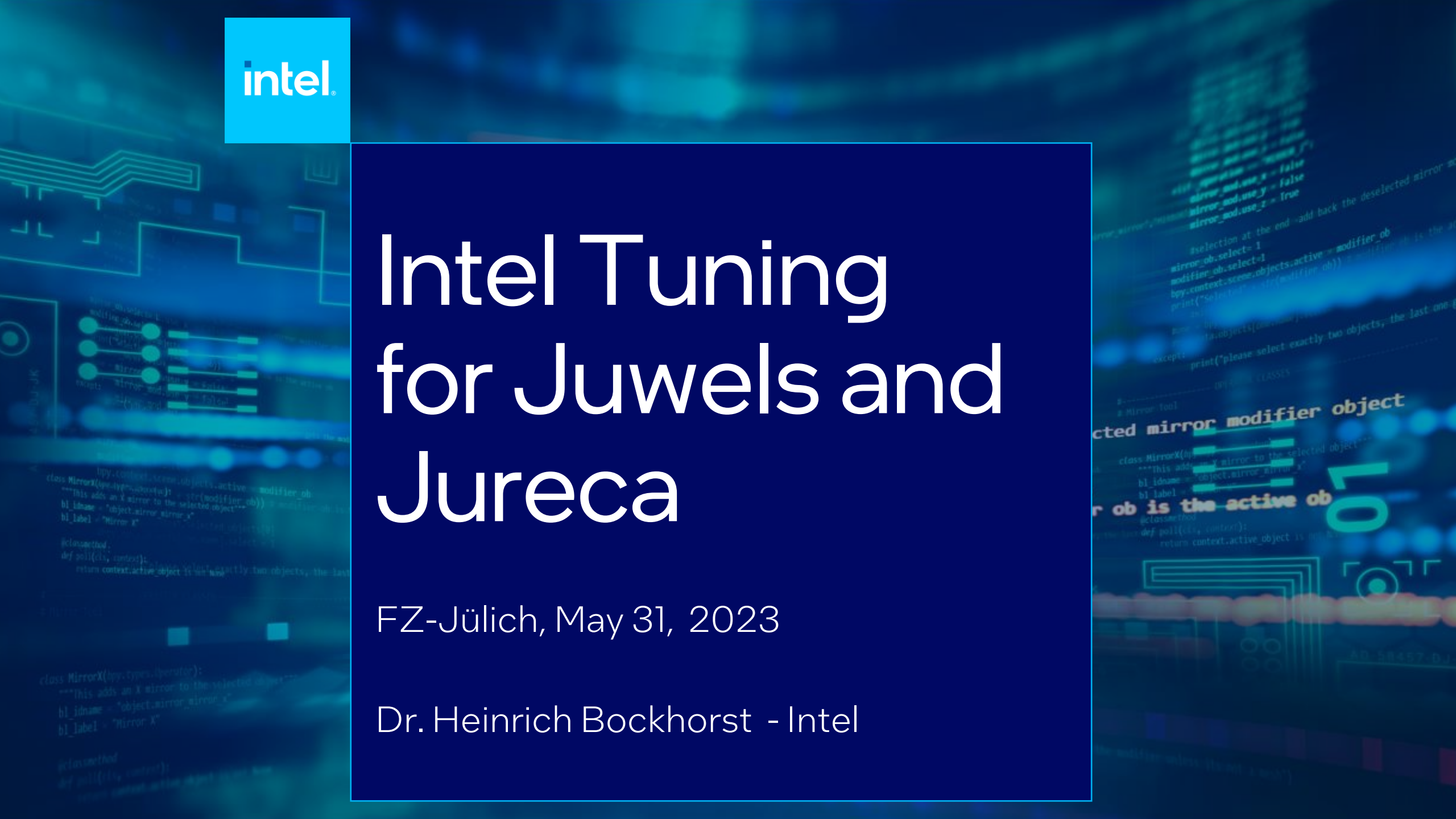


Intel Tuning for Juwels and Jureca

FZ-Jülich, May 31, 2023

Dr. Heinrich Bockhorst - Intel

Agenda

- Intel® Parallel Studio XE → oneAPI
- Intel® Compiler
- Application Performance Snapshot (APS)
- VTune Profiler
- Advisor
- Intel® MPI

Cross-Architecture Programming for Accelerated Compute, Freedom of Choice for Hardware

oneAPI: Industry Initiative & Intel Products

One Intel Software & Architecture group
Intel Architecture, Graphics & Software
November 2020

oneAPI

Intel® oneAPI **Base** Toolkit

Native Code Developers

A core set of high-performance tools for building C++, Data Parallel C++ applications & oneAPI library-based applications

Add-on **Domain-Specific** Toolkits

Specialized Workloads

Intel® oneAPI Tools for HPC

Deliver fast Fortran, OpenMP & MPI applications that scale

Intel® oneAPI Tools for IoT

Build efficient, reliable solutions that run at network's edge

Intel® oneAPI Rendering Toolkit

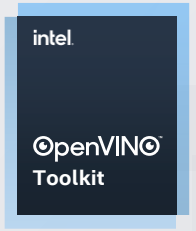
Create performant, high-fidelity visualization applications

Toolkits **powered by oneAPI**

Data Scientists & AI Developers

Intel® AI Analytics Toolkit

Accelerate machine learning & data science pipelines with optimized DL frameworks & high-performing Python libraries



Intel® Distribution of OpenVINO™ Toolkit

Deploy high performance inference & applications from edge to cloud

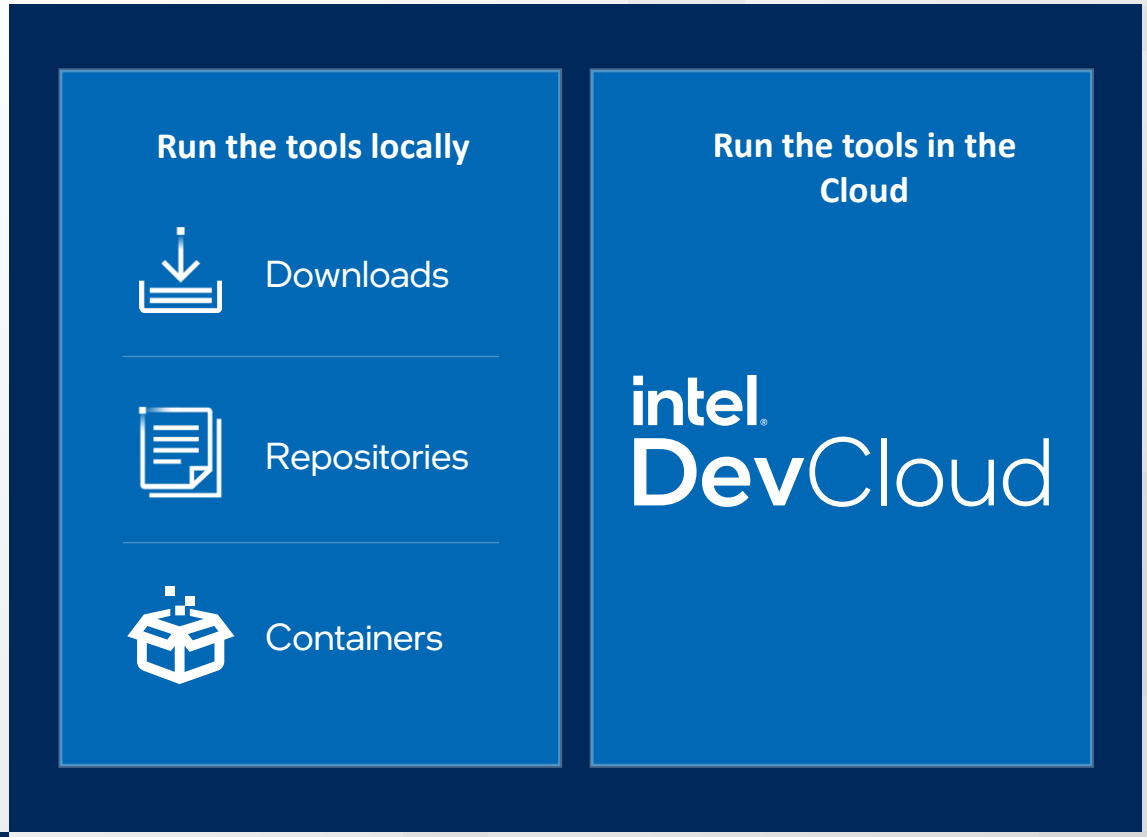
Get started quickly

Code Samples, Quick-start Guides, Webinars, Training

https://software.intel.com/oneapi

https://software.intel.com/devcloud

Note: select “For oneAPI”



oneAPI

Intel® Compilers

Intel® Compilers Going Forward

New underlying back-end Compilation Technology based on LLVM

New compiler technology available today in Intel® oneAPI Base & HPC Toolkit for DPC++, C++ and Fortran

Existing Intel proprietary "IL0" (ICC, IFORT) Compilation Technology compilers provided alongside new compilers

- *CHOICE! Continuity!*

BUT Offload (DPC++ or OpenMP TARGET) supported only with new LLVM-based compilers

- All Intel compilers are available on Juwels/Jureca: $ module load Intel

Intel® Compilers

Intel Compiler	Target	OpenMP Support	OpenMP Offload Support	Included in oneAPI Toolkit
Intel® C++ Compiler Classic, IL0 *icc/icpc/icl*	CPU	Yes	No	HPC
Intel® Fortran Compiler Classic, IL0 *ifort*	CPU	Yes	No	HPC
Intel® Fortran Compiler, LLVM *ifx*	CPU, GPU	Yes	Yes	HPC
Intel® oneAPI DPC++/C++ Compiler, LLVM *dpcpp*	CPU, GPU, FPGA*	Yes	Yes	Base
Intel® oneAPI DPC++/C++ Compiler, LLVM *icx/icpx*	CPU GPU*	Yes	Yes	Base

Compiler Binary Compatible and Linkable!

tinyurl.com/oneapi-standalone-components

Intel® Compiler Classic

still available but no further development!

Common optimization options (icc/ifort)

	Linux*
Disable optimization	-O0
Optimize for speed (no code size increase)	-O1
Optimize for speed (default)	-O2
High-level loop optimization	-O3
Create symbols for debugging	-g
Multi-file inter-procedural optimization	-ipo
Optimize for speed across the entire program ("prototype switch") ***fast* options definitions changes over time!**	-fast same as: -ipo –O3 -no-prec-div –static –fp-model fast=2 –xHost
OpenMP support	-qopenmp

https://tinyurl.com/icc-user-guide

High-Level Optimizations

Basic Optimizations with icc -O...

-O0 no optimization; sets -g for debugging

-O1 scalar optimizations
excludes optimizations tending to increase code size

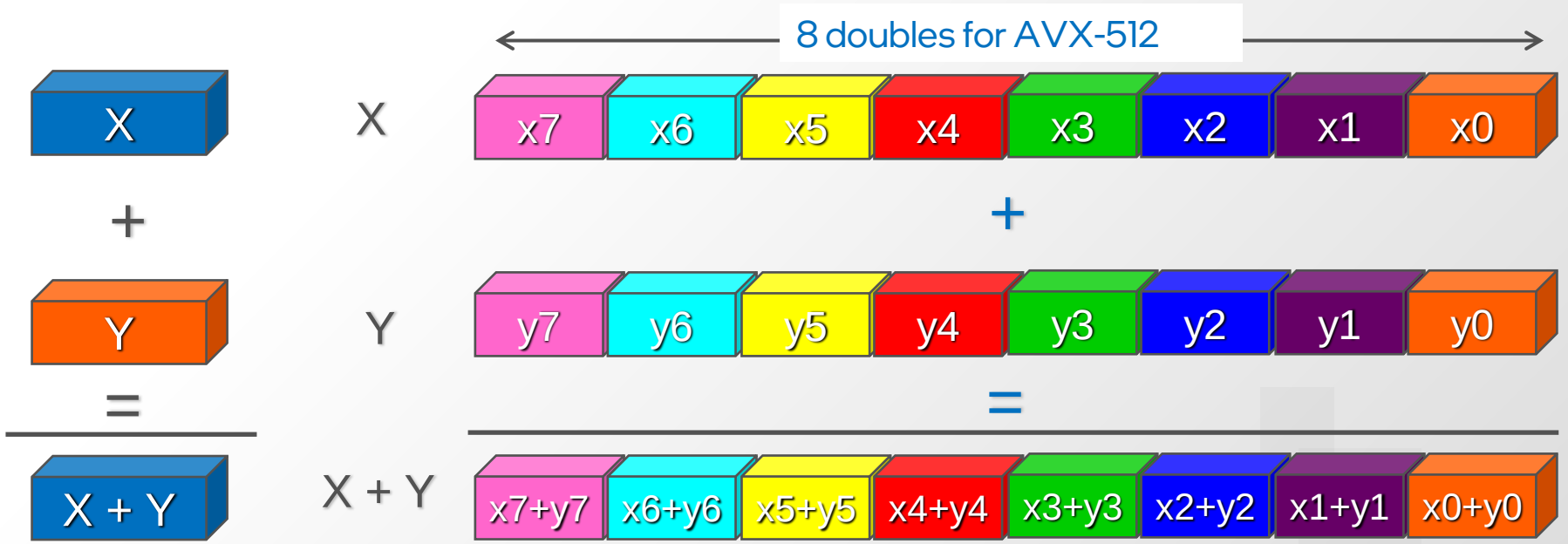
-O2 **default** for icc/icpc (except with -g)
includes **auto-vectorization**; some loop transformations, e.g. unrolling, loop interchange;
inlining within source file;
start with this (after initial debugging at -O0)

-O3 more aggressive loop optimizations
including cache blocking, loop fusion, prefetching, ...
suited to applications with loops that do many floating-point calculations or process large data sets

SIMD: Single Instruction, Multiple Data

```
for (i=0; i<n; i++) z[i] = x[i] + y[i];
```

- Scalar mode
  - one instruction produces one result
  - E.g. vaddss, (vaddsd)
- Vector (SIMD) mode
  - one instruction can produce multiple results
  - E.g. vaddps, (vaddpd)



Basic Vectorization Switches I

- Linux*, OS X*: **-x<feature>**
  - Might enable Intel processor specific optimizations
  - Processor-check added to “main” routine:
    Application errors in case SIMD feature missing or non-Intel processor with appropriate/informative message
  - Example: -xCORE-AVX512 (Juwels Xeon SKL)

- Linux*, OS X*: **-ax<features>**
  - Multiple code paths: baseline and optimized/processor-specific
  - Multiple SIMD features/paths possible, e.g.: **-axSSE2,CORE-AVX512**
  - Baseline code path defaults to **-xSSE2**

Basic Vectorization Switches II

- Special switch for Linux*, OS X*: **`-xHost`**
- Compiler checks SIMD features of current host processor (where built on) and makes use of latest SIMD feature available
- Code only executes on processors with same SIMD feature or later as on build host

LLVM-BASED INTEL COMPILERS

What is ICX?

- Close collaboration with Clang*/LLVM* community
- ICX is Clang front-end (FE), LLVM infrastructure
  - PLUS Intel proprietary optimizations and code generation
- Clang FE pulled down frequently from open source, kept current
  - Always up to date in ICX
  - We contribute! Pushing enhancements to both Clang and LLVM
- Enhancements working with community – better vectorization, opt-report, for example

tinyurl.com/blog-on-icx

Major Changes Overview

tinyurl.com/icc-to-icx-migration-guide

- LLVM is a different compilation technology. EXPECT differences
- Options:
  - **icx -qnextgen-diag** option to get a list of supported and unsupported options
- Use `-fiopenmp` or `-fiopenmp-simd` for OpenMP
- C/C++ Pragmas – a lot of Intel proprietary ones not supported
  - enable -Wunknown-pragmas to warn on unsupported pragmas
- __INTEL_LLVM_COMPILER is defined instead of __INTEL_COMPILER

Please switch to icx/icpx Compiler!

- Deprecation planed for second half of 2023
- Check the user guide for supported flags:

  https://www.intel.com/content/www/us/en/docs/dpcpp-cpp-compiler/developer-guide-reference/2023-1/alphabetical-option-list.html

- Check results and compare with icc/icpc results:

  -fp-model=fast is the default

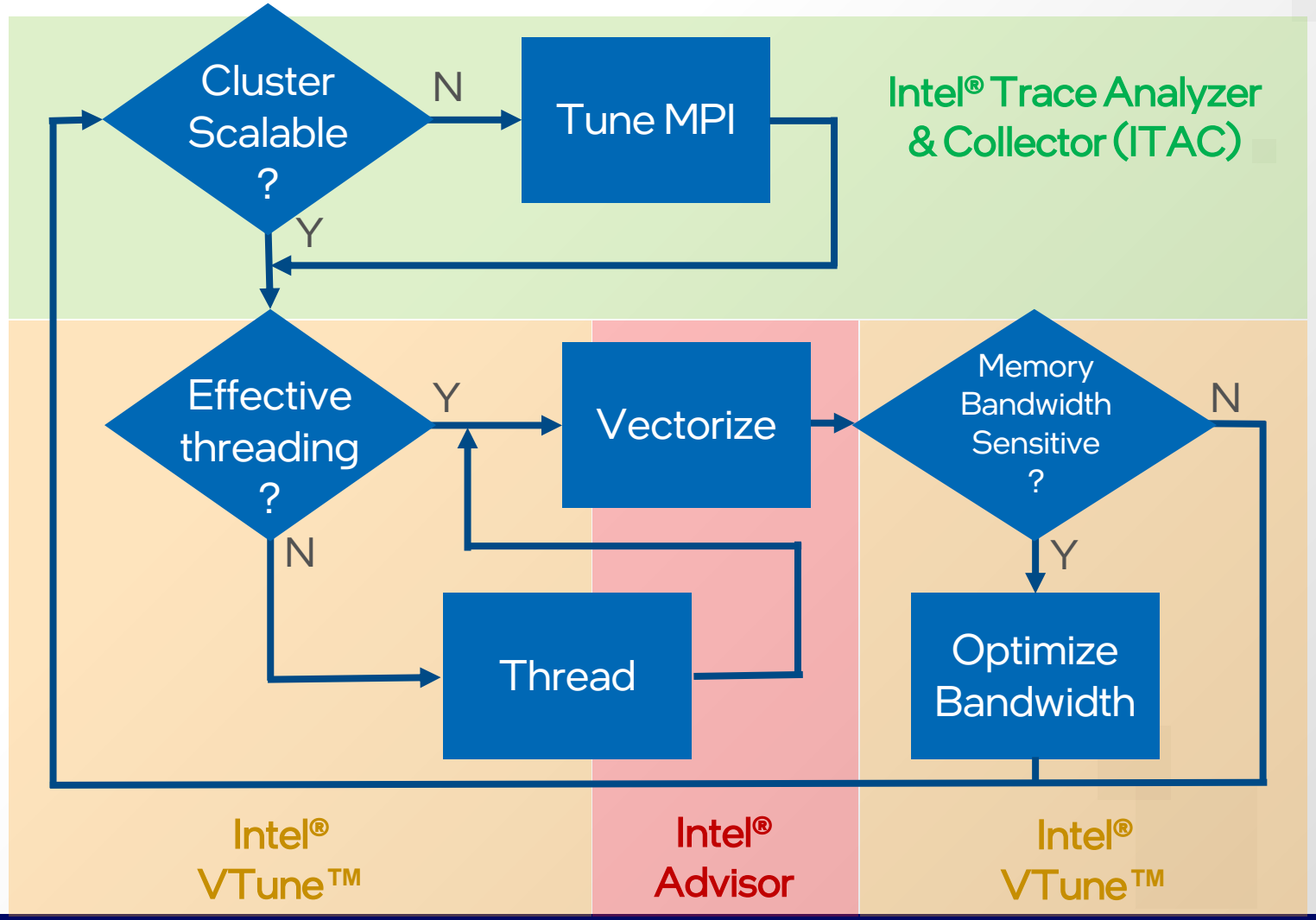
  -fp-model=precise might help to reproduce previous results

Build your own compiler (only for experts)

- Most of the features are included in the public llvm Intel version. You may test and contribute to the development.
- Interested? Check out: https://intel.github.io/llvm-docs/GetStartedGuide.html
- Build a clang compiler with latest features ahead of icpx and icx
- Some features may be missing in the public version
- Can also configure and build a CUDA backend compiler for offload to NVIDIA cards
- Existing compiler on Juwels for CUDA backend: **/p/usersoftware/paj1720/README.txt**

Which tool should I use?

Performance Analysis Tools for Diagnosis



Before dive to a particular tool..

- How to assess easily any potential in performance tuning?
- What to use on big scale not be overwhelmed with huge trace size, post processing time and collection overhead?
- Which tool should I use first?

- Answer: try Application Performance Snapshot (APS)

- Look for VTune module if available

APS Usage

Setup Environment

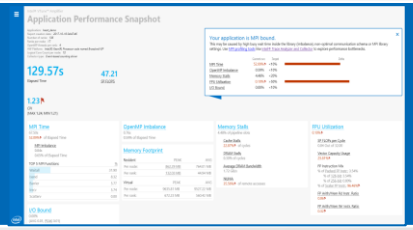
- $ source <path_to_vtune>/vtune_vars.sh # or load module

Run Application

- $ aps <application and args>
- MPI: $ mpirun <mpi options> aps <application and args>

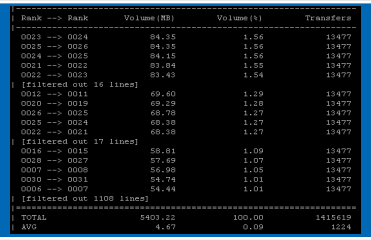
Generate Report on Result Folder

- $ aps –report <result folder>



Generate CL reports with detailed MPI statistics on Result Folder

- $ aps-report –<option> <result folder>



Application Performance Snapshot (APS)

Data in One Place: MPI+OpenMP+Memory Floating Point

Quick & Easy Performance Overview

- Does the app need performance tuning?

MPI & non-MPI Apps†

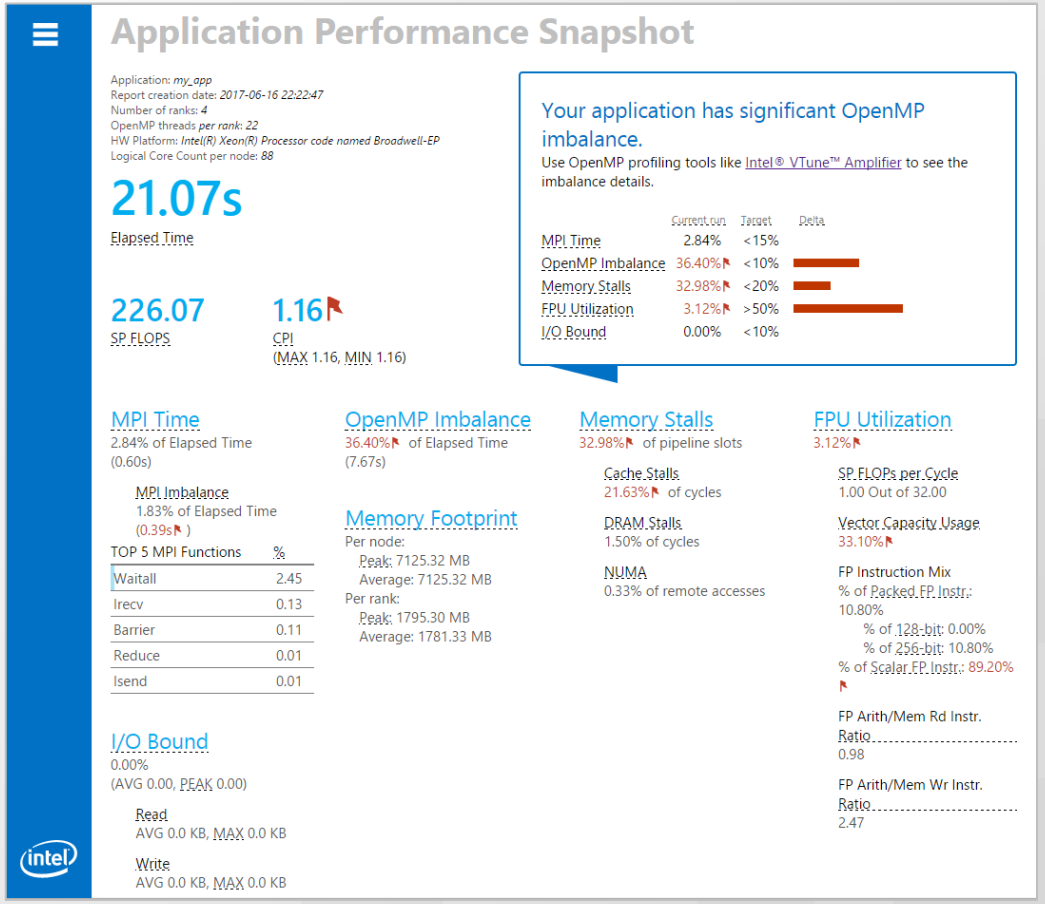
- Distributed MPI with or without threading
- Shared memory applications

Popular MPI Implementations Supported

- Intel® MPI Library
- MPICH & Cray MPI

Richer Metrics on Computation Efficiency

- CPU (processor stalls, memory access)
- FPU (vectorization metrics)



†MPI supported only on Linux*

APS Command Line Reports – Advanced MPI statistics

- Data Transfers for Rank-to-Rank Communication
  - aps-report –x <result>

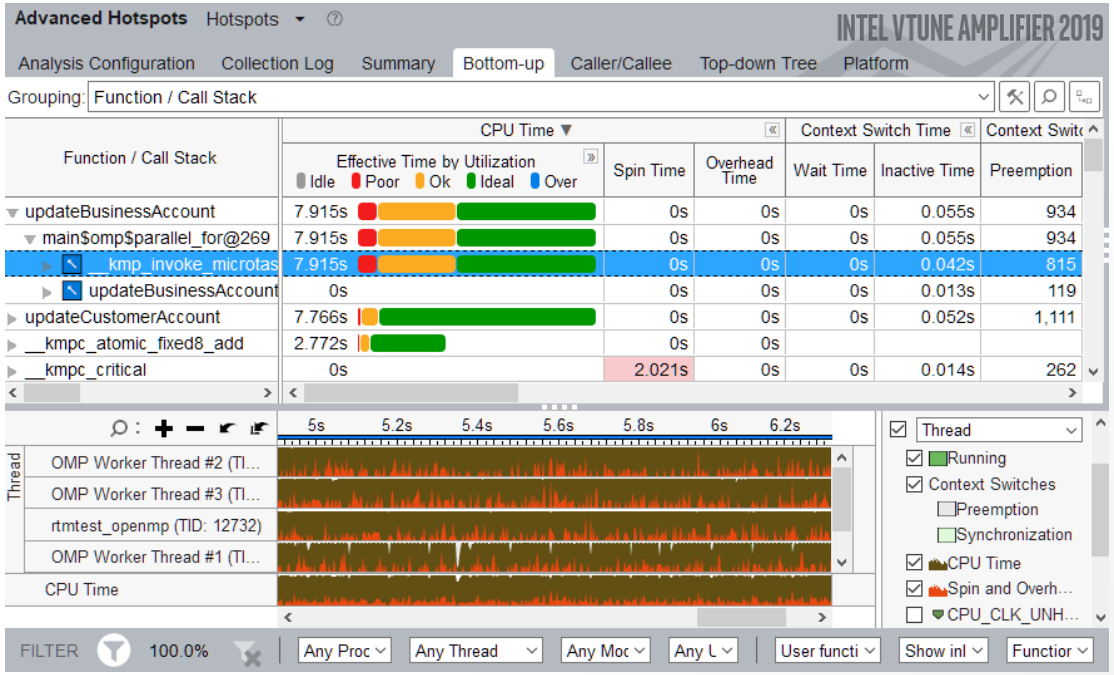
And many others – check
  - aps-report -help

Rank --> Rank	Volume(MB)	Volume(%)	Transfers
0023 --> 0024	84.35	1.56	13477
0025 --> 0026	84.35	1.56	13477
0024 --> 0025	84.15	1.56	13477
0021 --> 0022	83.84	1.55	13477
0022 --> 0023	83.43	1.54	13477
[filtered out 16 lines]			
0012 --> 0011	69.60	1.29	13477
0020 --> 0019	69.29	1.28	13477
0026 --> 0025	68.78	1.27	13477
0025 --> 0024	68.38	1.27	13477
0022 --> 0021	68.38	1.27	13477
[filtered out 17 lines]			
0016 --> 0015	58.81	1.09	13477
0028 --> 0027	57.69	1.07	13477
0007 --> 0008	56.98	1.05	13477
0030 --> 0031	54.74	1.01	13477
0006 --> 0007	54.44	1.01	13477
[filtered out 1108 lines]			
TOTAL	5403.22	100.00	1415619
AVG	4.67	0.09	1224

Intel® VTune™ Profiler

Analyze & Tune Application Performance

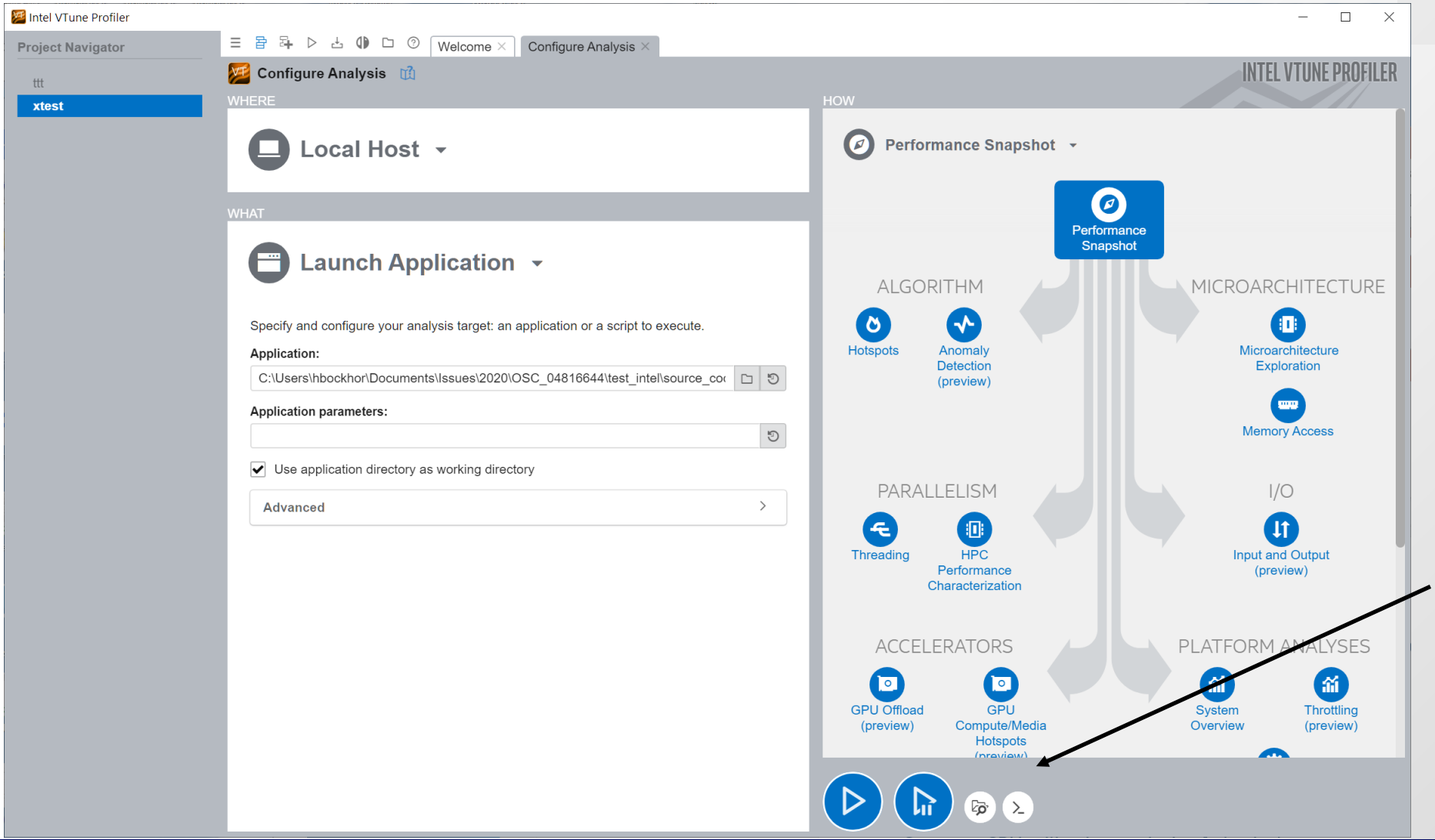
Intel® VTune™ Profiler



- Accurately profile C, C++, Fortran*, Python*, Go*, Java*, or any mix
- Optimize CPU, threading, memory, cache, storage & more
- Take advantage of Priority Support
  - Connects customers to Intel engineers for confidential inquiries (paid versions)
- A more accessible user interface provides a simplified profiling workflow
- Smarter, faster Application Performance Snapshot: Analyze CPU utilization of physical cores, pause/resume, more… (Linux*)

https://software.intel.com/content/www/us/en/develop/tools/vtune-profiler/get-started.html

Start a new Project



- Use GUI
- Or Command -Line

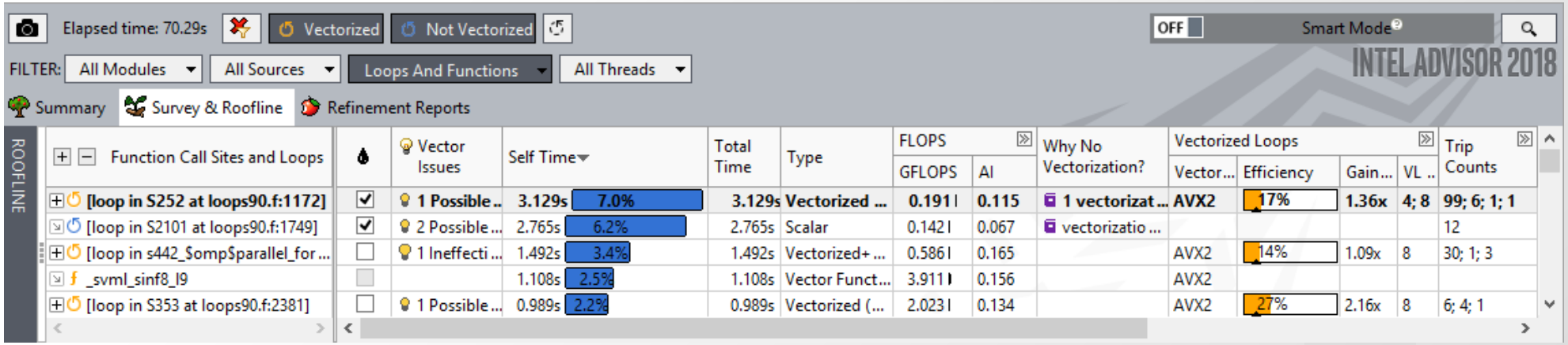
Get Command-Line

INTEL® ADVISOR

Intel® Advisor – Vectorization Advisor

Get breakthrough vectorization performance

- 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
- The data and guidance you need:
  - Compiler diagnostics + Performance Data + SIMD efficiency
  - Detect problems & recommend fixes
  - Loop-Carried Dependency Analysis
  - Memory Access Patterns Analysis

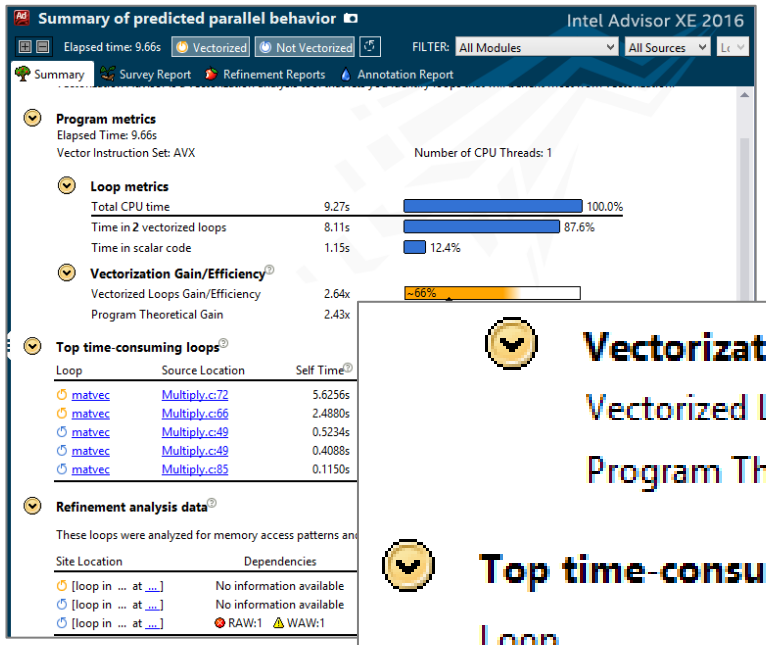


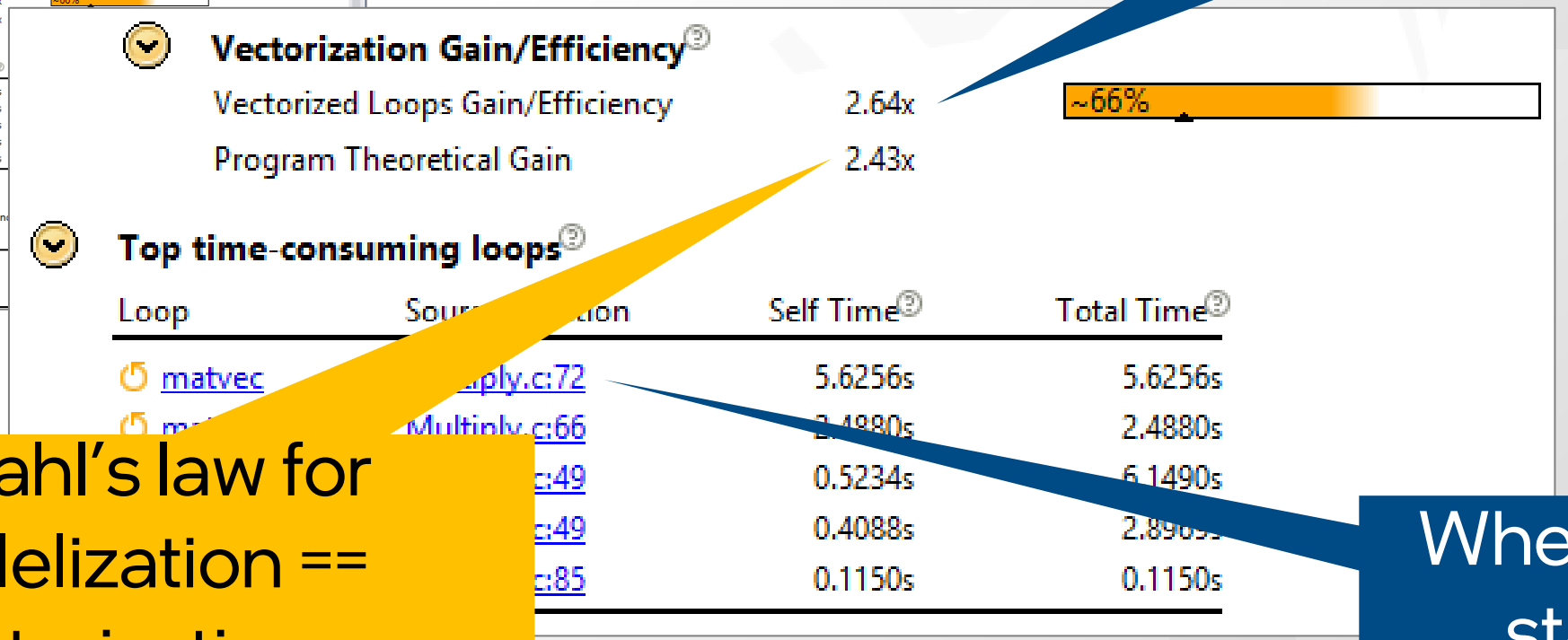
Optimize for AVX-512 with/without access to AVX-512 hardware

Part of oneAPI Base Toolkit

software.intel.com/advisor

Summary View: Plan Your Next Steps





What can I expect to gain?

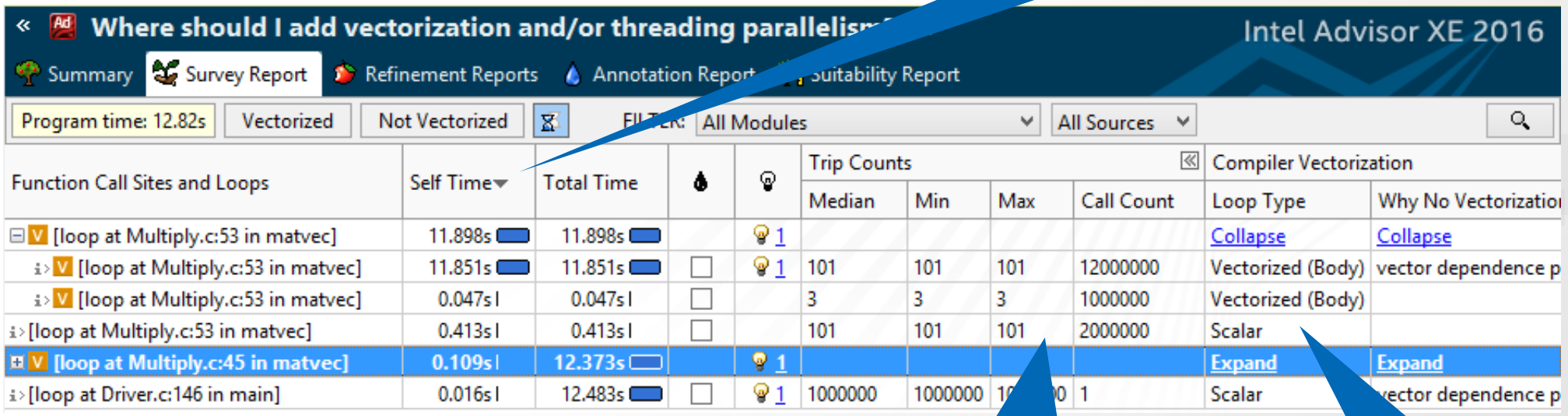
Amdahl's law for parallelization == vectorization

Where do I start?

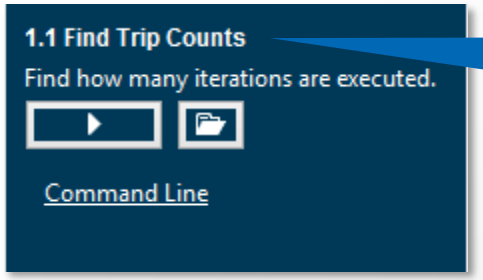
Critical Data Made Easy

Loop Trip Counts

Knowing the time spent in a loop is not enough!



Function Call Sites and Loops	Self Time	Total Time			Trip Counts				Compiler Vectorization	
					Median	Min	Max	Call Count	Loop Type	Why No Vectorizatio
[loop at Multiply.c:53 in matvec]	11.898s	11.898s		1					Collapse	Collapse
[loop at Multiply.c:53 in matvec]	11.851s	11.851s		1	101	101	101	12000000	Vectorized (Body)	vector dependence p
[loop at Multiply.c:53 in matvec]	0.047s	0.047s			3	3	3	1000000	Vectorized (Body)	
[loop at Multiply.c:53 in matvec]	0.413s	0.413s			101	101	101	2000000	Scalar	
[loop at Multiply.c:45 in matvec]	0.109s	12.373s		1					Expand	Expand
[loop at Driver.c:146 in main]	0.016s	12.483s		1	1000000	1000000	10	1	Scalar	ector dependence p



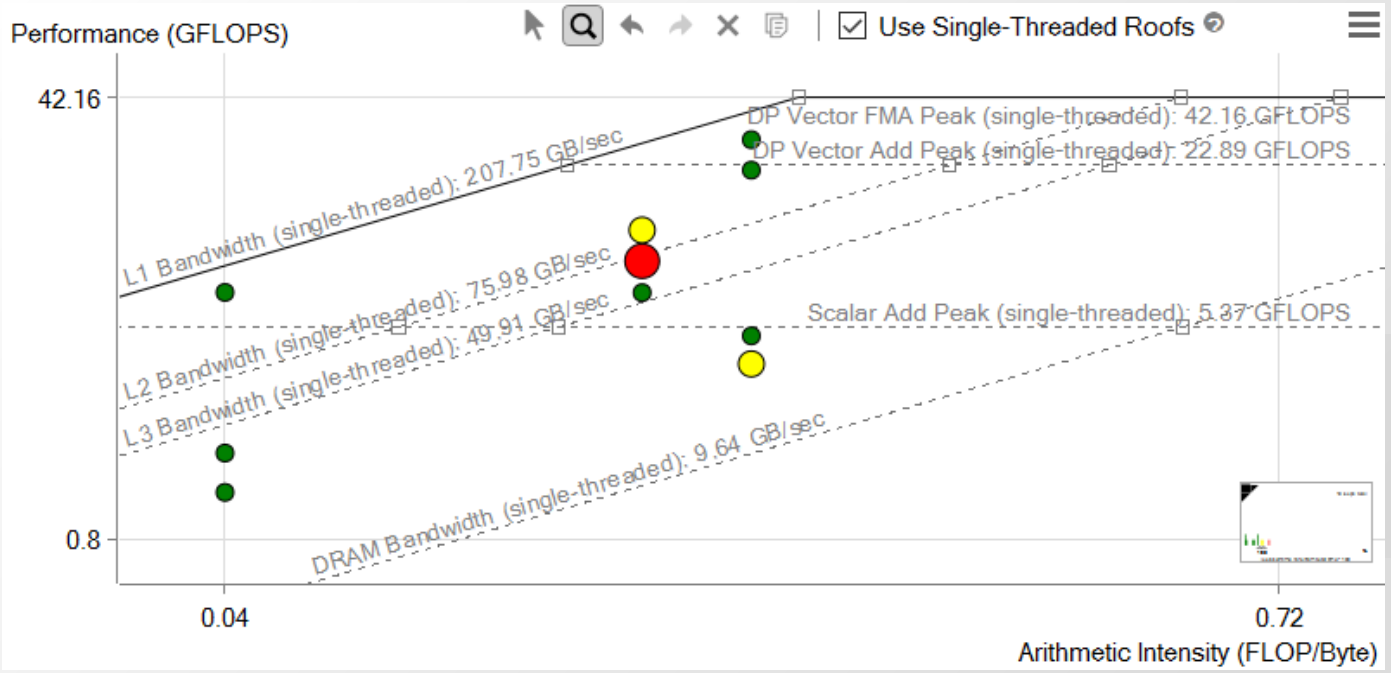
Check actual trip counts

Loop is iterating 101 times but called > million times

Since the loop is called so many times it would be a win if we can get it to vectorize.

What is a Roofline Chart?

- A Roofline Chart plots application performance against hardware limitations.
  - Where are the bottlenecks?
  - How much performance is being left on the table?
  - Which bottlenecks can be addressed, and which *should* be addressed?
  - What's the most likely cause?
  - What are the next steps?



Roofline first proposed by University of California at Berkeley:
Roofline: An Insightful Visual Performance Model for Multicore Architectures, 2009
Cache-aware variant proposed by University of Lisbon:
Cache-Aware Roofline Model: Upgrading the Loft, 2013

Advisor Resources

Intel® Advisor

- Product page – overview, features, FAQs…
- What's New?
- Training materials – Cookbooks, User Guide, Tutorials
- Support Forum
- Online Service Center - Secure Priority Support

Additional Analysis Tools

- Intel® VTune™ Profiler – performance profiler
- Intel® Inspector – memory and thread checker/ debugger
- Intel® Trace Analyzer and Collector - MPI Analyzer and Profiler

Additional Development Products

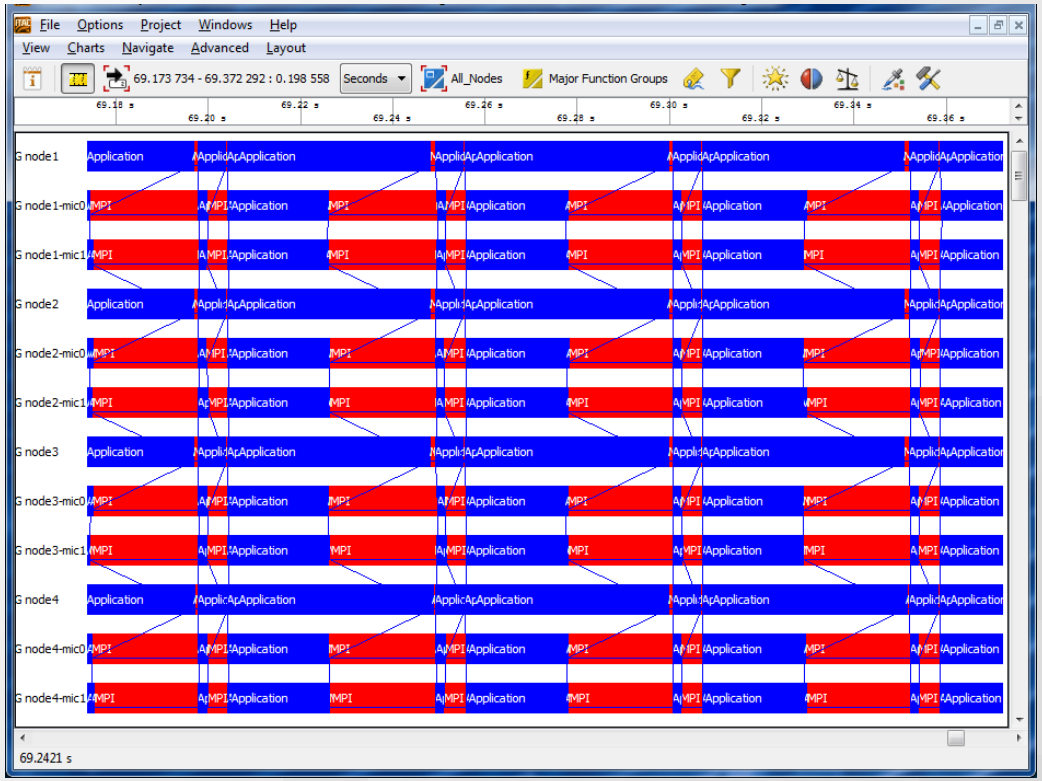
- Intel® oneAPI Toolkits

ITAC for MPI Analysis

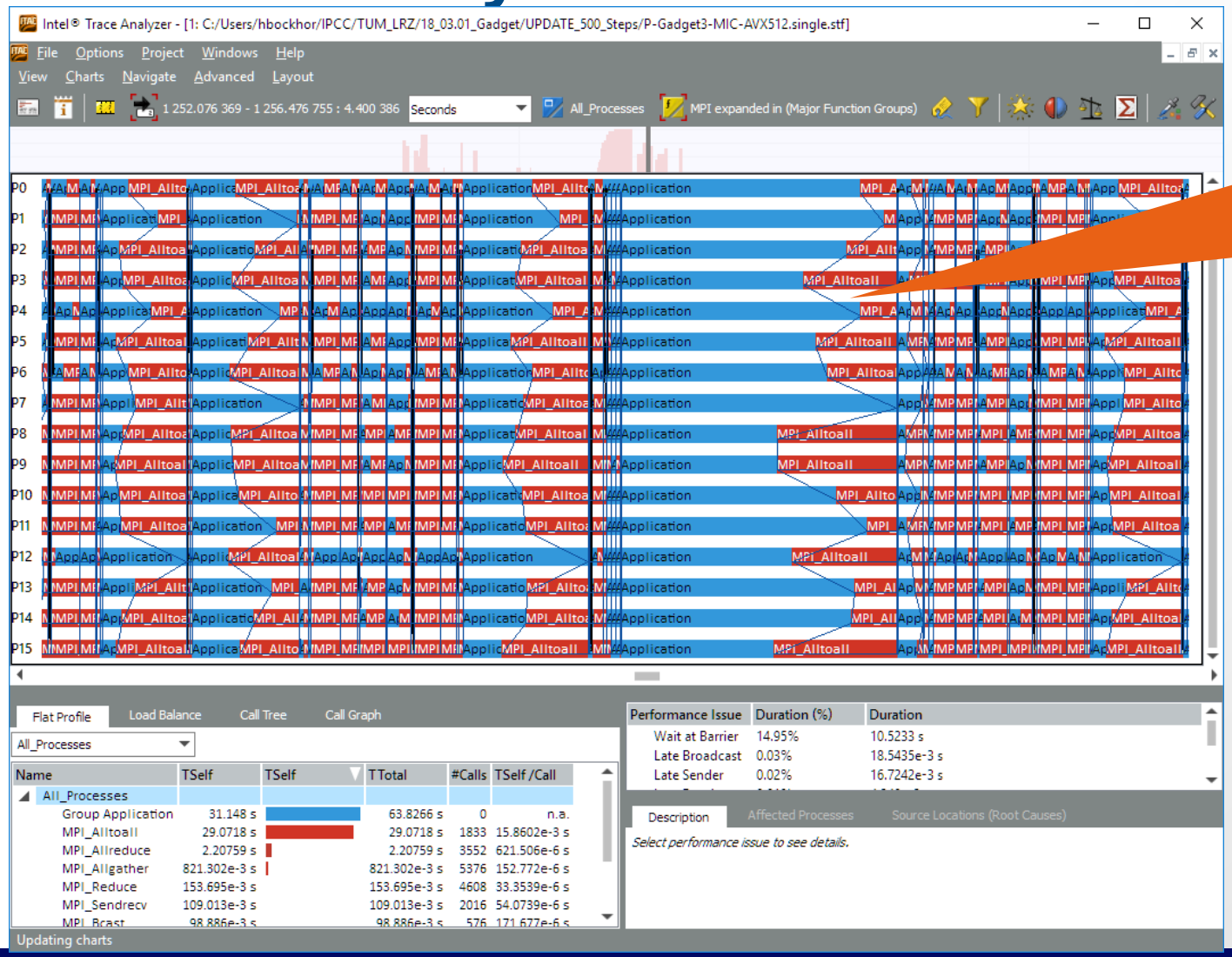
Efficiently Profile MPI Applications

Intel® Trace Analyzer & Collector

- Helps Developers
  - Visualize & understand parallel application behavior
  - Evaluate profiling statistics & load balancing
  - Identify communication hotspots
- Features
  - Event-based approach
  - Low overhead
  - Excellent scalability
  - Powerful aggregation & filtering functions
  - Idealizer
  - Scalable



ITAC Analysis



High Load imbalance causes MPI_Alltoall time

Online Resources

- Intel® MPI Library product page.
  - www.intel.com/go/mpi
- Intel® Trace Analyzer and Collector product page
  - www.intel.com/go/traceanalyzer
- Intel® Clusters and HPC Technology forums
  - http://software.intel.com/en-us/forums/intel-clusters-and-hpc-technology

Intel Modules installed on Juwels

- Compiler:
  - check available: $ module spider Intel
  - default: $ module load Intel
- VTune + APS:
  - check available: $ module spider vtune
  - default: $ module load VTune
- Advisor:
  - check available $ module spider advisor
  - default: $ module load Advisor
- Intel MPI:
  - check available: $ module spider intelMPI
  - default: $ module load IntelMPI
- Intel MKL:
  - check available: $ module spider mkl
  - default: $ module load imkl
- Alternative for Base Kit tools:

```
$ source /p/usersoftware/paj1720/intel/oneapi/setvars.sh
```

How to start?

- Compile with minimal options and run with APS (will provide tuning tips)
- Compile with -xhost and and check timing and APS report
- Optional! Compile with –xhost and –no-vec
  disables vectorization. Compare with previous timing
- Use: VTune Profiler: $ module load VTune/<version>
- Use: Advisor: $ module load Advisor/<version>
- Google for Intel related topics → Intel Developer Zone etc.
- For APS/VTune add to your batch job: `#SBATCH --disable-perfparanoid`

- Please set thread affinity e.g.: $ export KMP_AFFINITY=scatter,verbose
  This can speed up OMP programs up to 10X!

- Any questions: Heinrich.Bockhorst@Intel.com

Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation. Learn more at intel.com or from the OEM or retailer.

Your costs and results may vary.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

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. https://software.intel.com/en-us/articles/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. See backup for configuration details. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

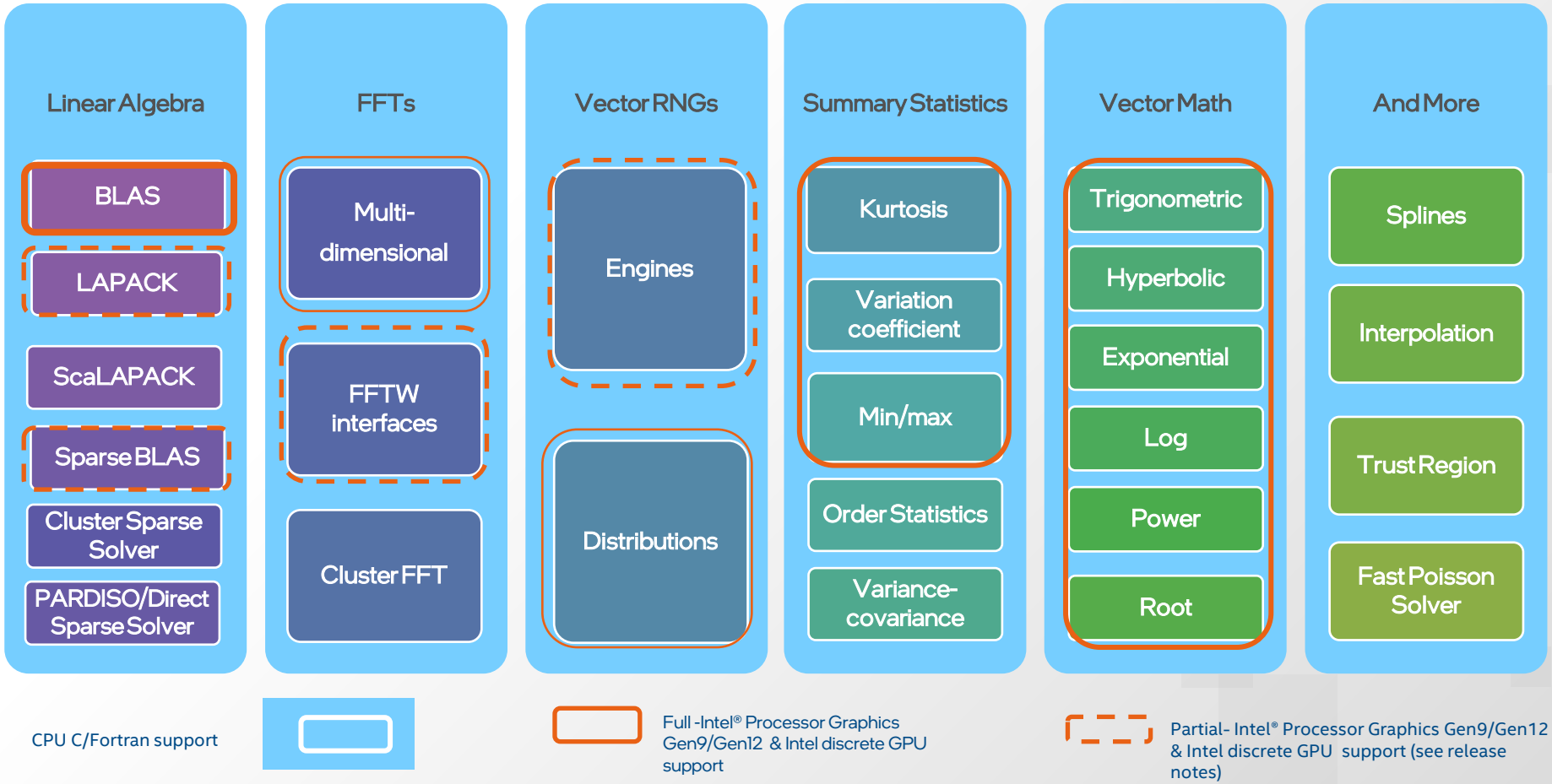
No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.


intel

Intel® oneAPI Math Kernel Library (oneMKL)

What's Inside Intel® oneAPI Math Kernel Library (oneMKL)



What's New for Intel® oneAPI Math Kernel Library(oneMKL) 2021.2-2022.0

- Introduced GPU support for the following new functionality:
  - BLAS – Batch & copy for unified shared memory(USM) & buffer APIs
  - Vector Statistics - RNG multinomial, PoissonV, hypergeometric, negative binomial and binomial distributions.
  - BLAS - Added SYCL support for in-place and out of place matrix copy/transposition
  - LAPACK - Enabled C/Fortran OpenMP offload support for select functions.
  - Sparse BLAS – Added support for variance matrix-matrix multiplication operations.
- General performance optimizations
- For detailed information please refer to the oneMKL Release Notes

Basic Vectorization Switches III

- Special switch in addition to CORE-AVX512: `-qopt-zmm-usage=[keyword]`
  - `[keyword] = [high | low]` ; Note: “low” is the default
  - Why choosing a defensive vectorization level?

    Frequency drops in vectorized parts. Frequency does not immediately increase after the vectorized loop. Too many small vectorized loops will decrease the performance for the serial part.

Next steps

- Toolkits are free but maybe too large (> 10 GB). For this workshop you may download to your laptop: VTune, Advisor, Inspector

- Standalone tools download: https://software.intel.com/content/www/us/en/develop/articles/oneapi-standalone-components.html


intel