



**INTEL[®]
EXPERIENCE
DAY**

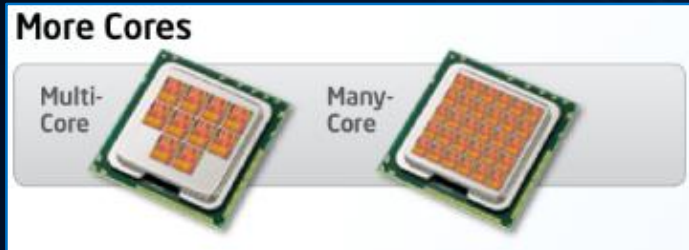


КАК INTEL[®] ADVISOR ПОМОЖЕТ ИСПОЛЬЗОВАТЬ ПЛАТФОРМЫ INTEL[®] НА 100%

Andrey Golovin, 29-Oct-2019, Moscow

INTEL ADVISOR: INSIGHTS FOR ACCELERATION

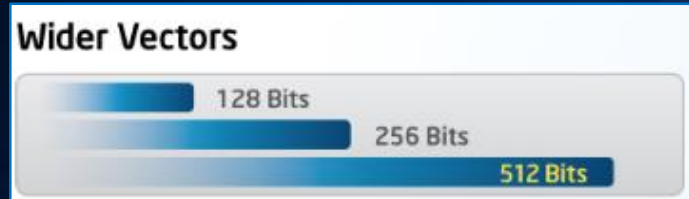
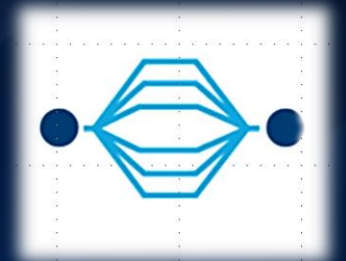
Part of Intel Parallel Studio. Responses to the hardware challenges:



Intel multi-core



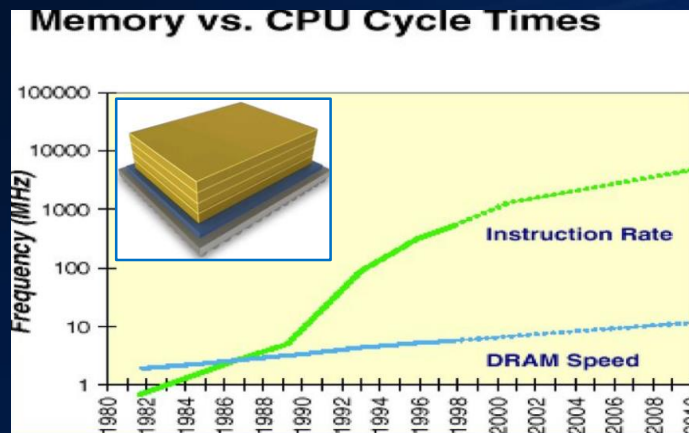
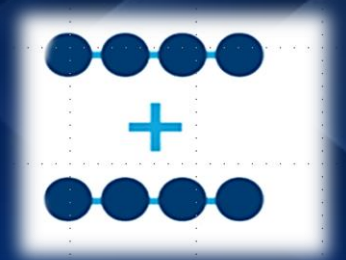
Threading Workflow



Intel SIMD AVX512 enabling



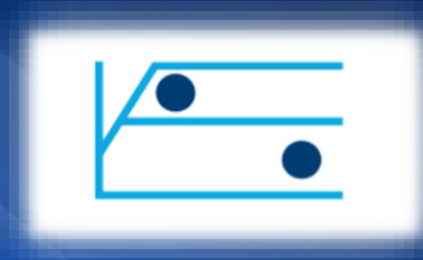
Vector Workflow



Memory hierarchies complexity



Roofline Workflow

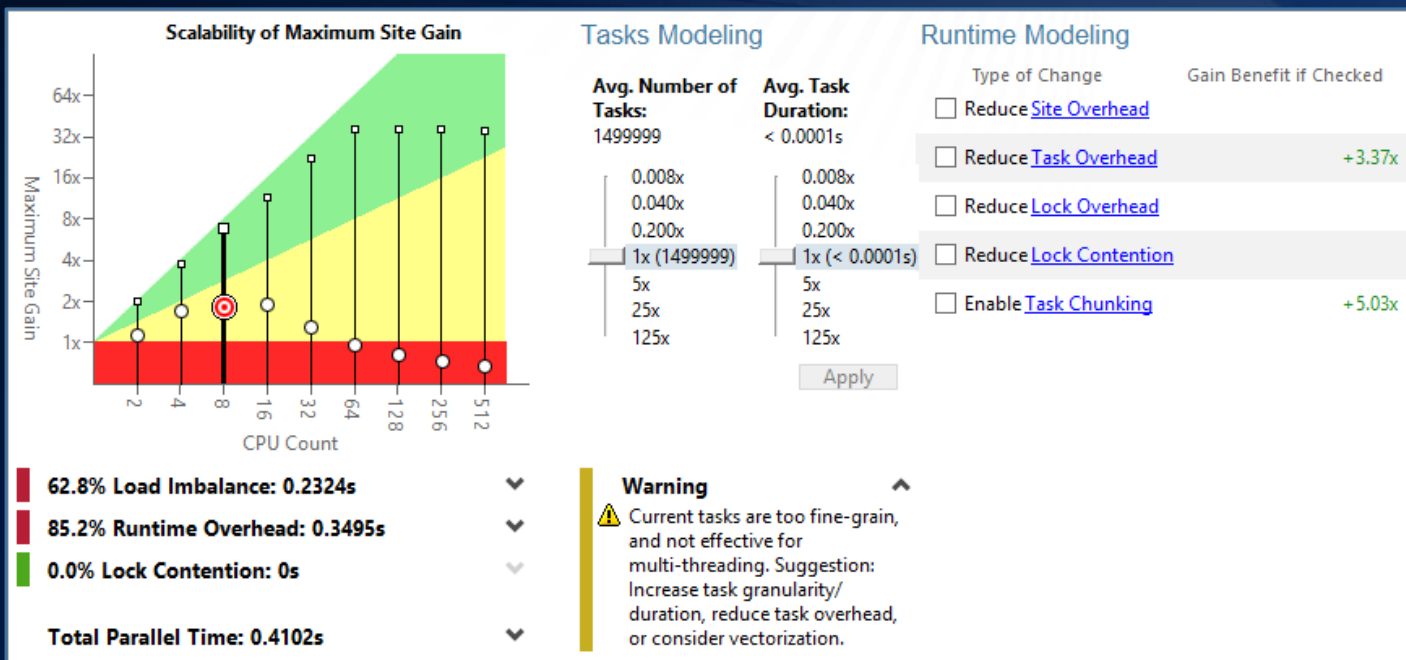
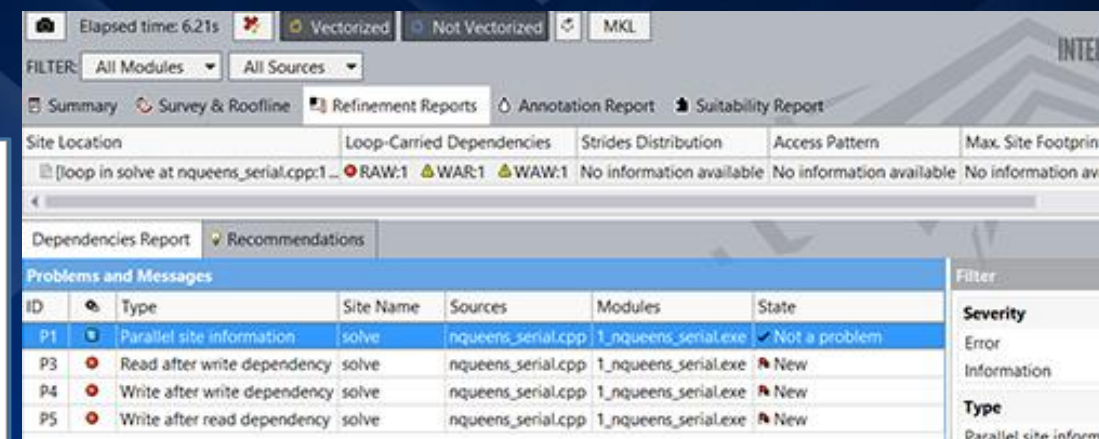
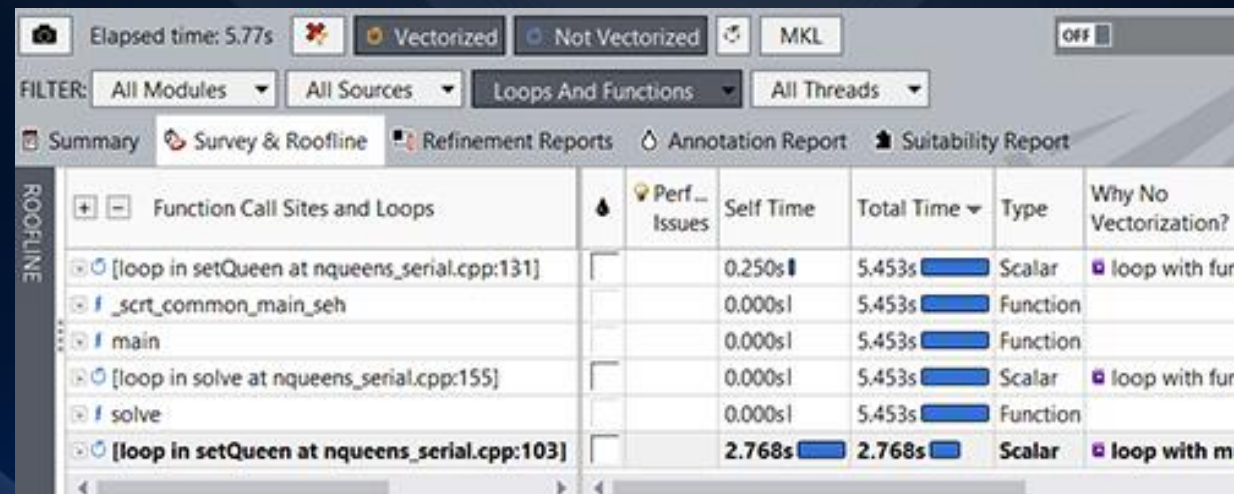


More acceleration is coming...

INTEL ADVISOR: THREADING WORKFLOW

Threading Advisor – multi-core performance and correctness design

- Scalability, task scheduler and imbalance modeling
- Data Dependencies “what-if” analysis

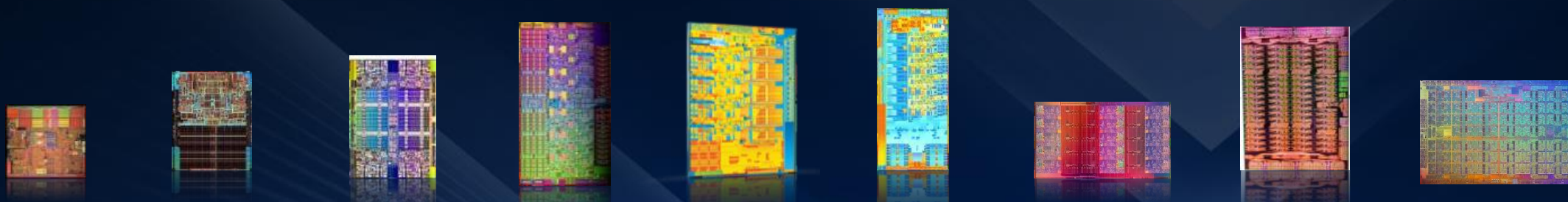


Sample Annotations

```
// Locking
ANNOTATE_LOCK_ACQUIRE();
Body();
ANNOTATE_LOCK_RELEASE();

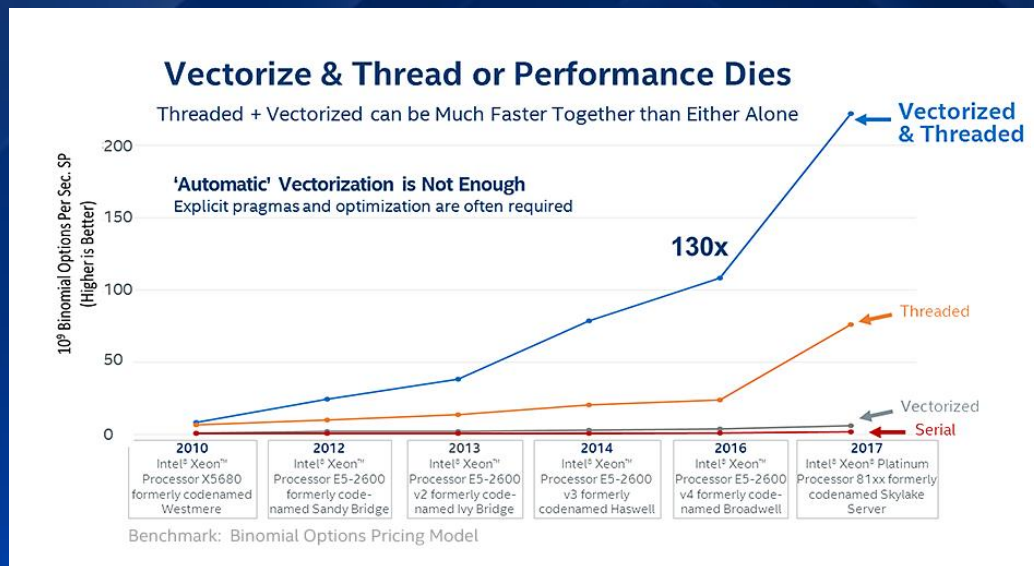
// Do-All Counted loops, one task
ANNOTATE_SITE_BEGIN(site);
For (I = 0; I < N; ++){
    ANNOTATE_ITERATION_TASK(task);
    {statement;}
}
```

INTEL ADVISOR: VECTORIZATION WORKFLOW



	Intel® Xeon® processor 64-bit	Intel® Xeon® processor 5100 series	Intel® Xeon® processor 5500 series	Intel® Xeon® processor 5600 series	Intel® Xeon® processor code-named Sandy Bridge EP	Intel® Xeon® processor code-named Ivy Bridge EP	Intel® Xeon® processor code-named Skylake EP	Intel® Xeon Phi™ coprocessor Knights Corner	Intel® Xeon Phi™ processor & coprocessor Knights Landing ¹
SIMD Width	128	128	128	128	256	256	256	512	512

**THREADING + VECTORIZATION
UP TO 130X FASTER ON BENCHMARK**



INTEL ADVISOR: VECTORIZATION WORKFLOW

Vectorization Advisor – AVX* SIMD analysis tool

- Integration with Intel Compiler perf model, SIMD / Mask register profiling
- Explicit programming model advise: “how do I fix it?”

The screenshot shows the Intel Advisor 2019 interface. At the top, it displays 'Elapsed time: 501.19s' and 'Vectorized' status. Below this is a table of 'Vectorized Loops' with columns for Vector, Efficiency, Gain, VL, Self GFLOPS, FP Mask, and Traits. A warning banner at the bottom states: 'Ineffective peeled/remainder loop(s) present. All or some source loop iterations are not executing in the loop body. Improve performance by moving source loop iterations from peeled/remainder loops to the loop body.' There are also buttons for 'Align data' and 'Reorder loops'.

All Advisor-detectable issues: [C++](#) | [Fortran](#)

Recommendation: Vectorize user function(s) inside loop

These user-defined function(s) are not vectorized or inlined by the compiler:

- apollo::common::math::linesegment2d::distanceto()
- apollo::common::math::linesegment2d::length()
- apollo::common::math::linesegment2d::projectontounit()
- apollo::common::math::vec2d::distanceto()
- apollo::common::math::vec2d::innerprod()
- apollo::common::math::vec2d::operator-()

To fix: Do one of the following:

- Enforce vectorization of the **source loop** by means of SIMD instructions and/or create a **target** function definition or declaration.

Target	Directive
Source loop	#pragma simd or #pragma omp simd
Inner function definition or declaration	#pragma omp declare simd

- If using the `Ob` or `inline-level` compiler option to control inline expansion within compiler discretion.

Example

```
#pragma omp declare simd
int f (int x)
{
    return x+1;
}
```

Site Location	Loop-Carried Dependencies	Strides Distribution	Access Pattern	Site Name
[loop in fPropagationSwap at lbpSUB.cpp:1247]	No information available	33% / 5% / 62%	Mixed strides	loop_site_60

Legend: blue color: fraction of unit stride accesses; yellow: "fixed" stride accesses ratio; red color: fraction of irregular (variable stride) accesses

Memory Access Patterns Report

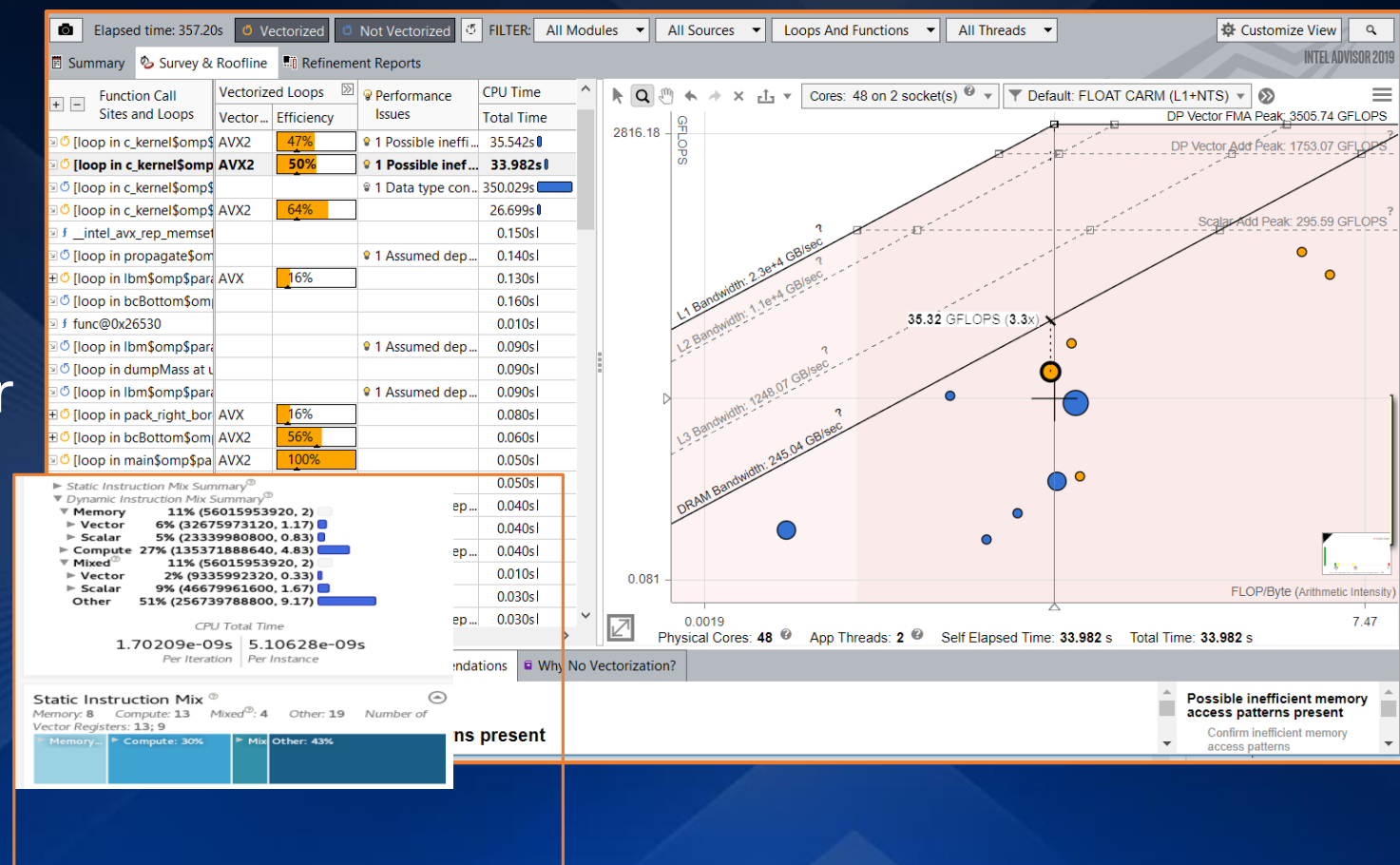
ID	Stride	Type	Source	Site Name	Variable
P1	3	Constant stride	lbpSUB.cpp:1248	loop_site_60	
P11	0; 1	Unit stride	lbpSUB.cpp:1253	loop_site_60	lbf,lbsy
P12	-289559; -274359; -14477; -13717; -13679; 723; 302519; 303279	Variable stride	lbpSUB.cpp:1253	loop_site_60	

```
1246 #endif
1247     for (int m=1; m<=half; m++) {
1248         nextx = fCppMod(i + lbv[3*m], Xmax);
1249         nexty = fCppMod(j + lbv[3*m+1], Ymax);
1250         nextz = fCppMod(k + lbv[3*m+2], Zmax);
1251         ilnext = (nextx * Ymax + nexty) * Zmax + nextz;
1252 #ifndef SWAP_OVERLAP
1253         fSwapPair (lbf[il*lbsitelength + 1*lbsy.nq + m + half], lbf[ilnext*lbsitelength + 1*lbsy.nq + m + half]);

```

INTEL ADVISOR: ROOFLINE AND MEMORY WORKFLOW

- Deep memory hierarchies characterization and simulation
- Visually intuitive modeling
- Highlights poor performing loops
- Shows performance 'headroom' for each loop
- Shows likely causes of bottlenecks
- Suggests next optimization steps



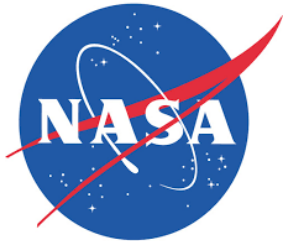
Roofline model proposed by Berkeley Lab, by Williams, Waterman, Patterson:
<http://www.eecs.berkeley.edu/~waterman/papers/roofline.pdf>
Cache-aware Roofline model: Upgrading the loft”
(Ilic, Pratas, Sousa, INESC-ID/IST, Thec Uni of Lisbon)
<http://www.inesc-id.pt/ficheiros/publicacoes/9068.pdf>

INTEL ADVISOR: CUSTOMER STORIES

Schlumberger



N* Novosibirsk State University
*THE REAL SCIENCE



Los Alamos
NATIONAL LABORATORY
EST. 1943



OAK RIDGE
National Laboratory



ANSYS



INTEL ADVISOR : KEY SPECIFICATIONS

Processors

- IA-32 or Intel® 64 architectures
- SSE2 or later SIMD instruction sets: Intel® Advanced Vector Extensions, Intel® Advanced Vector Extensions 2, Intel® Advanced Vector Extensions 512, and more

Operating Systems

- Windows
- Linux
- macOS (viewer only, no data collection)

Programming Languages

- C
- C++
- Fortran

Compilers

- Compilers from Intel
- Microsoft Visual C++* compiler
- GNU Compiler Collection (GCC)*
- Other compilers that follow the same standards

Development Environments and Interfaces

- Run as stand-alone applications
- Integrated with Microsoft Visual Studio*
- Command Line interface
- Python API

<https://software.intel.com/en-us/advisor>

Intel Advisor

