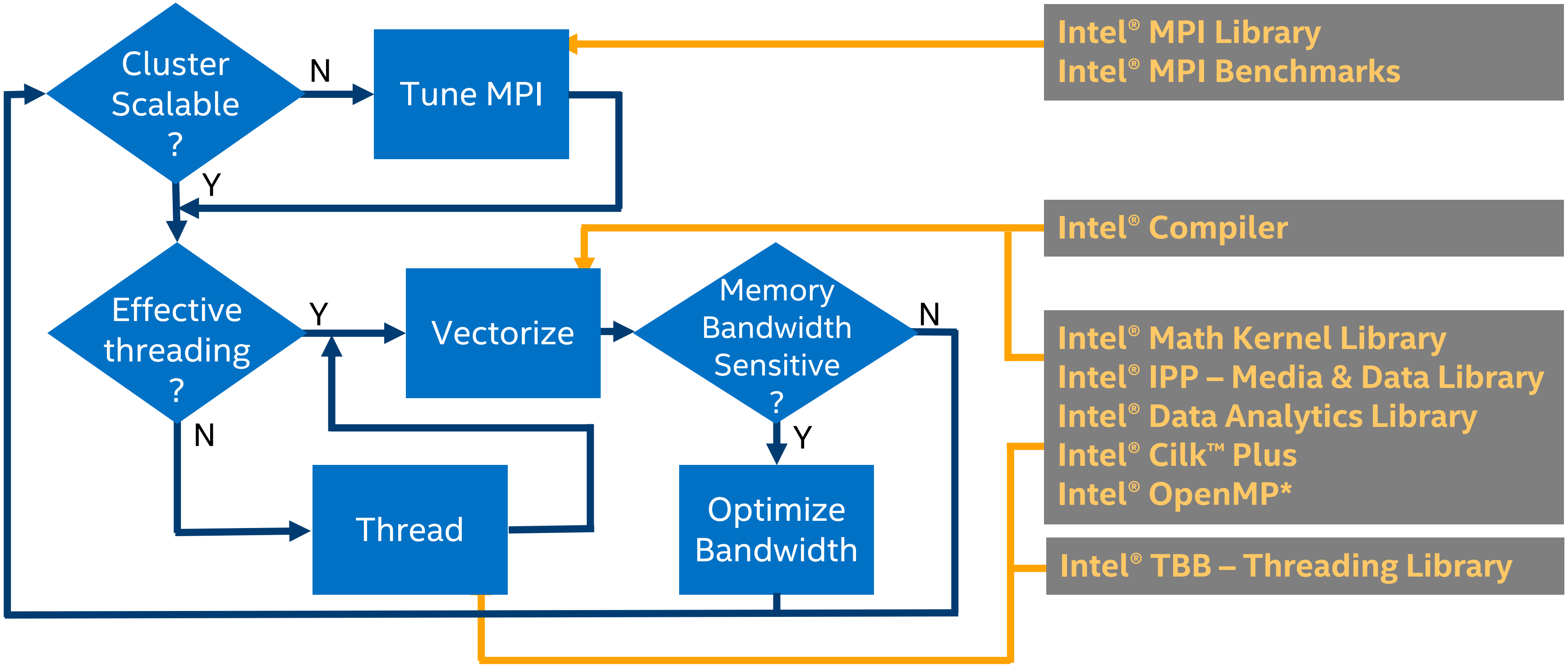
Performance Analysis Tools for Diagnosis

Intel® Parallel Studio XE



Tools for High Performance Implementation

Intel® Parallel Studio XE



intel®