Intel® Tuning for Juwels and Jureca

Dr. Heinrich Bockhorst

FZJ 2025, May 15th

Agenda

- oneAPI Initiative
- Intel® Compiler
- Application Performance Snapshot (APS)
- VTune Profiler
- Advisor
- MKL and MPI

oneAPI Industry Initiative

Break the Chains of Proprietary Lock-in

Freedom to Make Your Best Choice

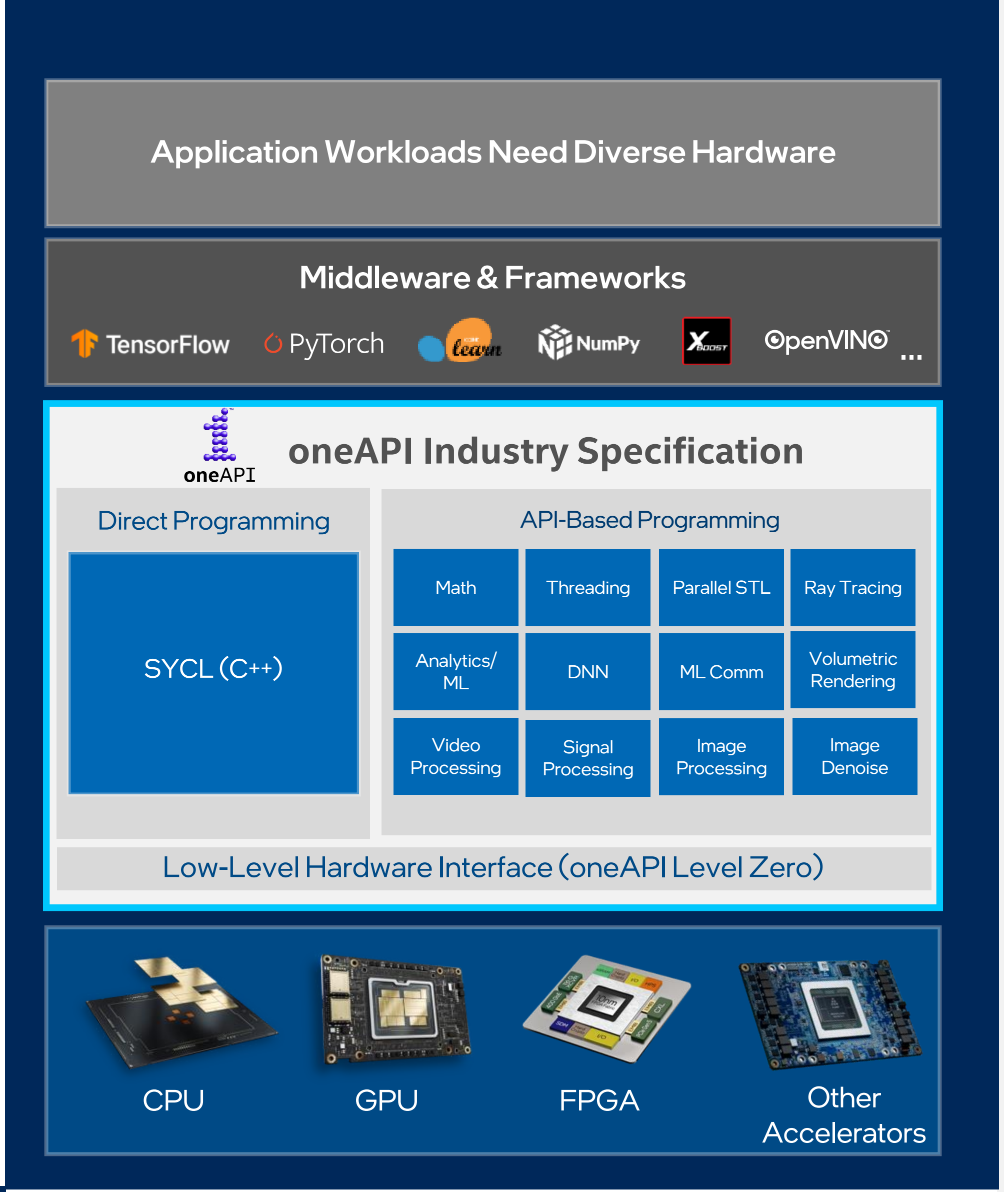
- One programming model for multiple architectures and vendors
- Cross-architecture code reuse for freedom from vendor lock-in

Realize all the Hardware Value

- Performance across CPU, GPUs, FPGAs, and other accelerators
- Expose and exploit cutting-edge features of the latest hardware

Develop & Deploy Software with Peace of Mind

- Open industry standards provide a safe, clear path to the future
- Compatible with existing languages and programming models including C, C++ with SYCL, Python, OpenMP, Fortran, and MPI
- Powerful libraries for acceleration of domain-specific functions



Application Workloads Need Diverse Hardware

Middleware & Frameworks

TensorFlow PyTorch learn NumPy XBOOST OpenVINO ...

oneAPI Industry Specification

Direct Programming

SYCL (C++)

API-Based Programming

Math | Threading | Parallel STL | Ray Tracing
Analytics/ML | DNN | ML Comm | Volumetric Rendering
Video Processing | Signal Processing | Image Processing | Image Denoise

Low-Level Hardware Interface (oneAPI Level Zero)

CPU GPU FPGA Other Accelerators

oneAPI

The productive, smart path to freedom for accelerated computing from the economic and technical burdens of proprietary programming models

Visit oneapi.com for more details

Intel® oneAPI DPC++/C++ Compiler

Conformant with SYCL 2020 Specification

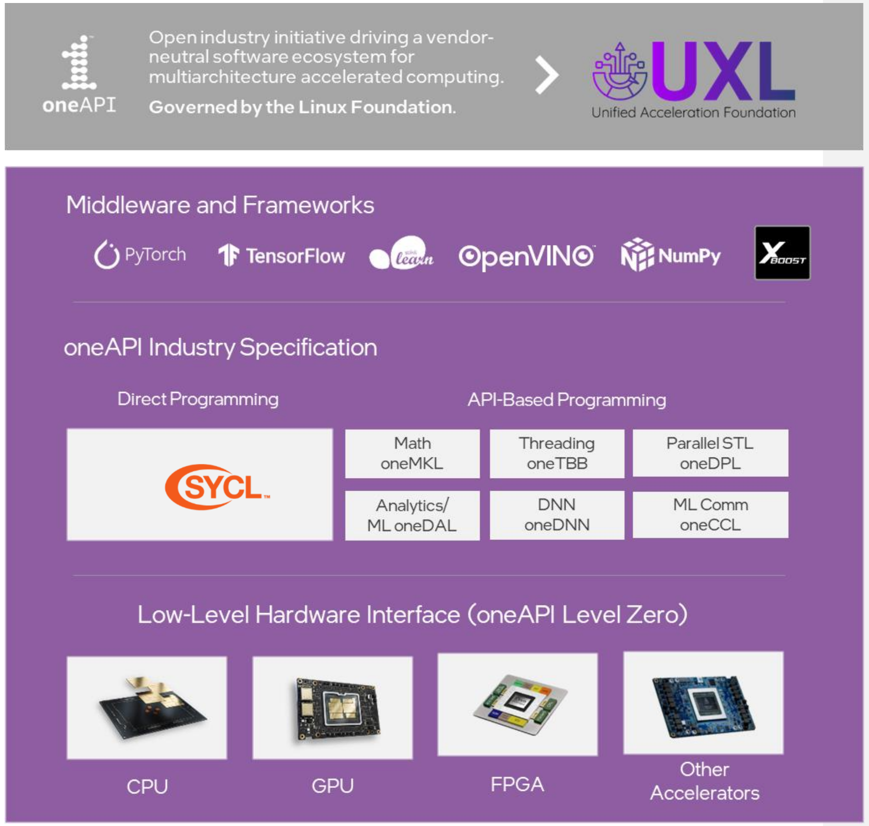
Unified Shared Memory Parallel Reductions, Work Group Algorithms, Class Template Argument Deductions, Simplification of Accessors, Expanded Interoperability, and more …

An Industry First:

SYCL 2020 Conformance on CPU and GPU

Intel is proud to contribute to a revolution anchored in SYCL:

An open ecosystem of
- software developers
- hardware vendors
- compilers and development tools
- APIs and specifications



oneAPI

Open industry initiative driving a vendor-neutral software ecosystem for multiarchitecture accelerated computing.

Governed by the Linux Foundation.

UXL

Unified Acceleration Foundation

Middleware and Frameworks

oneAPI Industry Specification

Direct Programming

API-Based Programming

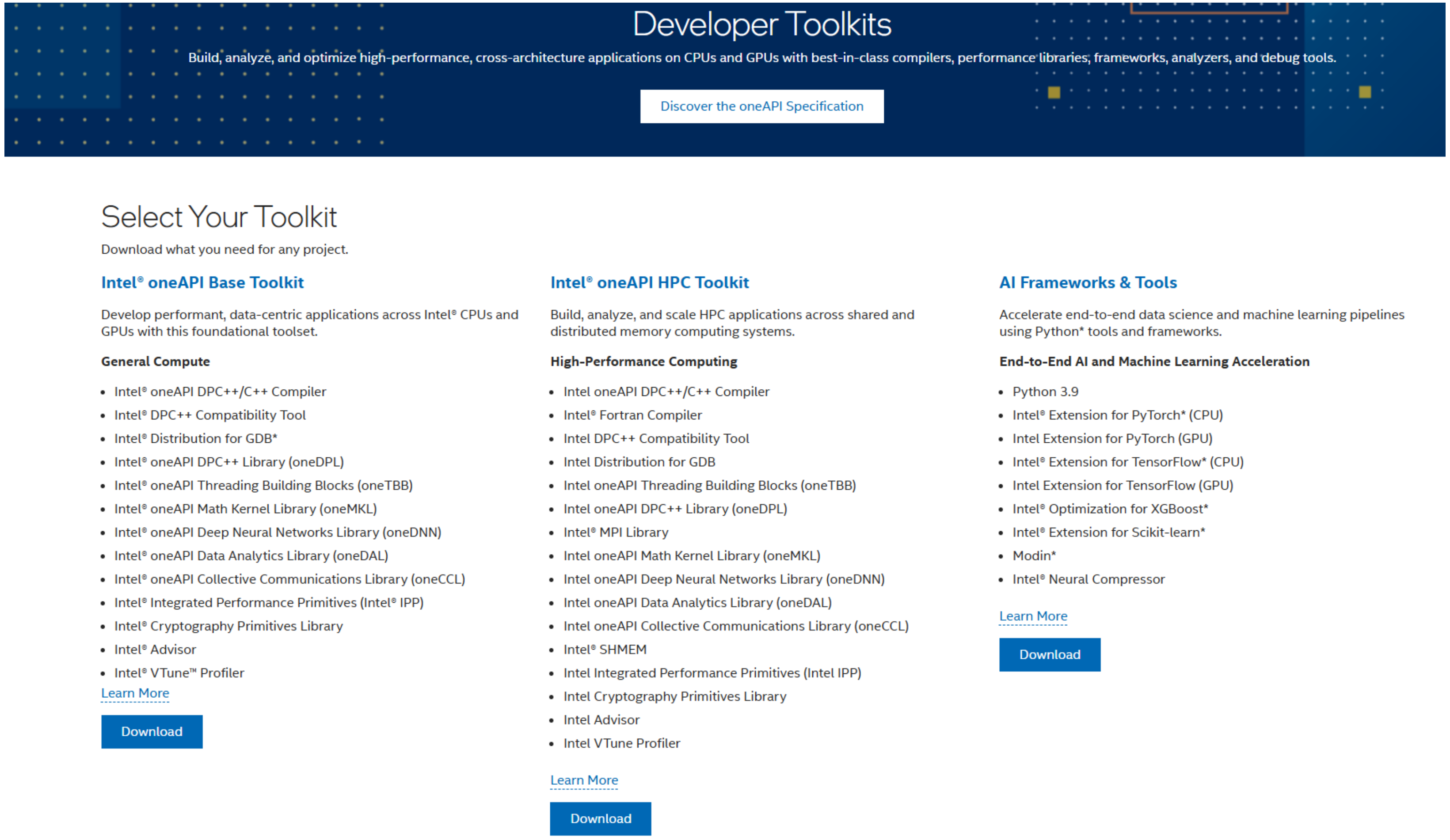
Math oneMKL	Threading oneTBB	Parallel STL oneDPL
Analytics/ ML oneDAL	DNN oneDNN	ML Comm oneCCL

Low-Level Hardware Interface (oneAPI Level Zero)

Intel® oneAPI DPC++/C++ Compiler:
with SYCL towards Open Multiarchitecture Computing

*Other names and brands may be claimed as the property of others. SYCL is a trademark of the Khronos Group Inc.

Toolkits for Developer: free download



Developer Toolkits

Build, analyze, and optimize high-performance, cross-architecture applications on CPUs and GPUs with best-in-class compilers, performance libraries, frameworks, analyzers, and debug tools.

Select Your Toolkit

Download what you need for any project.

Intel® oneAPI Base Toolkit

Develop performant, data-centric applications across Intel® CPUs and GPUs with this foundational toolset.

General Compute

- Intel® oneAPI DPC++/C++ Compiler
- Intel® DPC++ Compatibility Tool
- Intel® Distribution for GDB*
- Intel® oneAPI DPC++ Library (oneDPL)
- Intel® oneAPI Threading Building Blocks (oneTBB)
- Intel® oneAPI Math Kernel Library (oneMKL)
- Intel® oneAPI Deep Neural Networks Library (oneDNN)
- Intel® oneAPI Data Analytics Library (oneDAL)
- Intel® oneAPI Collective Communications Library (oneCCL)
- Intel® Integrated Performance Primitives (Intel® IPP)
- Intel® Cryptography Primitives Library
- Intel® Advisor
- Intel® VTune™ Profiler

Intel® oneAPI HPC Toolkit

Build, analyze, and scale HPC applications across shared and distributed memory computing systems.

High-Performance Computing

- Intel oneAPI DPC++/C++ Compiler
- Intel® Fortran Compiler
- Intel DPC++ Compatibility Tool
- Intel Distribution for GDB
- Intel oneAPI Threading Building Blocks (oneTBB)
- Intel oneAPI DPC++ Library (oneDPL)
- Intel® MPI Library
- Intel oneAPI Math Kernel Library (oneMKL)
- Intel oneAPI Deep Neural Networks Library (oneDNN)
- Intel oneAPI Data Analytics Library (oneDAL)
- Intel oneAPI Collective Communications Library (oneCCL)
- Intel® SHMEM
- Intel Integrated Performance Primitives (Intel IPP)
- Intel Cryptography Primitives Library
- Intel Advisor
- Intel VTune Profiler

AI Frameworks & Tools

Accelerate end-to-end data science and machine learning pipelines using Python* tools and frameworks.

End-to-End AI and Machine Learning Acceleration

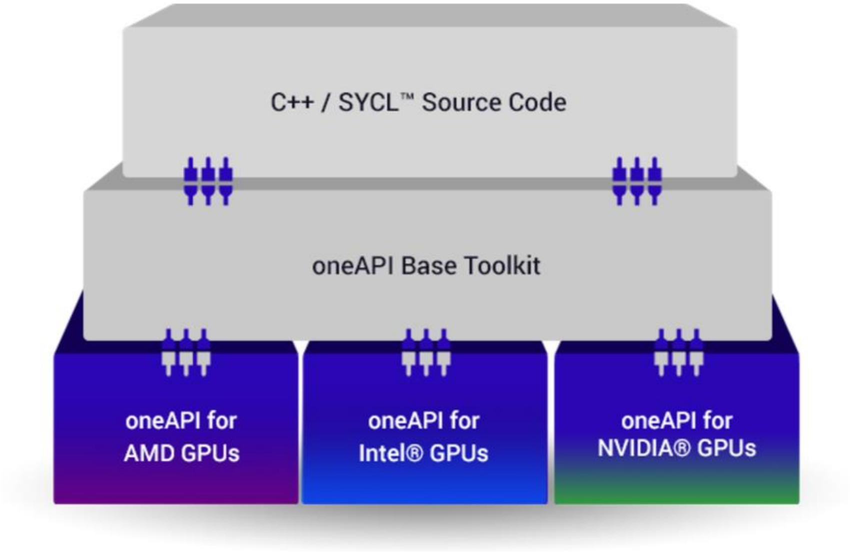
- Python 3.9
- Intel® Extension for PyTorch* (CPU)
- Intel Extension for PyTorch (GPU)
- Intel® Extension for TensorFlow* (CPU)
- Intel Extension for TensorFlow (GPU)
- Intel® Optimization for XGBoost*
- Intel® Extension for Scikit-learn*
- Modin*
- Intel® Neural Compressor

oneAPI for NVIDIA & AMD GPUs

- Free download of binary plugins to Intel® oneAPI DPC++/C++ Compiler:
  - Nvidia GPU
  - AMD GPU
- No need to build from source!
- Plug-ins updated quarterly in sync with SYCL 2020 conformance & performance

Priority Support

- Available through Intel, Codeplay & our channel
- Requires Intel Priority Support for Intel oneAPI DPC++/C++ Compiler
- Intel takes first call, Codeplay delivers backend support
- Codeplay provides access to older plug-in versions

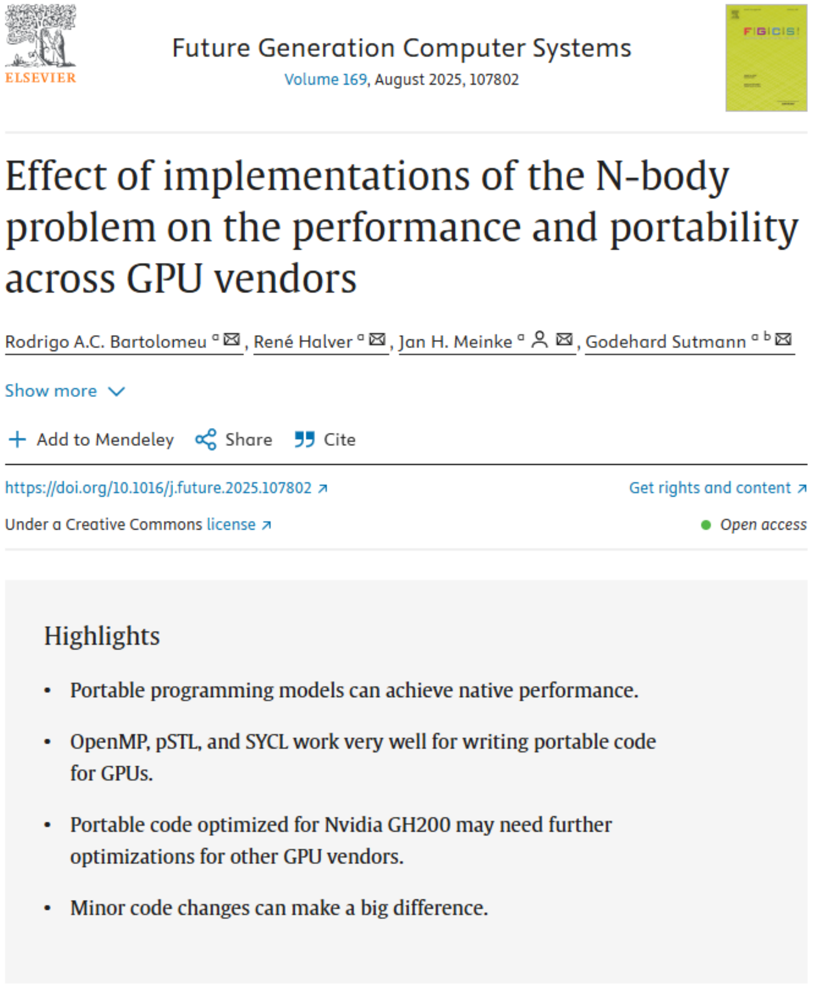


Nvidia GPU plug-in

AMD GPU plug-in

Codeplay blog

Paper on Performance Portability by Jülich Scientists



Future Generation Computer Systems

Volume 169, August 2025, 107802

Effect of implementations of the N-body problem on the performance and portability across GPU vendors

Rodrigo A.C. Bartolomeu [a], René Halver [a], Jan H. Meinke [a], Godehard Sutmann [a] [b]

https://doi.org/10.1016/j.future.2025.107802

Under a Creative Commons license

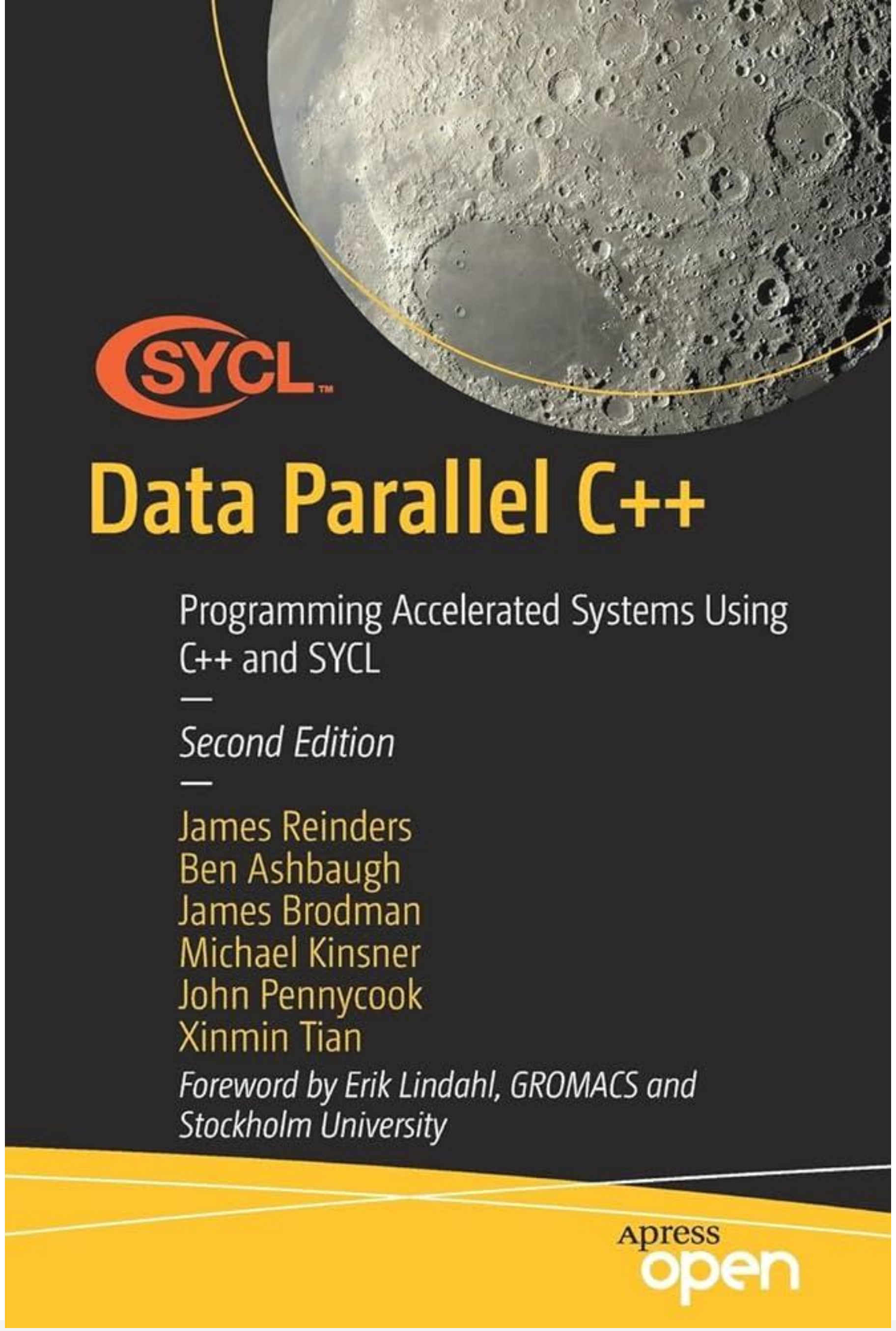
● Open access

Highlights

- Portable programming models can achieve native performance.
- OpenMP, pSTL, and SYCL work very well for writing portable code for GPUs.
- Portable code optimized for Nvidia GH200 may need further optimizations for other GPU vendors.
- Minor code changes can make a big difference.

Learn more about SYCL

- New SYCL book covering Khronos Group SYCL 2020 specification and more
- Heterogeneous programming using C++ with SYCL for CPU, GPU, FPGA, and other accelerators
- From foundations to advanced concepts
- Download a free copy https://link.springer.com/book/10.1007/978-1-4842-9691-2



SYCL™

Data Parallel C++

Programming Accelerated Systems Using C++ and SYCL

—

Second Edition

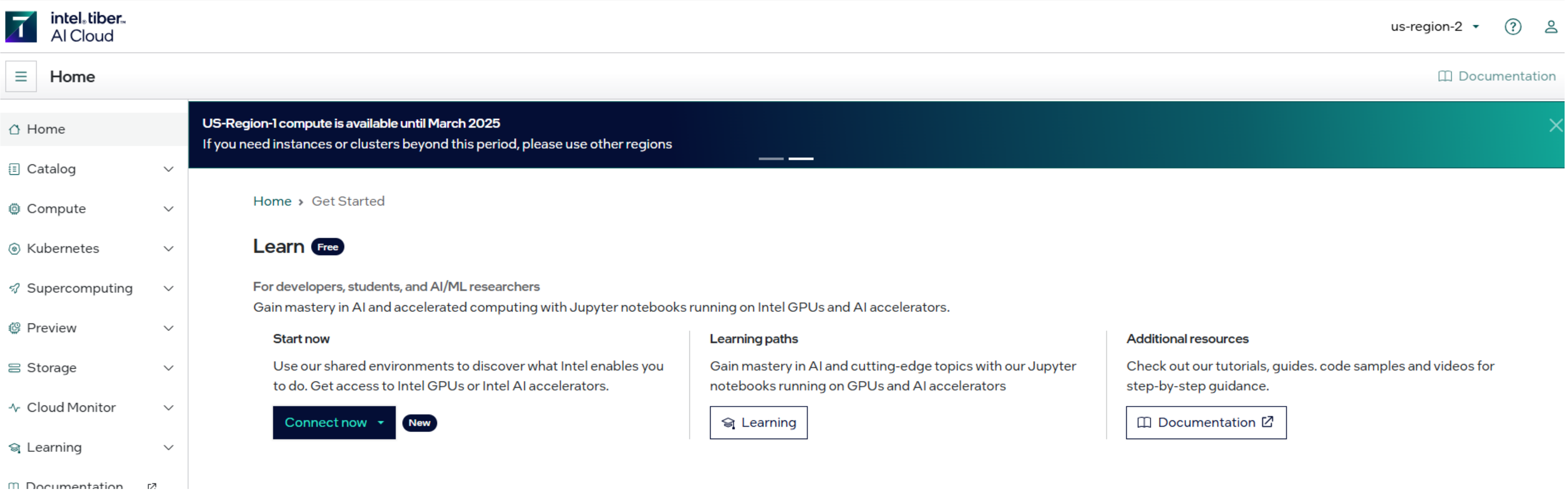
—

James Reinders
Ben Ashbaugh
James Brodman
Michael Kinsner
John Pennycook
Xinmin Tian

Foreword by Erik Lindahl, GROMACS and Stockholm University

Apress open

Intel® Tiber™ Cloud – cloud.intel.com



intel. tiber.
AI Cloud

us-region-2

Home

Documentation

- Home
- Catalog
- Compute
- Kubernetes
- Supercomputing
- Preview
- Storage
- Cloud Monitor
- Learning
- Documentation

US-Region-1 compute is available until March 2025
If you need instances or clusters beyond this period, please use other regions

Home › Get Started

Learn Free

For developers, students, and AI/ML researchers
Gain mastery in AI and accelerated computing with Jupyter notebooks running on Intel GPUs and AI accelerators.

Start now

Use our shared environments to discover what Intel enables you to do. Get access to Intel GPUs or Intel AI accelerators.

Connect now New

Learning paths

Gain mastery in AI and cutting-edge topics with our Jupyter notebooks running on GPUs and AI accelerators

Learning

Additional resources

Check out our tutorials, guides. code samples and videos for step-by-step guidance.

Documentation

intel®
Software

Intel® Compilers

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	Removed 2025
Intel® Fortran Compiler Classic, IL0 *ifort*	CPU	Yes	No	Removed 2025
Intel® Fortran Compiler, LLVM *ifx*	CPU, GPU	Yes	Yes	HPC
Intel® oneAPI DPC++/C++ Compiler, LLVM *icx/icpx*	CPU GPU*	Yes	Yes	Base/HPC

Compiler Binary Compatible and Linkable!

tinyurl.com/oneapi-standalone-components

Common optimization options

	Linux* icx (icc)
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
Profile guided optimization (multi-step build)	-fprofile-generate (-prof-gen) -fprofile-use (-prof-use)
Optimize for speed across the entire program ("prototype switch")	-fast same as "-ipo -O3 -static -fp-model fast" (-ipo -O3 -no-prec-div –static -fp-model fast=2 -xHost)
OpenMP support	-fiopenmp (-qopenmp)

IFX: Driving a New Era in Accelerated Computing

IFX: ALL that you like in IFORT *PLUS*

- OpenMP* 5.x and 6.0 Standards, offload to Intel GPUs from Fortran

  An open, portable Standard maintains your investment

  Best in class OpenMP features and support
- Fortran 2018 DO CONCURRENT supports automatic offload to Intel GPUs

Protecting your Fortran Investment

Same Fortran parser/analyzer you know and love from IFORT

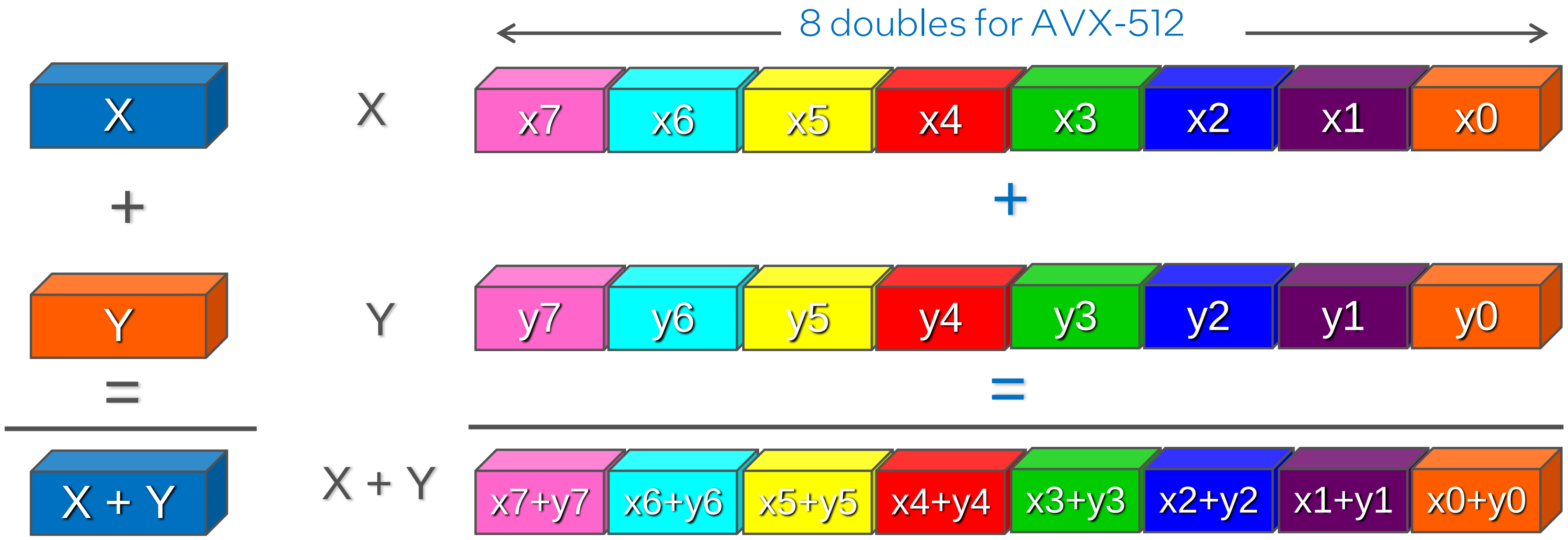
- Supports legacy DEC extensions, ***all F2018,*** ifort directives and features
- The majority of IFORT compiler directives and options you have used for years. And Microsoft Visual Studio* integration for Windows*

Binary compatible, mix and match ifx and ifort

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

- Special switch for icx, 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
- For AMD please check for the GNU flags e.g. **`-march=native`**

Please switch to icx/icpx/ifx Compiler!

- Deprecation of icc/ifort
- Check the user guide for supported flags:

  https://www.intel.com/content/www/us/en/docs/dpcpp-cpp-compiler/developer-guide-reference/2025-0/overview.htm

- Check results and compare with icc/icpc results:

  -fp-model=fast is the default

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

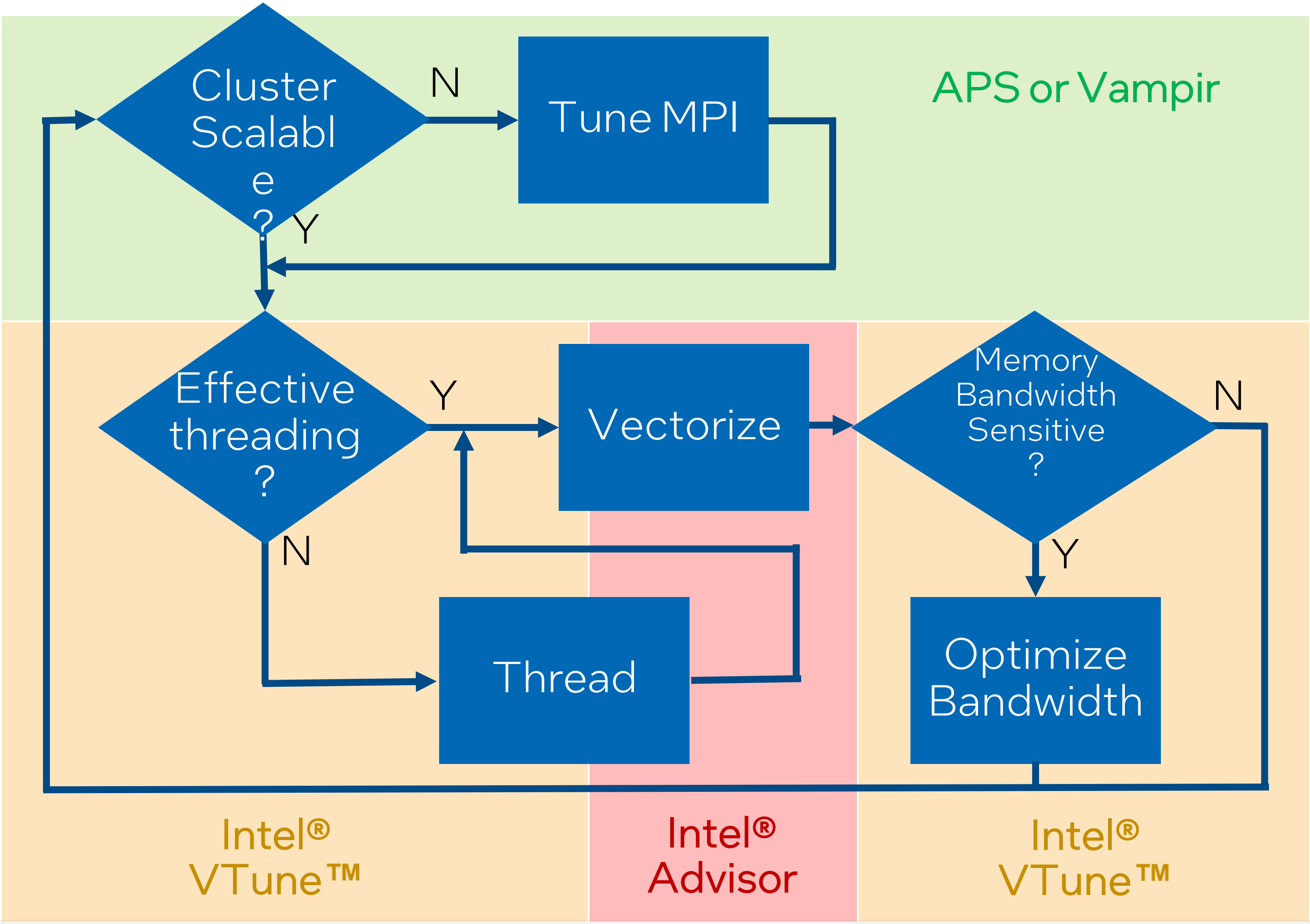
- Good flags to start with: -O2 -xhost

Online Resources

- Conformance to C/C++ standards – link
- GCC compatibility and interoperability – link
- Microsoft Visual C++ compatibility & Visual Studio integration – link
- Porting guide for *icc* users to *icx* – link
- C/C++/SYCL compiler forum – link

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: `$ module load VTune`

APS Usage

Setup Environment

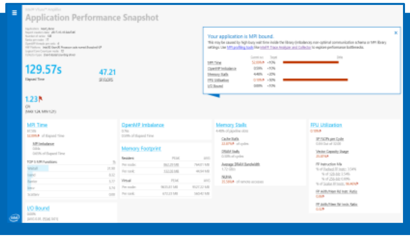
- $ source <path_to_vtune>/vtune_vars.sh # or load module

Run Application

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

Generate Report on Result Folder

- $ aps –report <result folder>



Application Performance Snapshot

129.57s

Elapsed Time

47.21

SP GFLOPS

1.23

CPI

(MAX 1.24, MIN 1.21)

Your application is MPI bound.

MPI Time

OpenMP Imbalance

Memory Stalls

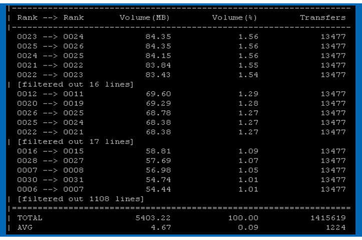
FPU Utilization

I/O Bound

Memory Footprint

Generate CL reports with detailed MPI statistics on Result Folder

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



```
|-----------------------------------------------------------------------
| 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
```

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†

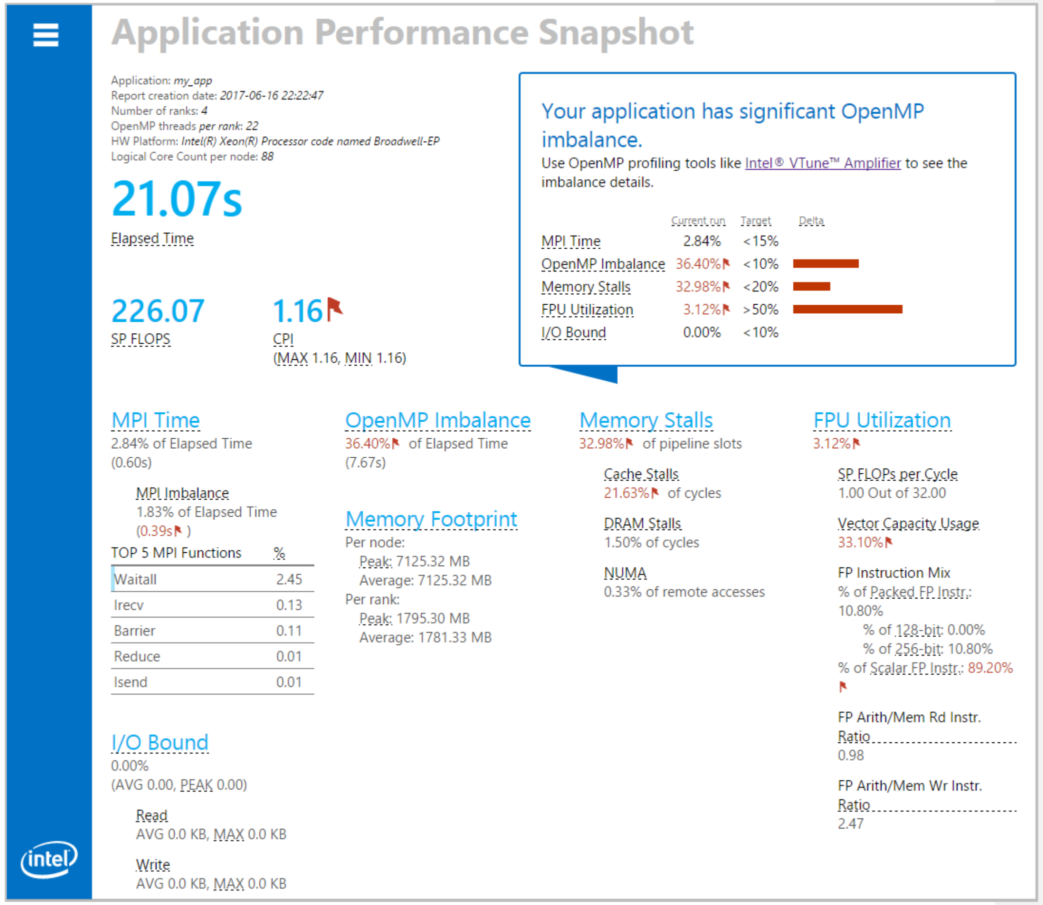
- Distributed MPI with or without threading
- Shared memory applications

Popular MPI Implementations Supported

- Intel® MPI Library
- MPICH & Cray MPI

Richer Metrics on Computation Efficiency

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



Application Performance Snapshot

Application: *my_app*
Report creation date: *2017-06-16 22:22:47*
Number of ranks: *4*
OpenMP threads *per rank*: *22*
HW Platform: *Intel(R) Xeon(R) Processor code named Broadwell-EP*
Logical Core Count per node: *88*

21.07s
Elapsed Time

226.07
SP FLOPS

1.16⚑
CPI
(MAX 1.16, MIN 1.16)

Your application has significant OpenMP imbalance.
Use OpenMP profiling tools like Intel® VTune™ Amplifier to see the imbalance details.

	Current run	Target	Delta
MPI Time	2.84%	<15%	
OpenMP Imbalance	36.40%⚑	<10%	
Memory Stalls	32.98%⚑	<20%	
FPU Utilization	3.12%⚑	>50%	
I/O Bound	0.00%	<10%	

MPI Time
2.84% of Elapsed Time
(0.60s)

MPI Imbalance
1.83% of Elapsed Time
(0.39s⚑)

TOP 5 MPI Functions	%
Waitall	2.45
Irecv	0.13
Barrier	0.11
Reduce	0.01
Isend	0.01

I/O Bound
0.00%
(AVG 0.00, PEAK 0.00)

Read
AVG 0.0 KB, MAX 0.0 KB

Write
AVG 0.0 KB, MAX 0.0 KB

OpenMP Imbalance
36.40%⚑ of Elapsed Time
(7.67s)

Memory Footprint
Per node:
Peak: 7125.32 MB
Average: 7125.32 MB
Per rank:
Peak: 1795.30 MB
Average: 1781.33 MB

Memory Stalls
32.98%⚑ of pipeline slots

Cache Stalls
21.63%⚑ of cycles

DRAM Stalls
1.50% of cycles

NUMA
0.33% of remote accesses

FPU Utilization
3.12%⚑

SP FLOPs per Cycle
1.00 Out of 32.00

Vector Capacity Usage
33.10%⚑

FP Instruction Mix
% of Packed FP Instr.: 10.80%
% of 128-bit: 0.00%
% of 256-bit: 10.80%
% of Scalar FP Instr.: 89.20% ⚑

FP Arith/Mem Rd Instr. Ratio
0.98

FP Arith/Mem Wr Instr. Ratio
2.47

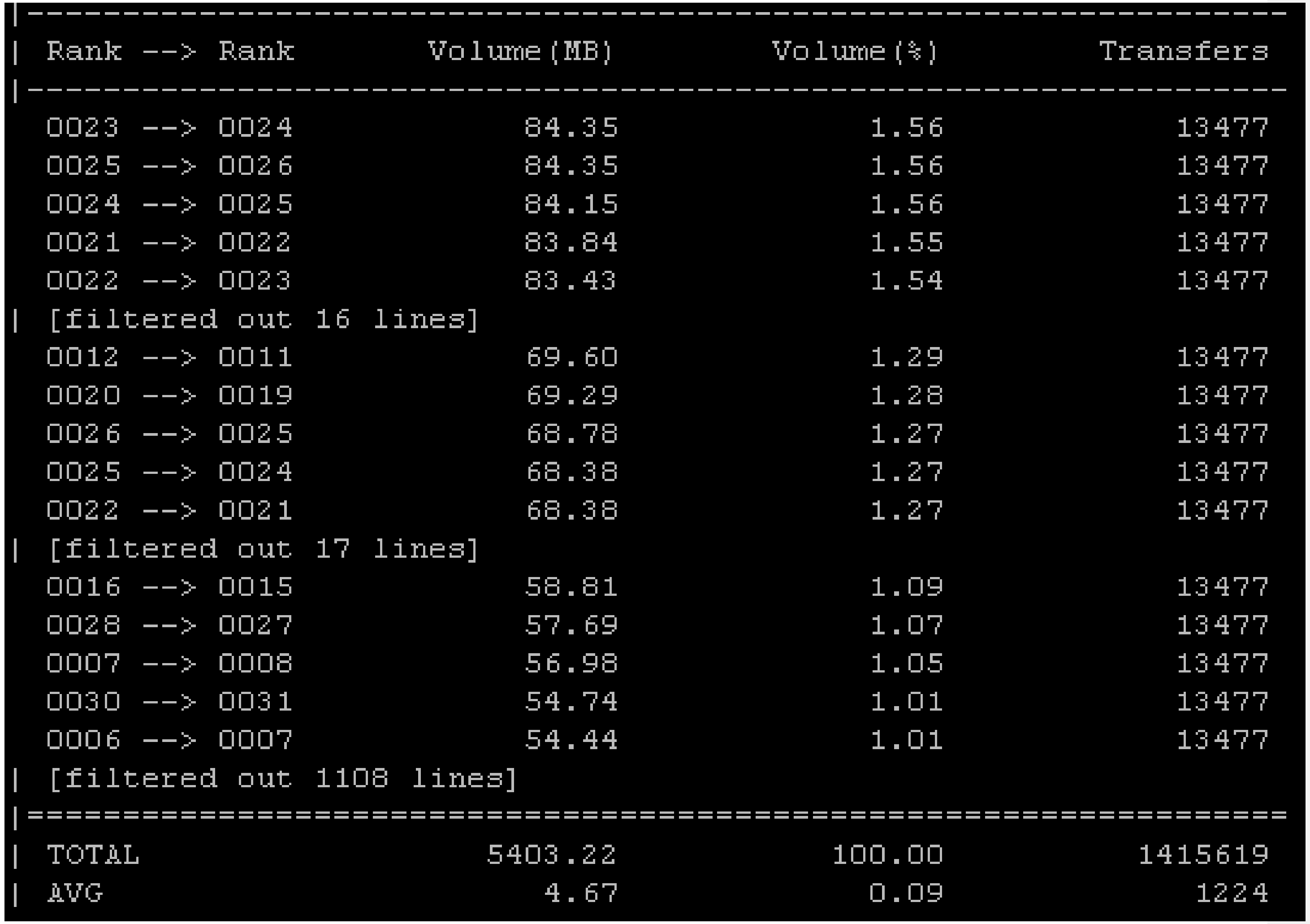
†MPI supported only on Linux*

APS Command Line Reports – Advanced MPI statistics

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

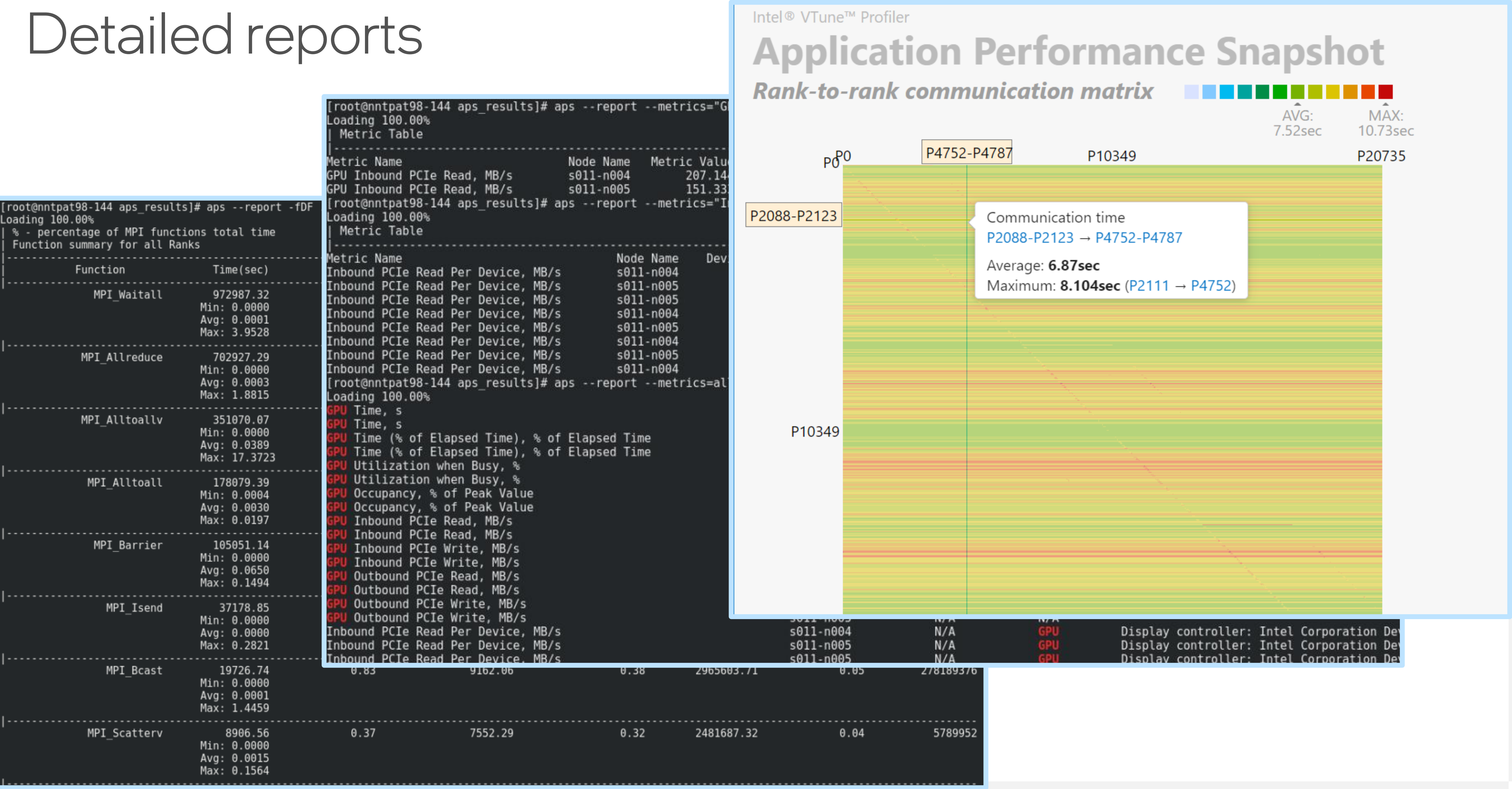
And many others – check

  - 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
```

Detailed reports



```
[root@nntpat98-144 aps_results]# aps --report -fDF
Loading 100.00%
| % - percentage of MPI functions total time
| Function summary for all Ranks
|-----------------------------------------------
|          Function               Time(sec)
|-----------------------------------------------
             MPI_Waitall         972987.32
                              Min: 0.0000
                              Avg: 0.0001
                              Max: 3.9528
|-----------------------------------------------
           MPI_Allreduce         702927.29
                              Min: 0.0000
                              Avg: 0.0003
                              Max: 1.8815
|-----------------------------------------------
           MPI_Alltoallv         351070.07
                              Min: 0.0000
                              Avg: 0.0389
                              Max: 17.3723
|-----------------------------------------------
            MPI_Alltoall         178079.39
                              Min: 0.0004
                              Avg: 0.0030
                              Max: 0.0197
|-----------------------------------------------
             MPI_Barrier         105051.14
                              Min: 0.0000
                              Avg: 0.0650
                              Max: 0.1494
|-----------------------------------------------
               MPI_Isend          37178.85
                              Min: 0.0000
                              Avg: 0.0000
                              Max: 0.2821
|-----------------------------------------------
               MPI_Bcast          19726.74       0.83          9162.06          0.38       2965603.71          0.05        278189376
                              Min: 0.0000
                              Avg: 0.0001
                              Max: 1.4459
|-------------------------------------------------------------------------------------------------------------------------------
            MPI_Scatterv           8906.56       0.37          7552.29          0.32       2481687.32          0.04          5789952
                              Min: 0.0000
                              Avg: 0.0015
                              Max: 0.1564
|-------------------------------------------------------------------------------------------------------------------------------
```

```
[root@nntpat98-144 aps_results]# aps --report --metrics="G
Loading 100.00%
| Metric Table
|----------------------------------------------------------
Metric Name                          Node Name    Metric Valu
GPU Inbound PCIe Read, MB/s          s011-n004         207.14
GPU Inbound PCIe Read, MB/s          s011-n005         151.33
[root@nntpat98-144 aps_results]# aps --report --metrics="I
Loading 100.00%
| Metric Table
|----------------------------------------------------------
Metric Name                                Node Name     Dev
Inbound PCIe Read Per Device, MB/s         s011-n004
Inbound PCIe Read Per Device, MB/s         s011-n005
Inbound PCIe Read Per Device, MB/s         s011-n005
Inbound PCIe Read Per Device, MB/s         s011-n004
Inbound PCIe Read Per Device, MB/s         s011-n005
Inbound PCIe Read Per Device, MB/s         s011-n004
Inbound PCIe Read Per Device, MB/s         s011-n005
Inbound PCIe Read Per Device, MB/s         s011-n004
[root@nntpat98-144 aps_results]# aps --report --metrics=al
Loading 100.00%
GPU Time, s
GPU Time, s
GPU Time (% of Elapsed Time), % of Elapsed Time
GPU Time (% of Elapsed Time), % of Elapsed Time
GPU Utilization when Busy, %
GPU Utilization when Busy, %
GPU Occupancy, % of Peak Value
GPU Occupancy, % of Peak Value
GPU Inbound PCIe Read, MB/s
GPU Inbound PCIe Read, MB/s
GPU Inbound PCIe Write, MB/s
GPU Inbound PCIe Write, MB/s
GPU Outbound PCIe Read, MB/s
GPU Outbound PCIe Read, MB/s
GPU Outbound PCIe Write, MB/s
GPU Outbound PCIe Write, MB/s
Inbound PCIe Read Per Device, MB/s                      s011-n004      N/A      GPU      Display controller: Intel Corporation De
Inbound PCIe Read Per Device, MB/s                      s011-n005      N/A      GPU      Display controller: Intel Corporation De
Inbound PCIe Read Per Device, MB/s                      s011-n005      N/A      GPU      Display controller: Intel Corporation De
```

Intel® VTune™ Profiler

Application Performance Snapshot

Rank-to-rank communication matrix

Communication time
P2088-P2123 → P4752-P4787
Average: **6.87sec**
Maximum: **8.104sec** (P2111 → P4752)

APS potential issues

- If drivers are not available and for other hardware like AMD, you may use only mpi or/and openMP analysis (for Intel MPI):

```
$ aps --collection-mode=mpi[,openmp]
```

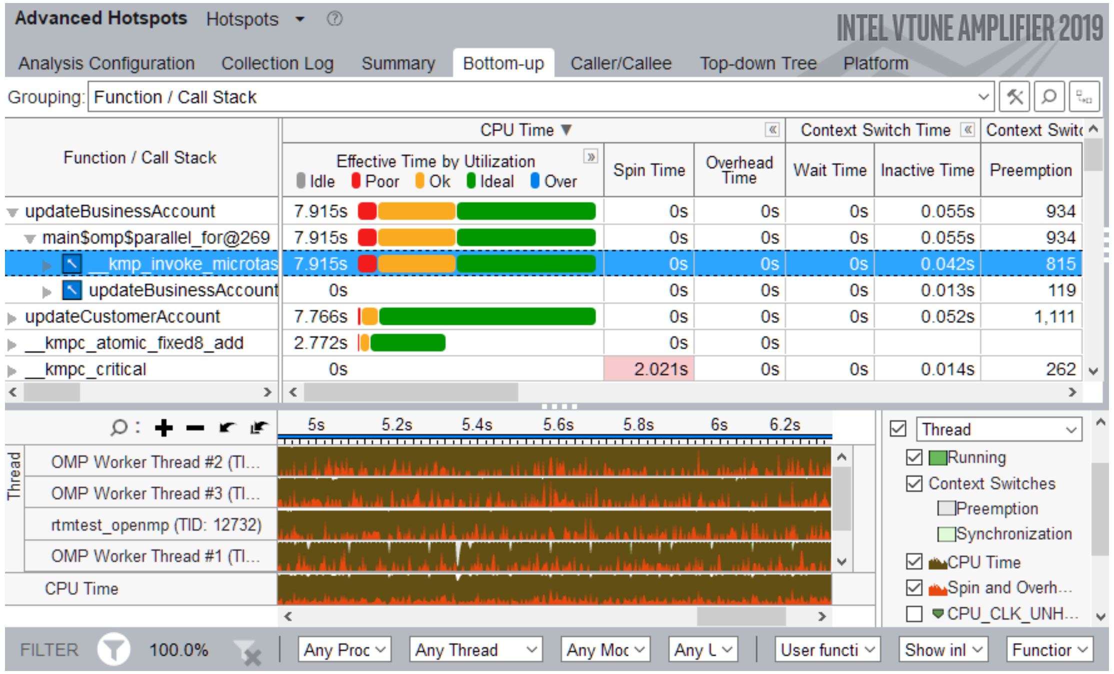
- If drivers are needed,you have to add this line to your batch job:

```
#SBATCH --disable-perfparanoid
```

Intel® VTune™ Profiler

Analyze & Tune Application Performance

Intel® VTune™ Profiler



Grouping: Function / Call Stack

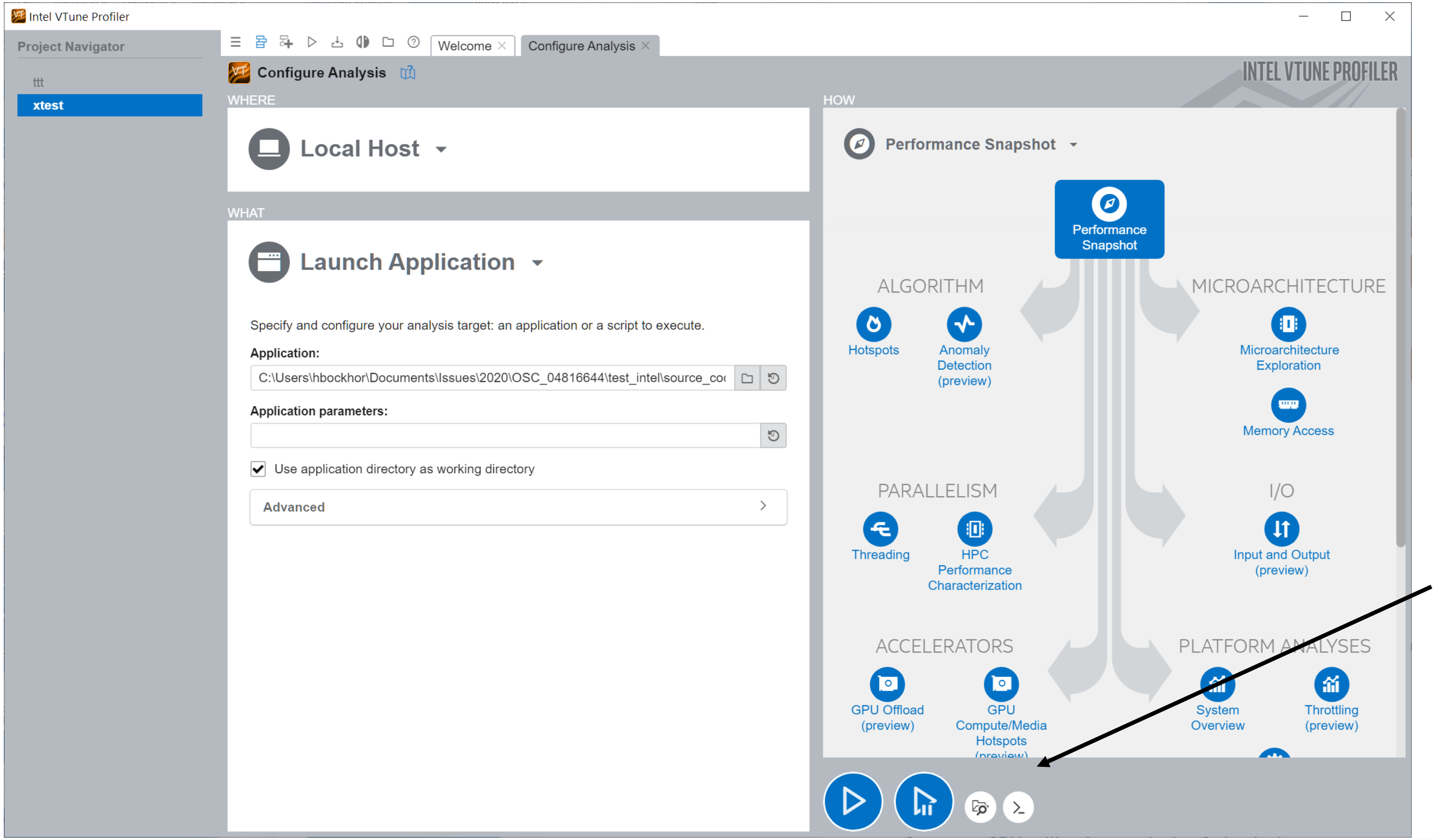
Function / Call Stack	CPU Time ▼			Context Switch Time		Context Switc
	Effective Time by Utilization Idle Poor Ok Ideal Over	Spin Time	Overhead Time	Wait Time	Inactive Time	Preemption
updateBusinessAccount	7.915s	0s	0s	0s	0.055s	934
main$omp$parallel_for@269	7.915s	0s	0s	0s	0.055s	934
__kmp_invoke_microtas	7.915s	0s	0s	0s	0.042s	815
updateBusinessAccount	0s	0s	0s	0s	0.013s	119
updateCustomerAccount	7.766s	0s	0s	0s	0.052s	1,111
__kmpc_atomic_fixed8_add	2.772s	0s	0s			
__kmpc_critical	0s	2.021s	0s	0s	0.014s	262

FILTER 100.0% Any Proc Any Thread Any Moc Any L User functi Show inl Functior

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

- 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*)

Start a new Project



Intel VTune Profiler

Project Navigator

ttt

xtest

Welcome

Configure Analysis

Configure Analysis

INTEL VTUNE PROFILER

WHERE

Local Host

WHAT

Launch Application

Specify and configure your analysis target: an application or a script to execute.

Application: C:\Users\hbockhor\Documents\Issues\2020\OSC_04816644\test_intel\source_co

Application parameters:

☑ Use application directory as working directory

Advanced

HOW

Performance Snapshot

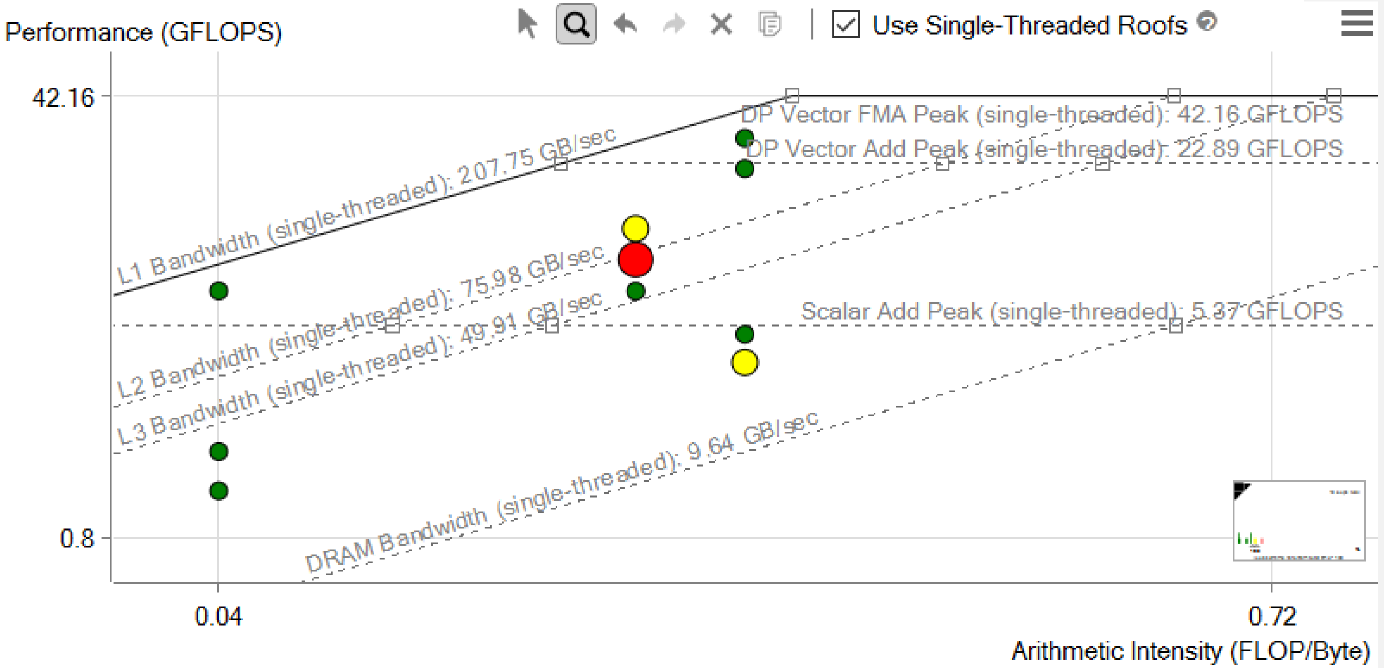
- Use GUI
- Or Command-Line

Get Command-Line

INTEL® ADVISOR

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?



☑ Use Single-Threaded Roofs

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 + VTune 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

Additional Development Products

- Intel® oneAPI Toolkits

INTEL® MKL
INTEL® MPI

Intel oneMKL (Math kernel libraries)

- https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html
- Math processing routines on CPU and GPU using C++, Fortran and python
- HPC and AI

Intel MPI

- https://www.intel.com/content/www/us/en/developer/tools/oneapi/mpi-library.html

- Is (Intel) GPU aware using device memory.

Intel Modules installed on Juwels/Jureca

▪ 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

intel®

How to start?

- Compile with minimal options and run with APS (will provide tuning tips)
- Compile with -O2 -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> and create a roofline report.
- 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

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**