Performance Optimization for Software Developers with oneAPI & Parallelization on Intel® Architecture

Edmund Preiss Business Development Manager

Software and Advanced Technology Group (SATG)



#### NOTICES AND DISCLAIMERS

Refer to https://software.intel.com/en-us/articles/optimization-notice for more information regarding performance and optimization choices in Intel software products.

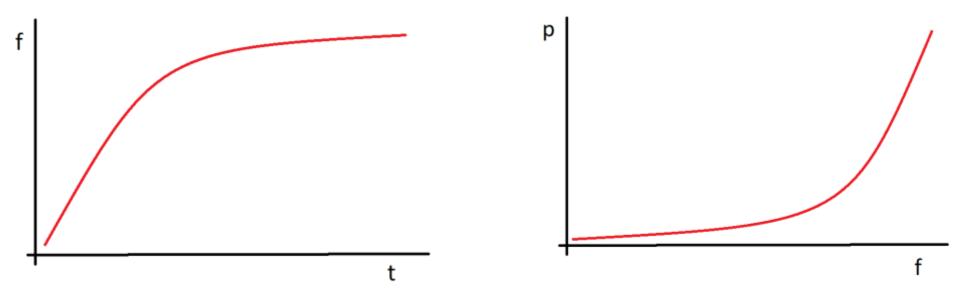
Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No product or component can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

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

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others. © Intel Corporation

### Welcome in the Parallelism Area !

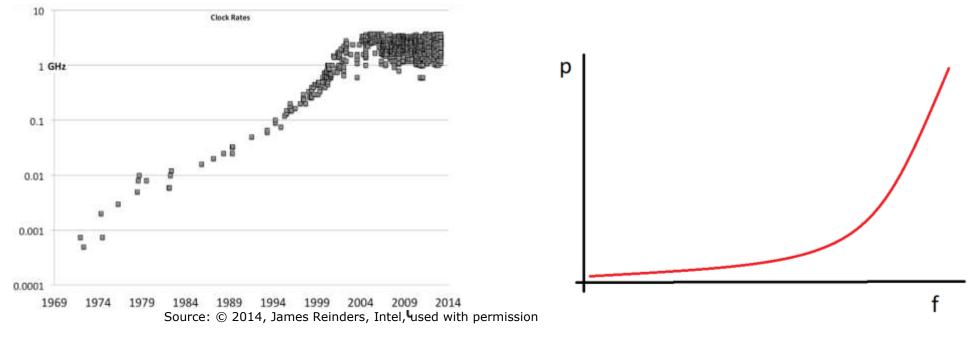


Microprocessor frequency over Time (history)

Microprocessor frequency versus Power consumption

... performance is not only the computer architect's job anymore ... performance increase is increasingly the job of the software developer

### Welcome in the Parallelism Area !



Microprocessor frequency over Time (history) Microprocessor frequency versus Power consumption

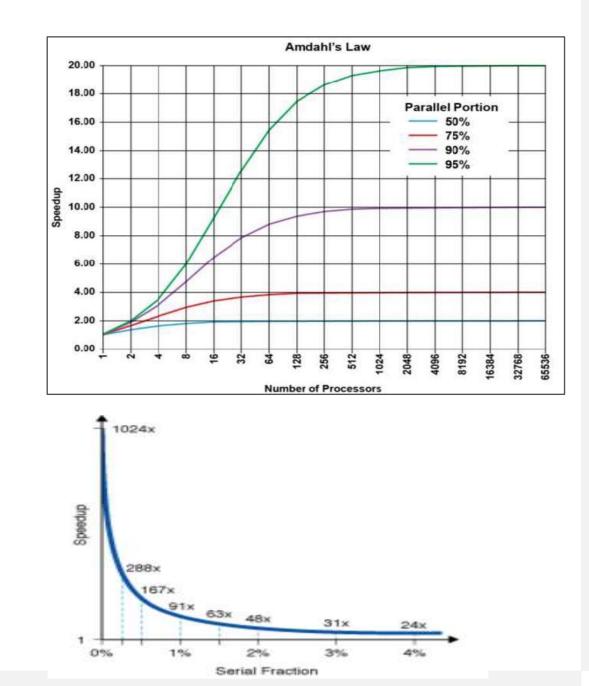
... performance is not only the computer architect's job anymore ... performance increase is increasingly the job of the software developer

### Amdahl's Law

"The speedup of a program using multiple processors in parallel computing is limited by the sequential fraction of the program."

- Gene Amdahl

Speedup = 
$$(s + p)/(s + p/N)$$
  
=  $1/(s + p/N)$ 

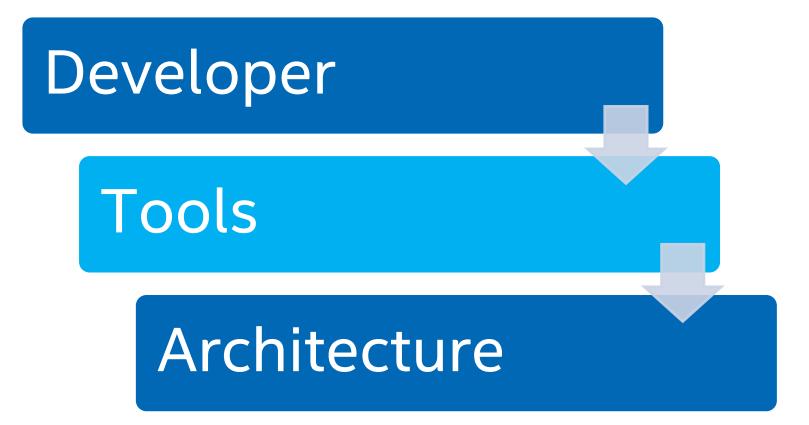


### The 'Free Lunch' is over

# "Parallelism == > Performance" (leads to)

# **Optimization** – Developer to make sure the above statement becomes reality!

### Parallelism on the Intel Architecture



### Code Optimization Principles

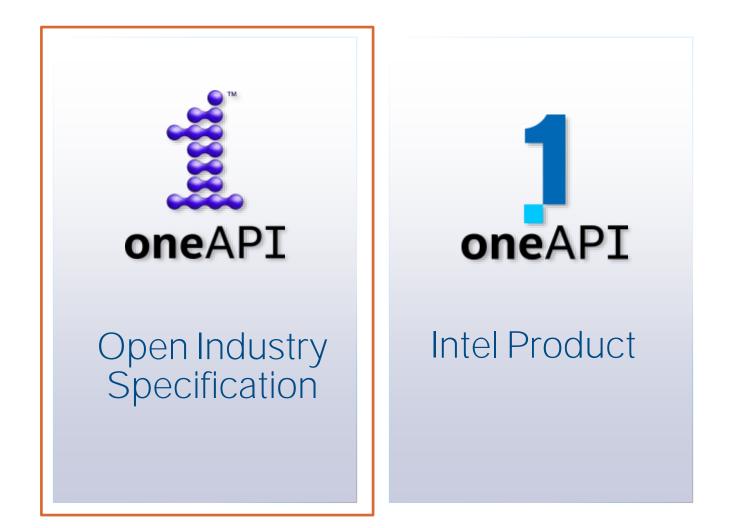
Stage 1: Use Optimized Libraries

Stage 2: Compile with Architecture-specific Optimizations

Stage 3: Analysis and Tuning

Stage 4: Check Correctness

### oneAPI – A Tools Development Framework



# Programming Challenges

for Multiple Architectures

Growth in specialized workloads

Variety of data-centric hardware required

Requires separate programming models and toolchains for each architecture

Software development complexity limits freedom of architectural choice

Applicat	Application Workloads Need Diverse Hardware						
Scalar	Vector	Spatial	Matrix				
	Middleware &	Frameworks					
CPU programming model	GPU programming model	FPGA programming model	Other accel. programming models				
model							
-t							
ø	o	D	١				
CPU	GPU	FPGA	Other accel.				

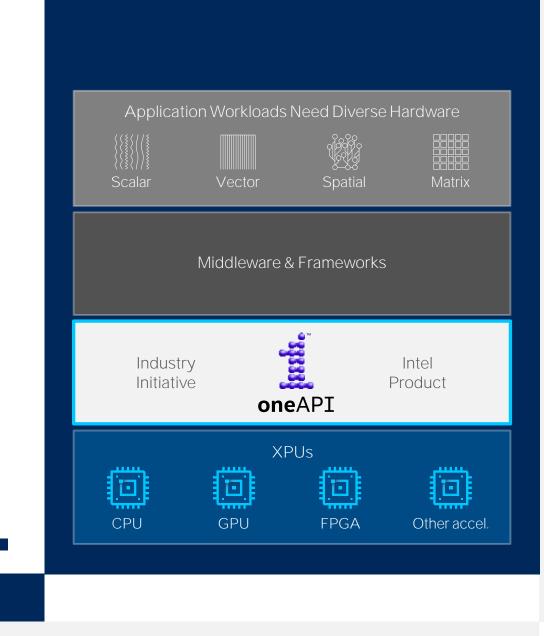
# Introducing oneAPI

Cross-architecture programming that delivers freedom to choose the best hardware

Based on industry standards and open specifications

Exposes cutting-edge performance features of latest hardware

Compatible with existing high-performance languages and programming models including C++, OpenMP, Fortran, and MPI



# oneAPI Industry Initiative

Break the Chains of Proprietary Lock-in

A cross-architecture language based on C++ and SYCL standards

Powerful libraries designed for acceleration of domainspecific functions

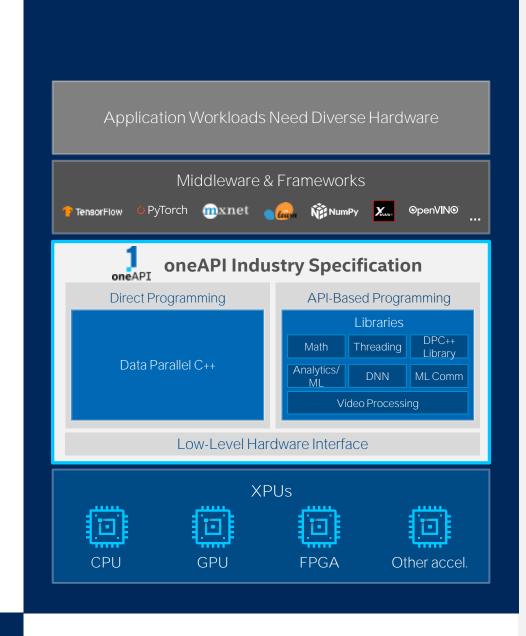
Low-level hardware abstraction layer

Open to promote community and industry collaboration

Enables code reuse across architectures and vendors



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



# Data Parallel C++

#### DPC++ = ISO C++ and Khronos SYCL

### Parallelism, productivity, and performance for CPUs and accelerators

- Delivers accelerated computing by exposing hardware features
- Allows code reuse across hardware targets, while permitting custom tuning for specific accelerators
- Provides an open, cross-industry solution to single-architecture proprietary lock-in

#### Based on C++ and SYCL

- Delivers C++ productivity benefits, using common, familiar C and C++ constructs
- Incorporates SYCL from the Khronos Group to support data parallelism and heterogeneous programming

#### Community Project to drive language enhancements

- Provides extensions to simplify data parallel programming
- Continues evolution through open and cooperative development

#### Apply your skills to the next innovation, not to rewriting software for the next hardware platform

#### Standards-based, Cross-architectural Language

Direct Programming: Data Parallel C++

Community Extensions

Khronos SYCL

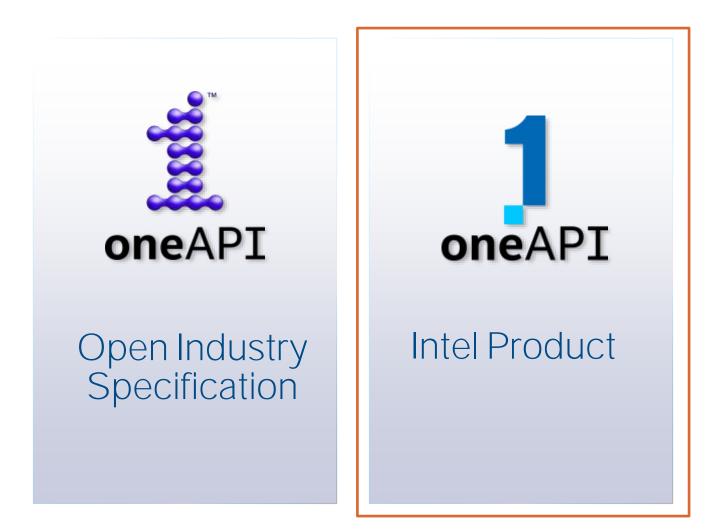
ISO C++

# Powerful oneAPI Libraries

- Designed for acceleration of key domain-specific functions
- Pre-optimized for each target platform for maximum performance

Intel® oneAPI Math Kernel Library oneMKL	Intel <sup>®</sup> oneAPI Deep Neural Network Library oneDNN
Intel <sup>®</sup> oneAPI Video Processing Library oneVPL	Intel <sup>®</sup> oneAPI Data Analytics Library oneDAL
Intel <sup>®</sup> oneAPI Threading Building Blocks oneTBB	Intel <sup>®</sup> oneAPI Collective Communications Library oneCCL
Intel® oneAPI DPC++ Library oneDPL	

### oneAPI – A Tools Development Framework

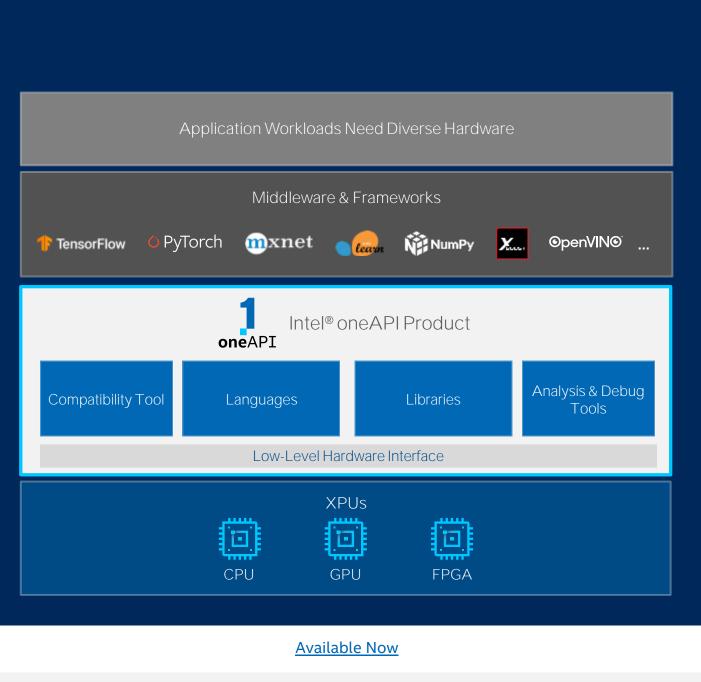


# Intel<sup>®</sup> oneAPI Product

Built on a Rich Heritage of Tools Based on Intel<sup>®</sup> Xeon<sup>®</sup> Processors, Now Expanded to XPUs

A complete set of advanced compilers, libraries, and porting, analysis & debugger tools

- Accelerates compute by exploiting cutting-edge hardware features
- Interoperable with existing programming models and code bases (C++, Fortran, Python, OpenMP, etc.), developers can be confident that existing applications work seamlessly with oneAPI
- Eases transitions to new systems and accelerators—using a single code base frees developers to invest more time on innovation



# Intel Compiler Transition: Classic to LLVM

#### Start Your Migration Now

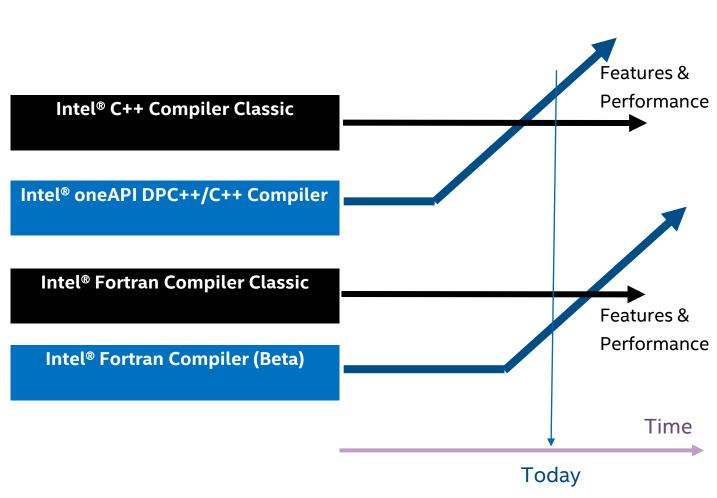
- Expand to XPUs
- Modern LLVM Infrastructure

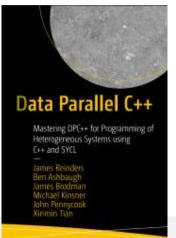
#### Intel<sup>®</sup> oneAPI DPC++/C++ Compiler

- Use for all new projects
- Migrate legacy projects

#### Intel<sup>®</sup> Fortran Compiler (Beta)

- Test drive now & Provide feedback
- Prepare for migration



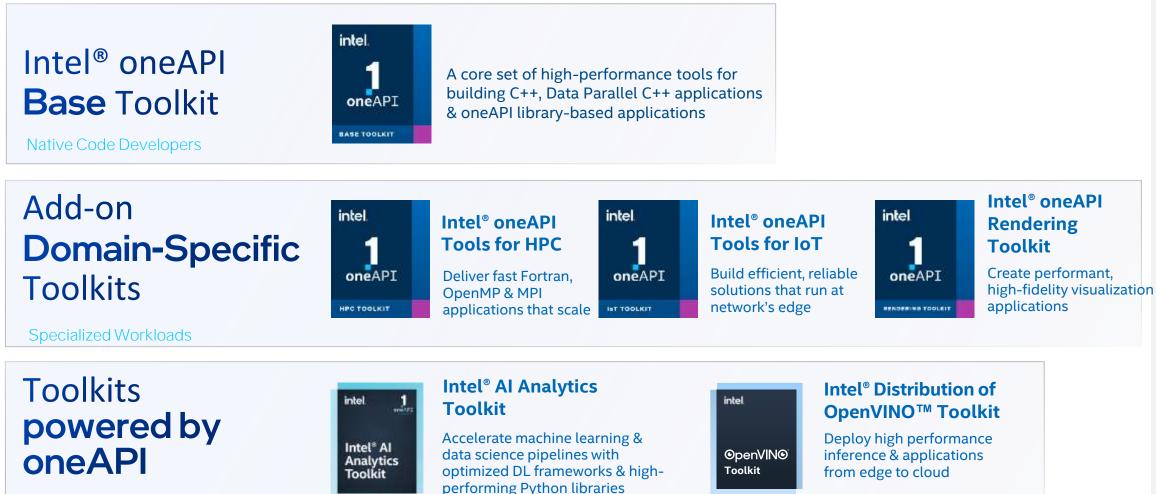


#### Performance, Quality, and Support Continues

### Intel<sup>®</sup> oneAPI Toolkits

A Complete Set of Proven Developer Tools Expanded from CPU to XPU





Data Scientists & AI Developers

### Streamlining Product Line

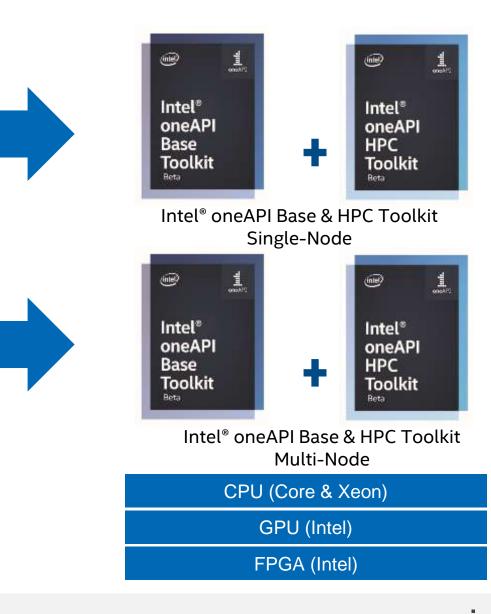


Professional Edition

(intel)

**STUDIO XE** 

CPU Focused (Core & Xeon)



#### Intel<sup>®</sup> oneAPI Base & HPC Toolkit

Intel<sup>®</sup> oneAPI Tools for HPC: Deliver Fast Applications that Scale

#### What is it?

A toolkit that adds to the Intel<sup>®</sup> oneAPI Base Toolkit for building high-performance, scalable parallel code on C++, Fortran, OpenMP & MPI from enterprise to cloud, and HPC to AI applications.

#### Who needs this product?

- OEMs/ISVs
- C++, Fortran, OpenMP, MPI Developers

#### Why is this important?

- Accelerate performance on Intel<sup>®</sup> Xeon<sup>®</sup> & Core<sup>™</sup> Processors and Accelerators
- Deliver fast, scalable, reliable parallel code with less effort; built on industry standards

#### Intel® oneAPI Base & HPC Toolkit

Direct Programming	API-Based Programming	Analysis & debug Tools
Intel® C++ Compiler Classic	Intel® MPI Library	Intel <sup>®</sup> Inspector
Intel <sup>®</sup> Fortran Compiler Classic	Intel® oneAPI DPC++ Library	Intel® Trace Analyzer & Collector
Intel® Fortran Compiler (Beta)	Intel® oneAPI Math Kernel Library	Intel® Cluster Checker
Intel® oneAPI DPC++/C++ Compiler	Intel® oneAPI Data Analytics Library	Intel® VTune™ Profiler
Intel® DPC++ Compatibility Tool	Intel <sup>®</sup> oneAPI Threading Building Blocks	Intel <sup>®</sup> Advisor
Intel® Distribution for Python*	Intel <sup>®</sup> oneAPI Video Processing Library	Intel <sup>®</sup> Distribution for GDB*
Intel® FPGA Add-on for oneAPI Base Toolkit	Intel <sup>®</sup> oneAPI Collective Communications Library	
	Intel® oneAPI Deep Neural Network Library	intel.
Intel® oneAPI <b>HPC</b> Toolkit +	Intel <sup>®</sup> Integrated Performance Primitives	

#### Render Your Vision in Highest Fidelity Intel<sup>®</sup> oneAPI Rendering Toolkit

### Powerful Libraries for High-Fidelity Visualization Applications

- Deliver high-performance, high-fidelity visualization applications on Intel<sup>®</sup> architecture
- Create amazing visual, hyper-realistic renderings via ray tracing with global illumination
- Access all system memory space to create renderings using the largest data sets
- Flexible, cost efficient development using open source libraries

Intel® Embree, part of Intel® oneAPI Rendering Toolkit, won an Academy Award® Technical Achievement Award in 2021



intel

<sup>1</sup> Avengers: Infinity War - Digital Domain, Marvel Studios, Chaos Group V-Ray

<sup>2</sup> Scene courtesy of Frank Meinl

<sup>3</sup> Model from Leigh Orf at University of Wisconsin. For more tornado visualization, visit Leigh Orf's site <sup>4</sup> Smoke volume, data courtesy OpenVDB example repository

<sup>5</sup> Moana Island Scene, Walt Disney Animation Studios , publicly available dataset: 15fps+,~160 billion prims

#### Intel oneAPI Rendering & Ray Tracing Libraries

#### Intel<sup>®</sup> Embree

High-Performance, Feature-Rich Ray Tracing & Photorealistic Rendering

#### Intel<sup>®</sup> Open Image Denoise

AI-Accelerated Denoiser for Superior Visual Quality

#### Intel<sup>®</sup> OpenSWR

High-Performance, Scalable, OpenGL\*-Compatible Rasterizer

#### Intel® Open Volume Kernel Library

Render & Simulate 3D Spatial Data Processing

Intel<sup>®</sup> OSPRay Scalable, Portable, Distributed Rendering API

Intel<sup>®</sup> OSPRay Studio Real-time rendering through a graphical user interface with this new scene graph application

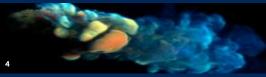
#### Intel<sup>®</sup> OSPRay for Hydra

Connect the Rendering Toolkit libraries to Universal Scene Description Hydra Rendering subsystem via plugin











#### Leafn More: intel.com/oneAPI-RenderKit

### Intel<sup>®</sup> oneAPI AI Analytics Toolkit

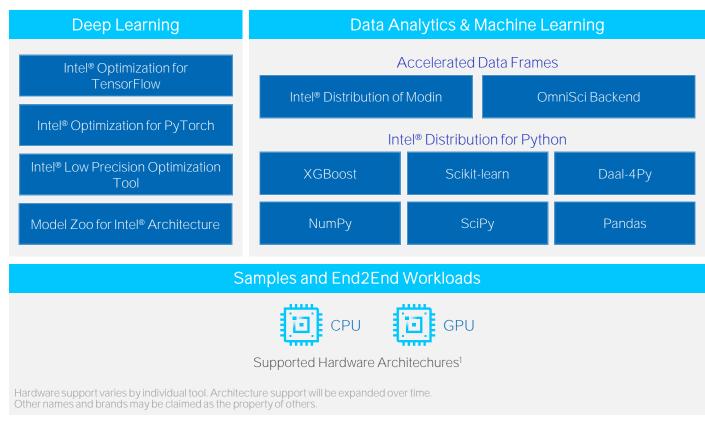
Accelerate end-to-end AI and data analytics pipelines with libraries optimized for Intel® architectures

#### Who Uses It?

Data scientists, AI researchers, ML and DL developers, AI application developers

#### Top Features/Benefits

- Deep learning performance for training and inference with Intel optimized DL frameworks and tools
- Drop-in acceleration for data analytics and machine learning workflows with compute-intensive Python packages







Benchmark

# Intel<sup>®</sup> Distribution of OpenVINO<sup>™</sup> toolkit

Calibration

Powered by oneAPI

#### Deliver High-Performance Deep Learning Inference

A toolkit to accelerate development of high-performance deep learning inference & computer vision in vision/AI applications used from edge to cloud. It enables deep learning on hardware accelerators & easy deployment across Intel® CPUs, GPUs, FPGAs, VPUs.

#### Who needs this product?

- Computer vision, deep learning software developers
- Data scientists
- OEMs, ISVs, System Integrators

#### Usages

Security surveillance, robotics, retail, healthcare, AI, office automation, transportation, non-vision use cases (speech, NLP, Audio, text) & more



#### **Deep Learning Traditional Computer Vision** Optimized Libraries & Code Samples Intel<sup>®</sup> Deep Learning Deployment Toolkit Optimized Inference Convert & Optimize For Intel CPU & GPU/Intel® Processor Graphics IR = Intermediate Representation file **Tools & Libraries** Open Model Zoo Increase Media/Video/Graphics Performance 40+ Model Samples Pretrained OpenCL<sup>™</sup> Drivers Intel<sup>®</sup> Media SDK Downloader Models & Runtimes **Open Source Version** For GPU/Intel<sup>®</sup> Processor Graphics Deep Learning Workbench

#### Intel<sup>®</sup> Distribution of OpenVINO<sup>™</sup> toolkit

Optimize Intel<sup>®</sup> FPGA (Linux<sup>\*</sup> only)

PGA RunTime Environment

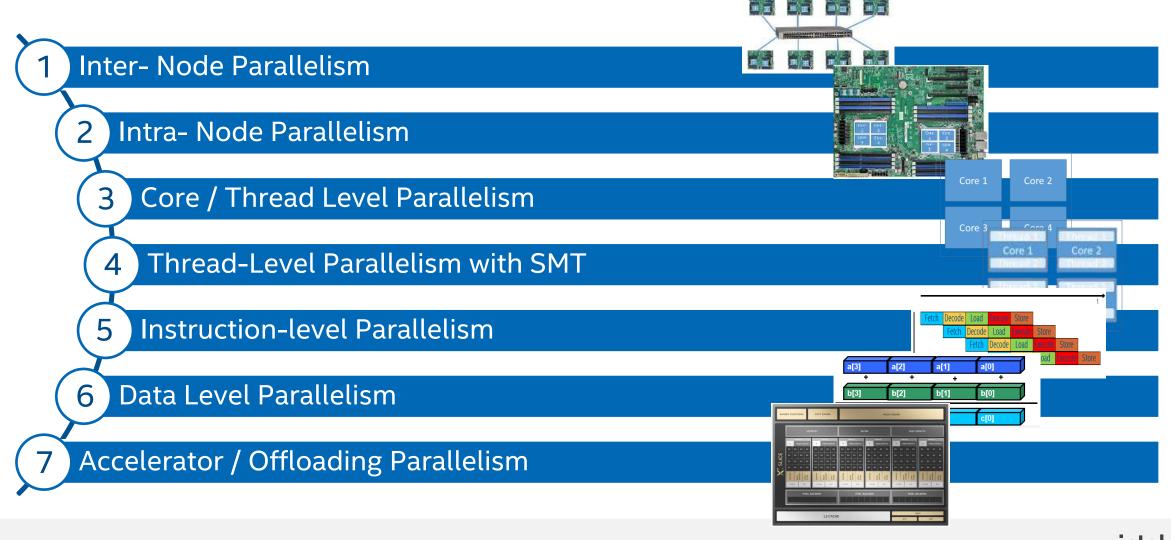
(from Intel" FPGA SDK for OpenCL<sup>™</sup>)

OpenVINO<sup>®</sup>

Sample

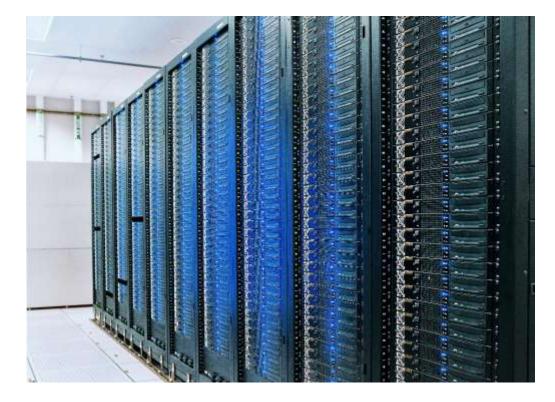
# Parallelism on the Intel Architecture Enabled by Intel® oneAPI Products

### The Seven Levels of Parallelism



Inter- Node Parallelism					
			T		
1 Inter- Node Parallelism		A CONTRACTOR	TRACTICAL CONTRACT		
2		<b>R H</b>	-	HA	
3	192 Canalis	PRO THINKS	Recting.	, FREITHING .	
4					
5					
6					
$\overline{7}$					

### Inter- Node Parallelism



- Most common scenario of HPC / Cluster applications
- Nodes connected by fast interconnect called fabric
- Partitioned Global Address Space (PGAS) & Distributed Memory Programming
- Message Passing Interface (MPI)
   most common approach

# Inter- Node Parallelism

What can I do?

- employ communicationavoiding algorithms (e.g. neighborhood collectives)
- overlap compute and communication where possible
- Load balance work between ranks
- In case of well scaling application, add more nodes ③

Which Tools?

- Identify scalability issues with the VTune<sup>™</sup> Profiler Application Performance Snapshot & the Trace Analyzer and Collector
- Fine Tune Collective Operations with Intel MPI – autotuner
- Where applicable
  - Math Kernel Library (oneMKL)
  - Data Analytics Library (oneDAL)
  - Collective Communications Library (oneCCL)

### Inter- Node Parallelism

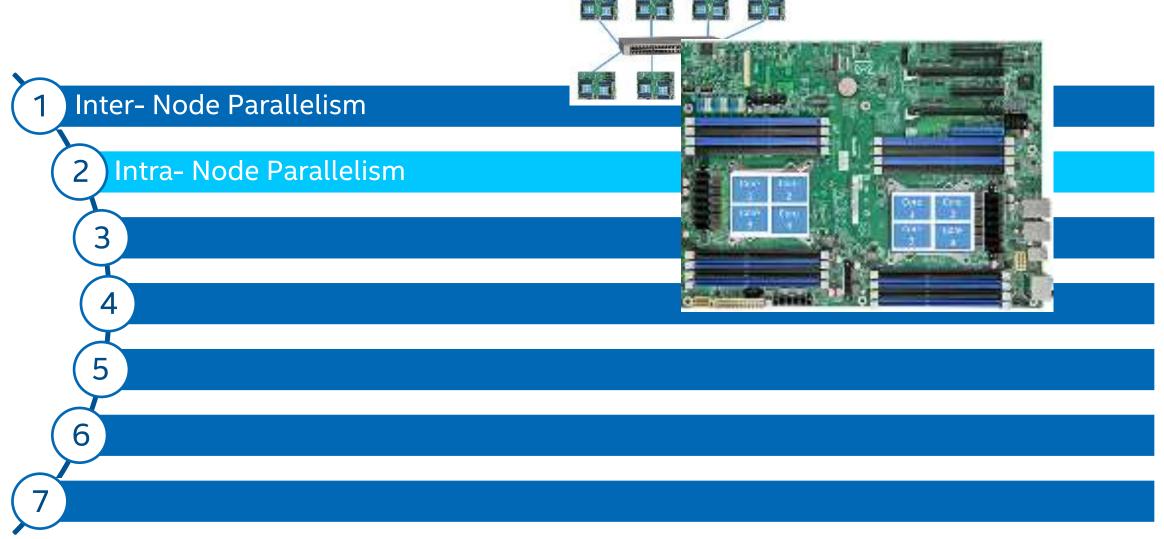
#### VTune HPC Analysis (right)

#### Application Performance Snapshots HTML export (left)

Abburganou Le	rformance Snapsh	0		
Annual Angle	2.05 M	0.00 35.09 matuats terminae	Vour application is NPT bound. The way to careful to the transmission of the second se	ny polosiana, no zaland (omunator strene o tel (bay elling) (be (d polos polonikas latinosis) jan
ANTI Time T	All         Classifier           1         Second           1         Second	instport	Mirrory Lubb Liber of permenent Control State of the Control State of the Control State State of the Control State State State of the Control State St	Wortschaften           State * Stranger * (som state)           Barger           Barger

Import Conference 10 Base of Time 11 51.4466       Service 21 Colored Time 11 51.4466         Import Conference 10 Service 21 Colored Time 11 Colored Time 1	HPC Performance Character	taation HPC Perform	mon Characterization	• @		INTEL	VTUNE AMPLIFIER 2	
PULLOS 1: 0000 DP (LOS) 5: 101 DP (LOS) 5:	Analysis Configuration Collection	ing Surroury Bette	muz Caller/Calles	Jup dawn Due		19		
Concerning and a set of constant of the set of the	SP GELOPS	0,000						
Average CPU Programs <sup>1</sup> 5 2 GHz That These Counts 90 9 9 9 9 9 9 9 9 9 9 9 9 9								
Statist These Courts 10 Set Effective Physical Core Utilization <sup>10</sup> : 25.6% (11.268 out of 44) * Effective Lagoal Core Utilization <sup>10</sup> : 25.6% (11.268 out of 44) * Effective Lagoal Core Utilization <sup>10</sup> : 25.6% (11.268 out of 44) * Endet Lagoal Core Utilization <sup>10</sup> : 25.6% (11.268 out of 44) * Endet Lagoal Core Utilization <sup>10</sup> : 25.6% (11.268 out of 44) * Endet Lagoal Core Utilization <sup>10</sup> : 25.6% (11.268 out of 44) * Endet Lagoal Core Utilization <sup>10</sup> : 25.6% (11.268 out of 44) * Endet Lagoal Core Utilization <sup>10</sup> : 25.6% (11.268 out of 44) * Endet Lagoal Core Utilization <sup>10</sup> : 25.6% (11.268 out of 44) * Endet Lagoal Core Utilization <sup>10</sup> : 25.6% (11.268 out of 44) * Endet Lagoal Core Utilization <sup>10</sup> : 25.6% (11.268 out of 44) * Endet Lagoal Core Utilization <sup>10</sup> : 25.6% (11.268 out of 44) * Endet Lagoal Core Utilization <sup>10</sup> : 26.7% (11.268 out of 44) * Endet Lagoal Core Utilization <sup>10</sup> : 26.7% (11.268 out of 44) * Endet Lagoal Core Utilization <sup>10</sup> : 26.7% (11.268 out of 44) * Endet Lagoal Core Utilization <sup>10</sup> : 26.7% (11.268 out of 44) * Endet Lagoal Core Utilization Lagoal Mathematic approximation approximation of the opport	CP1 Rate	1.485 *						
<ul> <li>Stetctive Physical Core Utilization <sup>1</sup>/<sub>1</sub>: 25.65% (11.268 out of 44) <sup>K</sup> Extend that a particle applies the <sup>1</sup>/<sub>2</sub> and <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 44) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub>2</sub>: 25.6% (12.268 out of 46) <sup>K</sup> Extend that provide applies the <sup>1</sup>/<sub></sub></li></ul>								
Piretries Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (22, 347 or) at 81 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (22, 347 or) at 81 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (22, 347 or) at 81 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (22, 347 or) at 81 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (22, 347 or) at 81 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (22, 347 or) at 81 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie CPU Utilization Hangent  Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilizati	Total Thread Count:	90						
Piretries Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (22, 347 or) at 81 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (22, 347 or) at 81 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (22, 347 or) at 81 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (22, 347 or) at 81 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (22, 347 or) at 81 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (22, 347 or) at 81 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie Layer Core Utilization <sup>1</sup> / <sub>1</sub> , 254 (*)  Piretrie CPU Utilization Hangent  Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilization Hangent Piretrie CPU Utilizati	Effective Physical C	ore Utilization	25.6% (11.268)	out of 44) 🎮				
Solid The found is parted if a given if a 2735 to 339 Provide Region Taxe *: a 47576 (MATRIX) Details of Region Taxe *: a 47576 (MATRIX) This section has DisordPredication *: a 47476 (MATRIX) This section for a 444 (MATRIX) MATRIX #: a 44								
Entratured Mark Time () 15.478 (2009) Gen 2017 Regions for Veterinta Cala The section has Don-MP Regions with the lightest potential for performance improvement. The Peterintal Cale metric shows the elapsed time that region was optimized to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in Entrative constraints of the instruction was provided to have in  Entrative constraints of the instruction was provided to have in  Entrative constraints of the instruction was provided to have in  Entrative constraints of the instruction was provided to have in  Entrative constraints of the instruction was provided to have in  Entrative constraints of the instruction was provided to have in  Entrative constraints of the instruction was provided to have instruction was provided to have instruction  Entrative constraints of the instruction  Entrative constrain	the second se							
GreatWP Recent Case The State S	O Parabel Region Time <sup>(1)</sup> 48	8951(94.7%)						
The OperAMP Regions the Table Affect of August and the Algebra to bacterial for performance improvement. The Potential Gain metric shows the elagebra time back be used if the subject of and the algebra to back the algebra time of the algebra to back the algebra time of the algebra								
This section that OperAPP regions with the Nathen potential for performance improvement. The Petential Gain metric shows the shaped three that could be used if the region was optimized to have a function of the section of the secti								
Lad Induition example in runtice eartheat         OpenMP Regim       OpenMP Regim <th colspan<="" td=""><td></td><td></td><td>CONTRACTOR OF A</td><td></td><td></td><td></td><td></td></th>	<td></td> <td></td> <td>CONTRACTOR OF A</td> <td></td> <td></td> <td></td> <td></td>			CONTRACTOR OF A				
transvésorational Abende 1. Nachda dynamic man (More FLorids FLO), general, eff, origine 2. Western hug 2. 2017 7. 27, 26, 15, 15, 15, 16, 66, 66, 76, 76, 76, 76, 76, 76, 76, 7			at potential for performa	nce improvement, the Potential Gain metric shows the elapsed	time that could be saved if t	he nygon w	vex optimized to have i	
rg_envictantiale       Production       Product					OpenMP Potential Gain	181	OpenMP Region The	
<pre>texts Text Rest Rest Rest Rest Rest Rest Rest Res</pre>								
rg_setWormplaneWell Bispreder (Nachov depender of well Primer (E. 20. spenning, well and (Machov depender of Well Primer (E. 20. spenning, well and (Machov depender of Well Primer (E. 20. spenning, well and (Machov depender of Well Primer (E. 20. spenning, well and (Machov depender of Well Primer (E. 20. spenning, well and (Machov depender of Well Primer (E. 20. spenning, well and (Machov depender of Well Primer (E. 20. spenning, well and (Machov depender of Well Primer (E. 20. spenning, well and (Machov depender of Well Primer (E. 20. spenning, well and (Machov depender of Well Primer (E. 20. spenning, well and (Machov depender of Well Primer (E. 20. spenning), well and (Machov depende								
the obs/lamp/lamping lamping la								
Outcome       Note in particular ner-second direction of the logical second direction								
	"NUM is possible for non-summary	the entries.						
Bendertift Utilization Hintogram      Vectorization - 44.9%      of F         butture law	Cache Bound C: 1 Cache Bound C: 1 Cache Bound C: DRAM Bound C: DRAM Bound C:	14.1% N of Pipel 33.4% N of Close 4.0% of Close 4.0% of Close 29.7% N of Elaco	mika ticka					
O Vectorization → 44.9% P of F       Image: provide the provi								
○ Vectorization :: 44.9% M of F       Image: 1 minute in the first of F       Image: 1 minute in the first of F         ○ DP FLOP: 1 DDB       DDB       Image: 1 minute in the first of F       Image: 1 minute in the first of F         Image: 1 DDB + DDB       DDB       Image: 1 minute in the first of F       Image: 1 minute in the first of F         Image: 1 DDB + DDB       Image: 1 minute in the first of F       Image: 1 minute in the first of F       Image: 1 minute in the first of F         Image: 1 DDB + DDB       Image: 1 minute in the first of F       Image: 1 minute in the first of F       Image: 1 minute in the first of F         Image: 1 minute in the first of F       Image: 1 minute in the first of F       Image: 1 minute in the first of F       Image: 1 minute in the first of F         Image: 1 minute in the first of F       Image: 1 minute in the first of F       Image: 1 minute in the first of F       Image: 1 minute in the first of F         Image: 1 minute in the first of F       Image: 1 minute in the first of F       Image: 1 minute in the first of F       Image: 1 minute in the first of F         Image: 1 minute in the first of F       Image: 1 minute in the first of F       Image: 1 minute in the first of F       Image: 1 minute in the first of F         Image: 1 minute in the first of F       Image: 1 minute in the first of F       Image: 1 minute in the first of F       Image: 1 minute in the first of F         Image: 1 minute in the first of F <td>() Sandwinth Ofatteoin Hit</td> <td>inge anv</td> <td>D • = = = = 1</td> <td></td> <td></td> <td>. 27</td> <td>Auto Marban</td>	() Sandwinth Ofatteoin Hit	inge anv	D • = = = = 1			. 27	Auto Marban	
0         50         50         100	The second s	9% 🎙 of F					Construction of the second sec	
□ DP FLOP:         2 0%         □ Profession         ■ A symp         ■ Profession         ■ Professintet the Profession         ■ Professintet		6.0%		Contraction of the second s			er Mitternen Sera	
Packed 7     44.9%     124-bit 7     40.9%     40.9		2.0%			and the state of t			
216.52 <sup>1</sup> 4.1% Sodar <sup>2</sup> 05.1% <sup>2</sup> <sup>2</sup> come 1 <sup>2</sup> <sup>1</sup>		44.9%			Contraction of the state		W . Truttee	
214-54 * 4.1% 2 constant to the second secon		40.811 * 1414		Reconstruction of the intervention of the second states of the second second second second second second second	and header that below while a	( bellet)		
Scolar") (5.1% )	10 M M M M M M M M M M M M M M M M M M M		and a		within the diff. of a line of the	and the local data		
		100 T 100 T	1	b. Second a solution in the second states in the second s second second se second second s Second second se Second second sec	standard by had by a been shad and shad a con-	C MARKED IN		
(a) (Chickey) (Alson)		100 T 100 T	lalm.	s, Anna an Sanna Anna Anna Anna Anna Anna	en(I)(MI)(I) bel ANAIGRANS	DARAD	W maked	
		65.1%.9	lalm.		State Street Street		a bad tables	
		11.1%.9 1 + series			State Street Street		a - faul titles	

### Intra- Node Parallelism



### Intra- Node Parallelism



- Dual-Socket system most common datacenter node
- Cache Coherent Non-Uniform Memory Access (ccNUMA) System
- Shared Memory based Parallel Programming – addressed by multiple Processes (e.g. MPI) and / or multiple Threads per Process (e.g. OpenMP)
- Possibility of accelerators connected via PCI or similar (e.g. CXL)

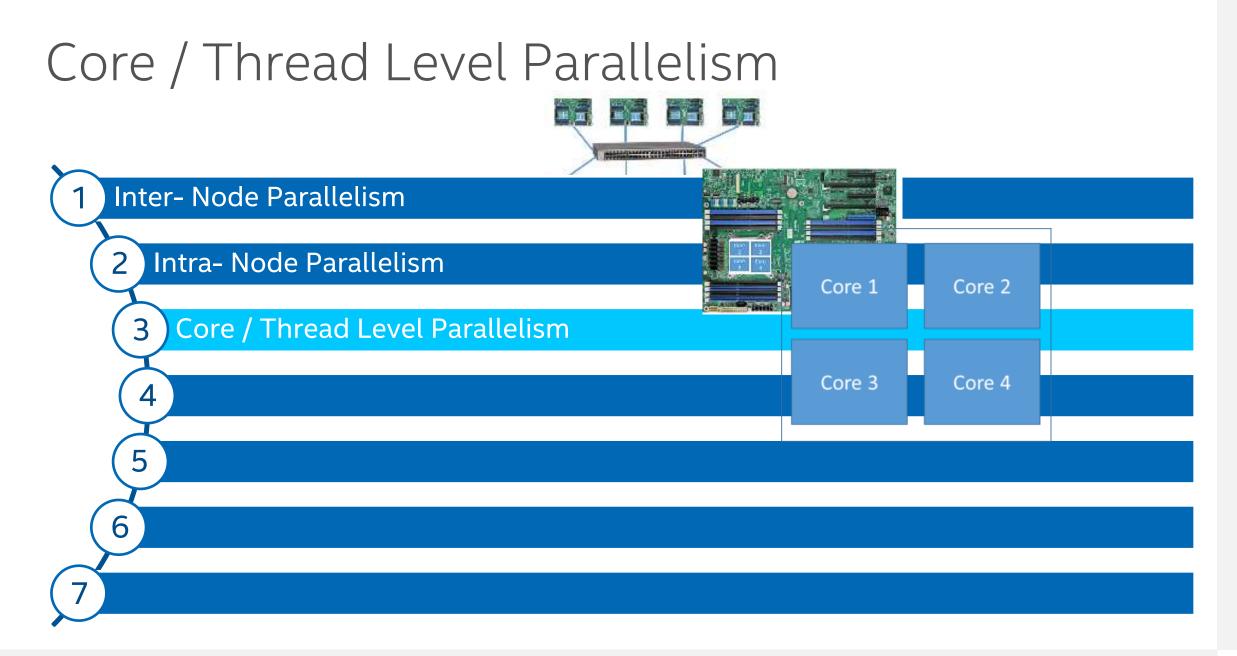
## Intra- Node Parallelism

What can I do?

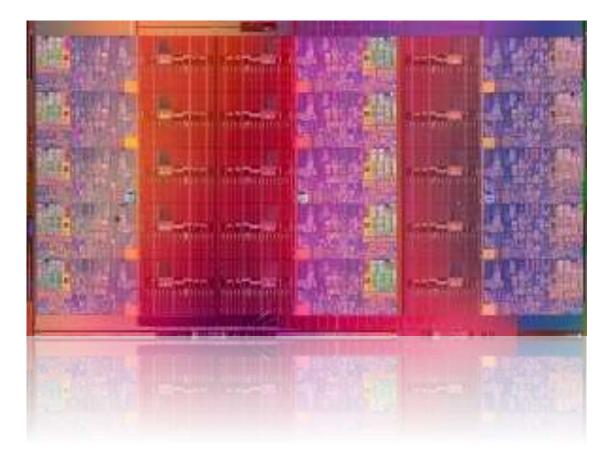
- Tune for best Memory Locality (NUMA optimization)
- Employ proper process pinning
- Avoid frequent involvement of Operating System (OS) Kernel e.g. by using scalable memory allocators

#### Which Tools?

- Identify issues with the VTune<sup>™</sup> Profiler
  - Hotspots
  - Memory Access
  - HPC Performance Characterization
- Identify issues with the VTune<sup>™</sup> Profiler Application Performance Snapshot



# Core / Thread Level Parallelism



- Multi-Core CPU with 10s of cores interconnected by a ring or mesh
- Some levels of private cache (L1/L2) and shared cache (L3)
- Dynamically scaling core frequencies
- Shared Memory based Parallel Programming

# Core / Thread Level Parallelism

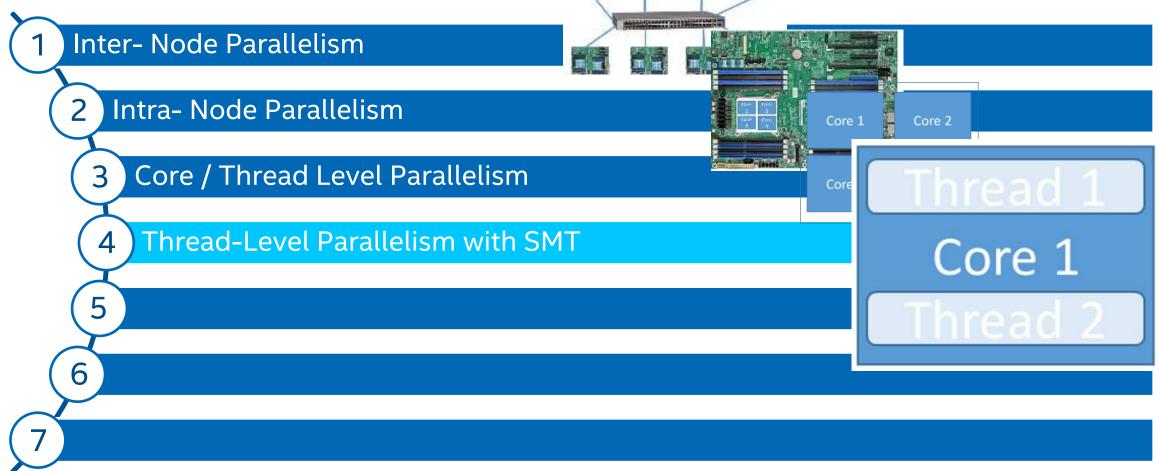
What can I do?

- Tune for best Memory Access (Cache-Blocking, non-temporal stores, ...)
- Load-Balance Threading
- Utilize performance optimized libraries wherever possible

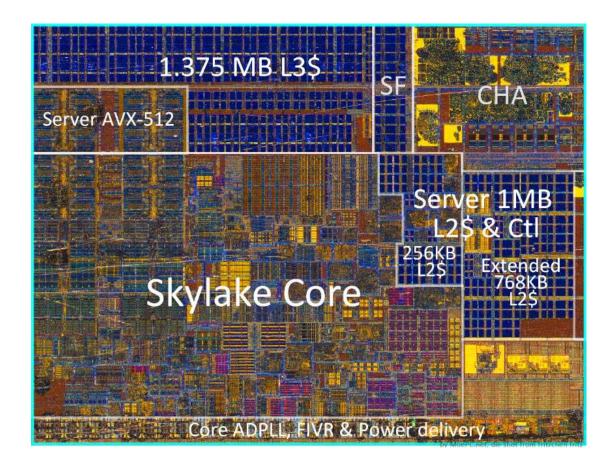
Which Tools?

- Identify issues with the VTune<sup>™</sup>
   Profiler
  - Hotspots
- Math Kernel Library
- Threading Building Blocks
- Intel C++ / Fortran Compiler OpenMP Runtime

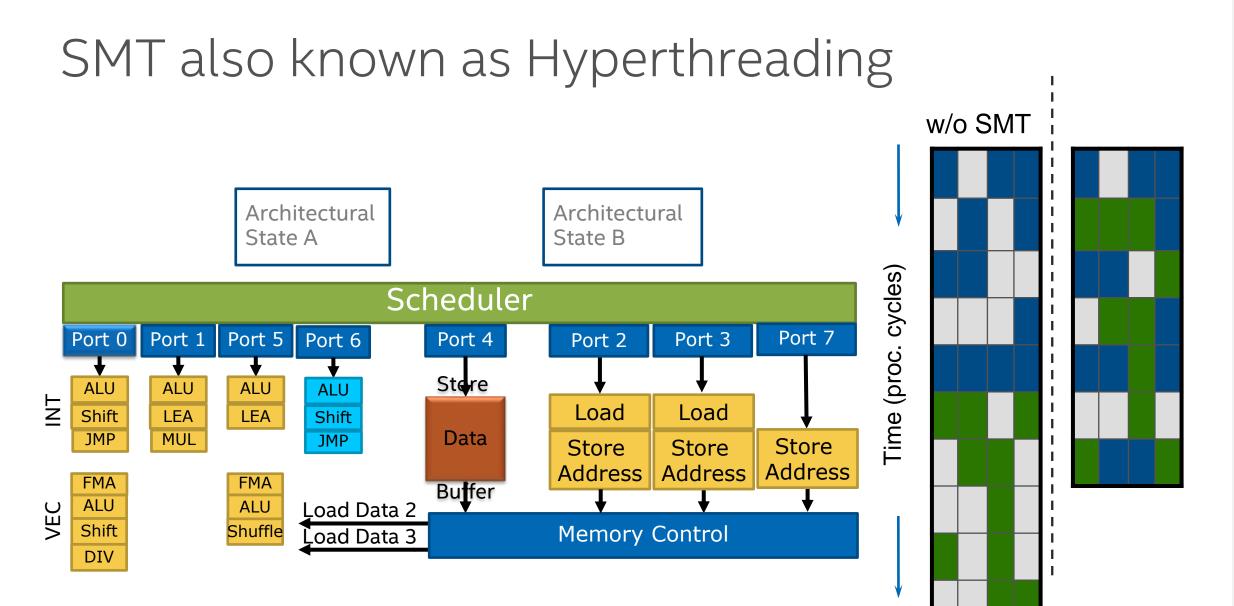
# Thread-Level Parallelism with Simultaneous MultiThreading (SMT)



## Thread-Level Parallelism with Simultaneous MultiThreading (SMT)



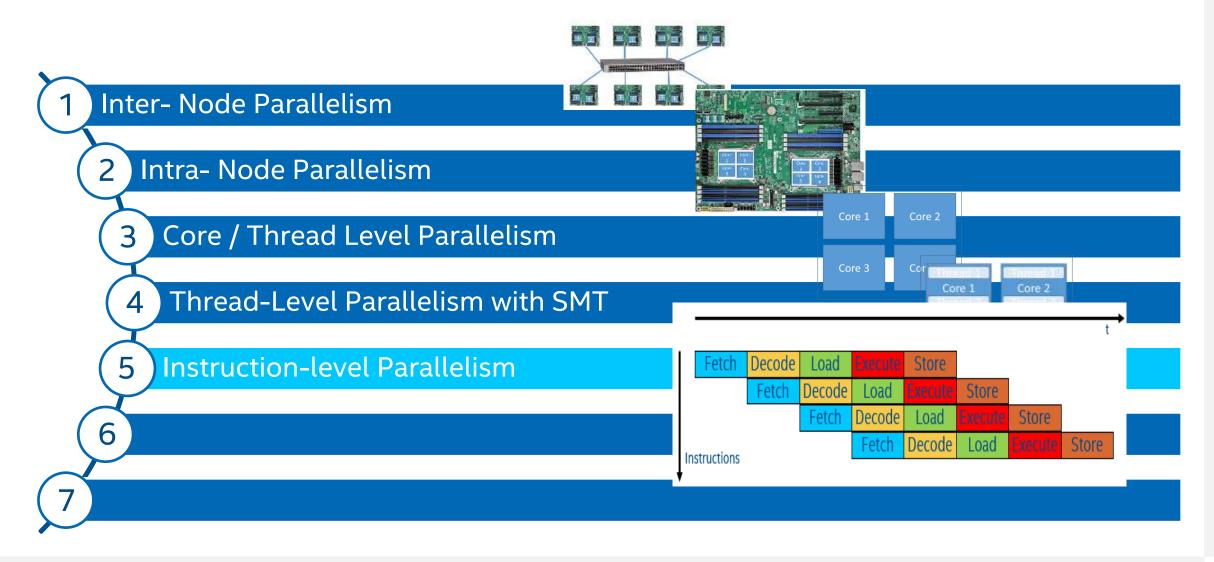
- X86 ISA Microprocessor SMT Core
- Partitioned Front End with multiple buffers keep arch. state
- Shared Backend with execution blocks
- SMT driving Instruction Level Parallelism (ILP) increase, while complementing Out of Order Execution
- Implicit Programming Model -Threading

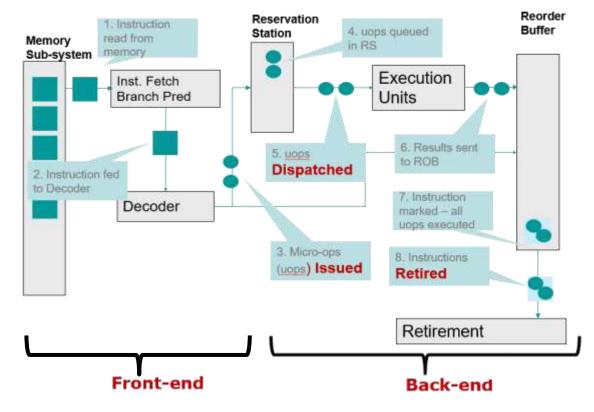


Thread-Level Parallelism with Hyperthreading (SMT) What can I do? Which Tools?

- Enable SMT in the BIOS
- Use pinning strategies to leverage / not leverage SMT
  - OMP\_PLACES / KMP\_AFFINITY
  - I\_MPI\_PIN\_CELL etc.
- Effective SMT usage requires instruction diversity between threads

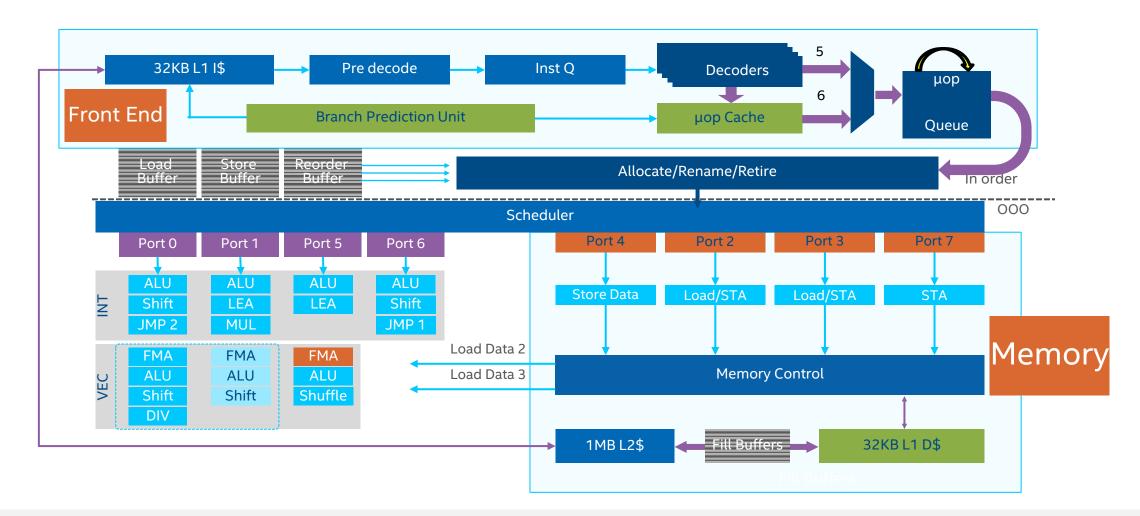
- Identify issues with the VTune<sup>™</sup> Profiler
  - Microarchitecture Analysis
- MKL BLAS3 routines leverage SMT
- Use the Pinning Simulator for Intel<sup>®</sup> MPI Library
- Use the Intel Compiler OpenMP Runtime environment variables / APIs for pinning





#### The life of a program instruction

- X86 ISA Microprocessor SMT Core
- Complex Instruction Set Computer (CISC) Frontend with Reduced Instruction Set Computer (RISC) Backend
- SuperScalar Out-of-Order Backend increasing ILP (Speculative Execution)
- Von Neumann Architecture (L1+) and Harvard Architecture (L1) Core
- Pipelining (parallelism over time)
  - Through Front-end & Back-end stages
  - Through Execution Units (ALUs)

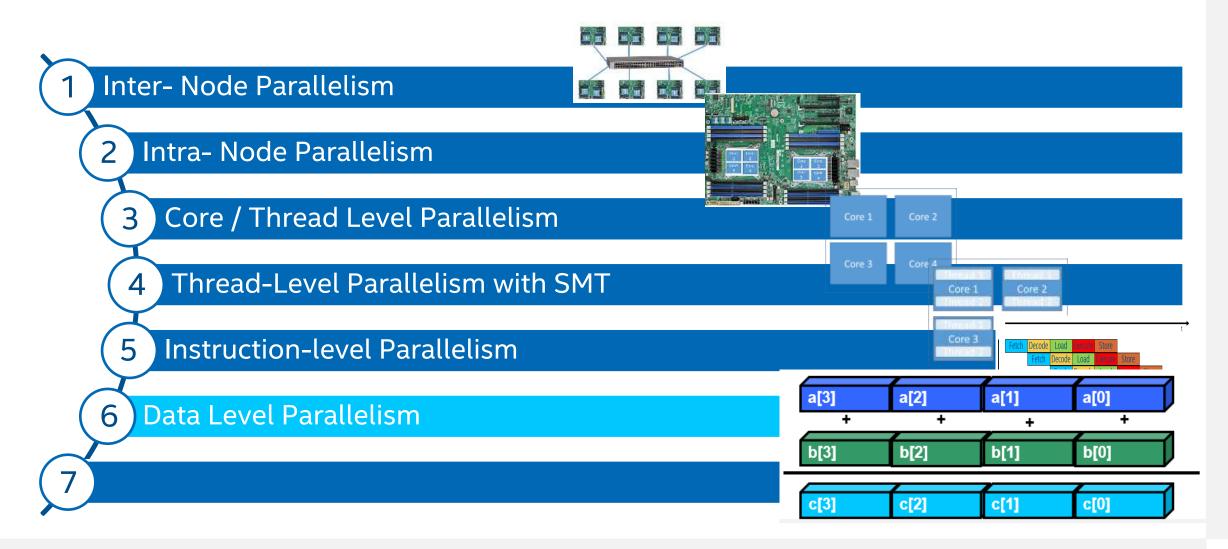


What can I do?

- ILP mostly done by the Compiler use the right compiler
- Effective ILP usage requires instruction diversity wherever possible
- Some instructions may stall faster than others – e.g. try to replace multiple divisions by reciprocal

Which Tools?

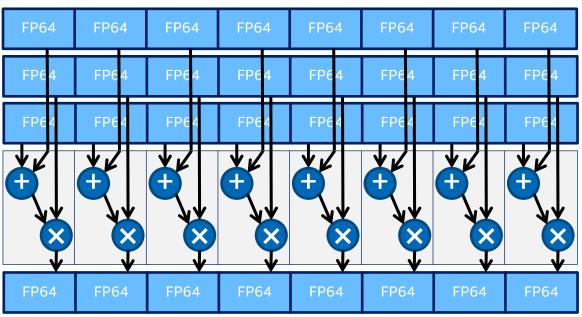
- Identify issues with the VTune<sup>™</sup>
   Profiler
  - Microarchitecture Analysis
- Intel<sup>®</sup> DPC++/C++ Compiler, Intel<sup>®</sup> C++ Compiler Classic, Intel<sup>®</sup> Fortran Compiler Classic, & Intel<sup>®</sup> Fortran Compiler (Beta)



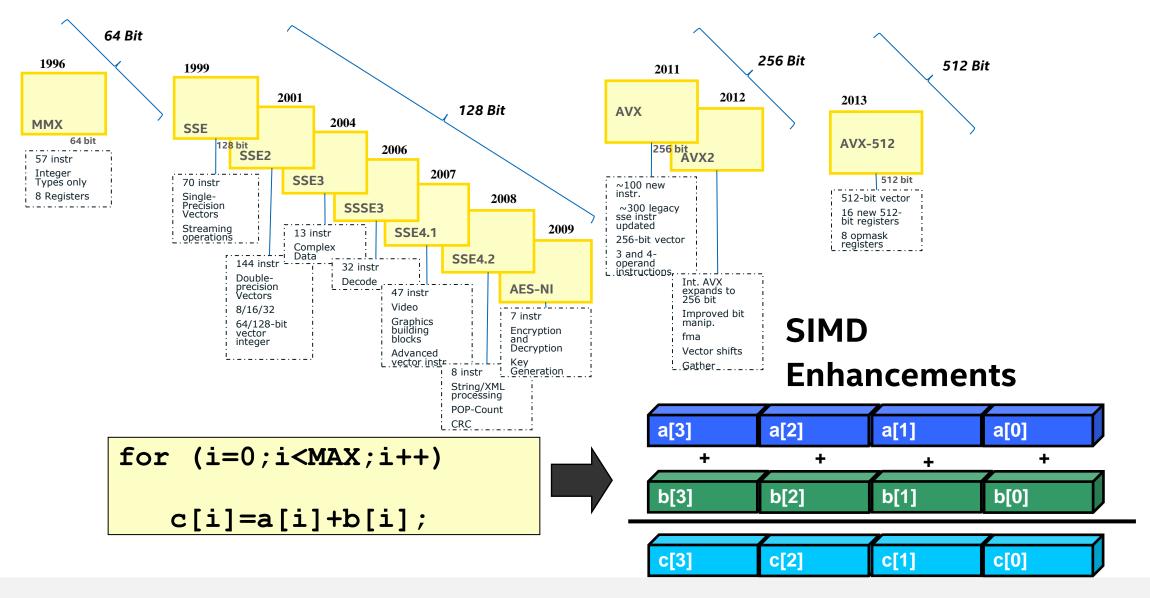
Fused Multiply Add (FMA)

AVX-512 Vector Instruction

 $\mathsf{D} = (\mathsf{A} + \mathsf{C}) \times \mathsf{B}$ 



- X86 ISA Extensions
- Flynn's taxonomy: Single Instruction Multiple Data (SIMD)
- Code candidates: Loops to split iterations in chunks of SIMD width – reducing the trip count
- Vectorization can happen implicitly with Compiler generating instructions



**Performance Libraries (e.g. IPP and MKL)** 

**Compiler: Fully automatic vectorization** 

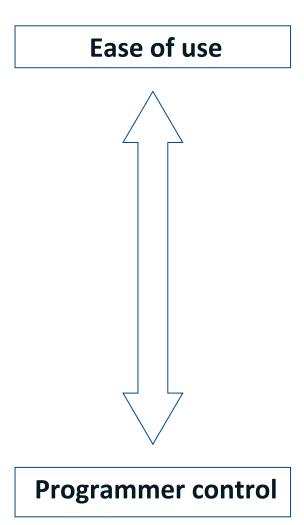
Compiler: Auto vectorization hints (#pragma ivdep, ...)

Explicit vectorization with OpenMP\* 4.0 and later (SIMD Directive)

SIMD intrinsic class (F32vec4 add)

Vector intrinsic (mm\_add\_ps())

Assembler code (addps)



What can I do?

- Use the most advanced Vector ISA available on your machine – the compiler default is SSE2
- Add code pragmas to the nonvectorized loops in order to feed compiler with domain knowledge
- Re-arrange code to achieve
  - loops with known trip count at runtime
  - branch-free loops to increase efficiency
- Tell the compiler to use full 512bit vector width in case of compute heavy loops

#### Which Tools?

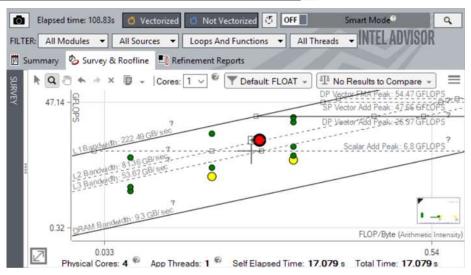
- Intel<sup>®</sup> DPC++/C++ Compiler, Intel<sup>®</sup> C++ Compiler Classic, Intel<sup>®</sup> Fortran Compiler Classic, & Intel<sup>®</sup> Fortran Compiler (Beta)
- Use the Intel<sup>®</sup> Advisor to
  - Identify vector code candidates
  - Determine vector efficiencies with
  - Determine code efficiencies with the Roofline Analysis
  - Step-by-Step guide towards efficient vector code

#### **Programming Guidelines for Vectorization**

Elapsed time: 125.72s OVectorized	Loops	ot Vectorized And Functio		Threads 🔻				OFF	Sm	art M	ode <sup>®</sup> NTEL A	DVISOR	م 201
🛛 Summary 🤣 Survey & Roofline 📲 Re	finemen						Vectori	zed Loops	11	-	<b></b>	Instruction	Set
+       -       Function Call Sites and Loops         Image: Comparison of the state o		Perfor Issues	Self Time 🕶	Total Time	Туре	Why No Vectorization?		Efficiency	Gain	VL			Da
[loop in main at roofline.cpp:295]			18.538s	18.538s 0	Vectorized (B		AVX	~100%	5.34x	4	5.34x		Flo
[loop in main at roofline.cpp:310]			18.394s	18.394s 0	Vectorized (Bo		AVX	~100%	5.34x	4	5.34x		Flo
☑ <sup>™</sup> [loop in main at roofline.cpp:221]			14.741s 🔲	14.741s 0	Scalar	novector dire							Flo
☑ੴ [loop in main at roofline.cpp:234]			11.117s 🗖	11.117s	Scalar	inner loop w							Flo
[loop in main at roofline.cpp:247]			6.967s	6.967s1	Vectorized (Bo		AVX	~31%	1.22x	4	1.22x	Inserts; U	Flo
I →	4							all control					Þ

#### Advisor Vectorized Loop Summary (top-left)

Advisor Roofline (bottom right)



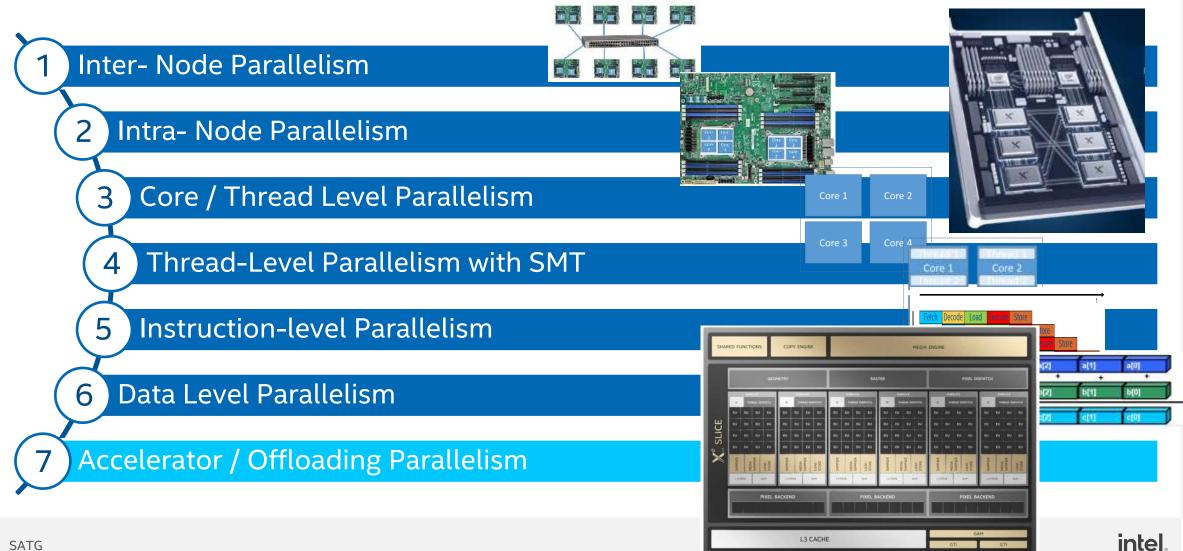
## Data Level Parallelism – legacy instructions



#### What is / was that?

#### The X87 Coprocessor

- FP extension of the 8086 ISA later (80486) integrated
- Today, x87 instructions rarely used, If however, numerical results might differ (80Bit intermediate)!!!
- Avoid these instructions in favor of reproducible result for both, scalar & vector code



58

SHAR	SHARED FUNCTIONS COPY ENGINE								MEDIA ENGINE															
	GEOMETRY								RASTER								FIREL DISPATCH							
		'n				'n.				100			-	100		inatter .		1033		MATE-		1000		10000
	ti.	ŵ	=	D/	-	10	tu.	81	10	ш	67	m	11/	ŧù.		-	-	ŵ	Ξŵ	÷.	ŵ	w		ŵ.
X° slice	89	eni	ev.	89	80	w	EU	84	m	#¥	80	m	~	EU.	#0	e.	en.	60	-ev	EN.	tv.	- 89		ŧv
SL	ŧV.			- EV	84	10	10	84	w	.10	••	w	.64	614	40	- 84	ευ.			. 600	w			117
2	w.	eù:	-	w	•	ю.	80	eu	111	80	ev.	-					80	-	w	eu		ιų.	ęs.	1Ú
×	zitioners.	-	amont	NTCHE	Simon	NETTA	smeet	Inches In	shares	- Hereit	Sumula	1040	NUMPON		Diants	UD4D	Windows	and the	thett	0101	Soouth	- North	Survey.	Cross.
	10/045 (See 10/046 0)#							11/104 Sam (11/104 Sam							Antena har Antena har									
	PIXEL BACKEND							PIXEL BACKEND								PIXEL BACKEND								
L3 CACHE GTI								АМ	G	ħ														

- GPGPU denser compute
- Slice -> SubSlice -> Execution Units (EUs)
- Each EU has
  - 8 Threads (similar to SMT)
  - 28KB Register File
  - 2x 4-wide SIMD units
- Offload based Programming enabled by oneAPI

```
Data Parallel C++ (DPC++) -
```

```
Hello World
#include <CL/sycl.hpp>
using namespace sycl;
```

```
int main() {
    std::vector<float> A(1024), B(1024), C(1024);
    // some data initialization
    {
        buffer bufA {A}, bufB {B}, bufC {C};
        queue q;
        q.submit([&](handler &h) {
            auto A = bufA.get_access(h, read_only);
            auto B = bufB.get_access(h, read_only);
            auto C = bufC.get_access(h, write_only);
            h.parallel_for(1024, [=](auto i){
                  C[i] = A[i] + B[i];
            });
        });
    });
    for (int i = 0; i < 1024; i++)
        std::cout << "C[" << i << "] = " << C[i] << std::endl;
    }
}</pre>
```

- Intel<sup>®</sup> oneAPI Accelerator
   Programming
  - C/C++
    - Intel <sup>®</sup> DPC++ Compiler
    - Intel<sup>®</sup> C++ Compiler with OpenMP Offloading
  - Fortran
    - Intel <sup>®</sup> Fortran Compiler with OpenMP Offloading

}

What can I do?

- Enable your code today for Intel Integrated Graphics GPUs and upcoming datacenter GPGPUs
- Reduce data transfers between host and device
- Minimize communication / synchronization in between individual threads

Which Tools? – Intel® oneAPI

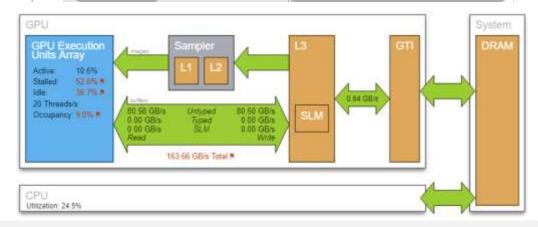
- Intel<sup>®</sup> DPC++/C++ Compiler & Intel<sup>®</sup> Fortran Compiler (Beta)
- Intel<sup>®</sup> oneMKL
- Use the Intel<sup>®</sup> Advisor to
  - Find offloading candidates with the Offload Advisor
  - Estimate code speedup of GPU with the Offload Advisor
- Use Intel<sup>®</sup> VTune<sup>™</sup> Profiler to
  - Analyze the offload performance

#### Offload Advisor – Efficiency

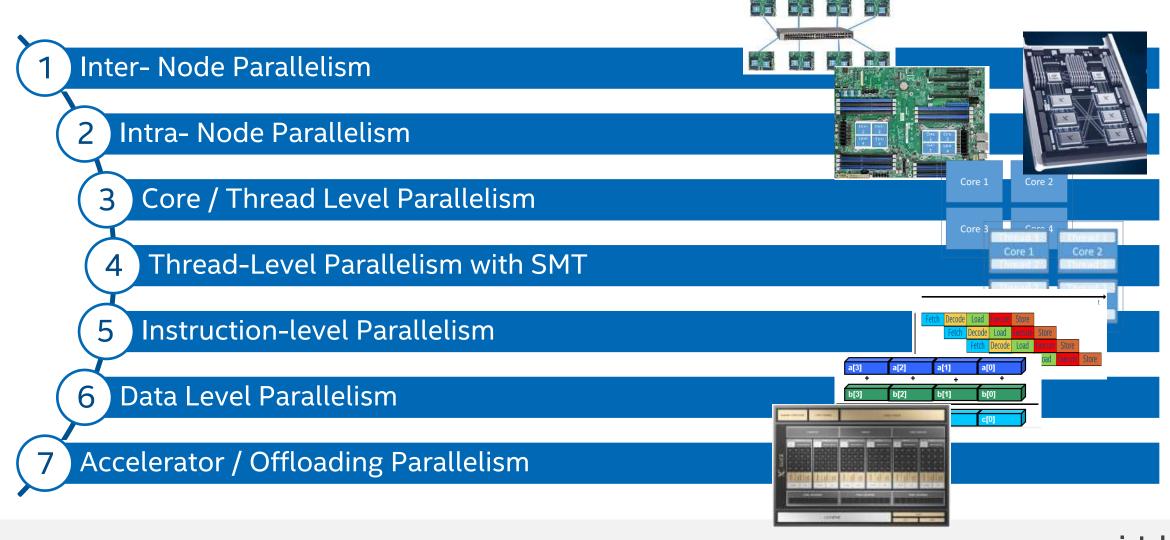
Intel® Advisor Beta OFFLOAD ADVISOR Summary   Offloaded Reg	ions   Non Offloaded Region	s   Call Tree   Confi	guration   Logs
ΔΔγ .	Number of 1 Offloads ⑦	Fraction of Accelerated Cod	e <sup>@</sup> 99%
Program metrics ⑦			
Original ⑦ 25.07s			
Accelerated ② 5.85s			
Target Platform Gen9 GT2	Time on Host ⑦	0.27s	5%
Number of Offloads ⑦ 1	Time on Accelerator ⑦	5.58s	
Speed Up for Accelerated Cod 4.4x	Data Transfer Tax 🕖	0s	
			0.50/
Amdahl's Law Speed Up ⑦ 4.3x	Invocation Tax ⑦	<0.01s	95%

#### VTune – GPU Offload Analysis

So	ource Assembly 💵 = 😽 🔲 🛻	۵ <u>**</u>
🛦	Source	<ul> <li></li></ul>
158	dx = ptr[j].pos[0] - ptr[i].pos[0];	75,002,500
159	dy = ptr[j].pos[1] - ptr[i].pos[1];	12,500,000
160	<pre>dz = ptr[j].pos[2] - ptr[i].pos[2];</pre>	12,500,000 📒
161		
162	distanceSqr = dx*dx + dy*dy + dz*dz	87,500,000
163	distanceInv = 1.0 / sqrt(distanceSo	12,500,000 📒
164		
165	ptr[i].acc[0] += dx * G * ptr[j].ma	162,503,750
166	ptr[i].acc[1] += dy * G * ptr[j].ma	150,000,000
167	ptr[i].acc[2] += dz * G * ptr[j].ma	150,000,000



## The Seven Levels of Parallelism



## The Seven Levels of Parallelism

#### Parallelism on the Intel Architecture

Inter- Node Parallelism

Intra- Node Parallelism

Core / Thread Level Parallelism

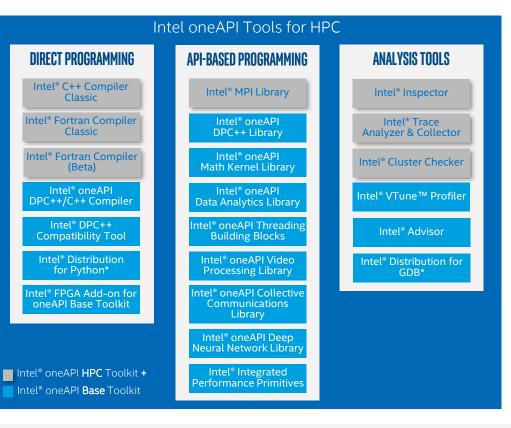
Thread-Level Parallelism with SMT

Instruction-level Parallelism

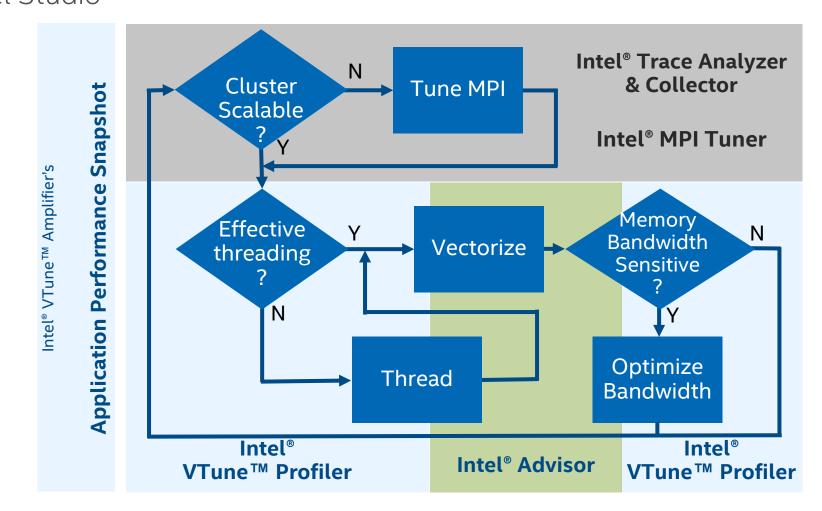
Data Level Parallelism

Accelerator / Offloading Parallelism

#### Supported by the Intel<sup>®</sup> oneAPI Base & HPC Toolkits



## Performance Analysis Tools for Diagnosis





Code Modernization is not always 'easy & quick' (a.k.a. "rewrite"). Common methodology is to (1) analyze, then (2) optimize.

Two themes consistently re-occur – Task and Data Parallelism.

To achieve best CPU performance, make sure your programs are efficiently vectorized and are well parallelized.

Especially the latter one also counts for GPUs.

## (Recorded) Tools Webinars

- Intel<sup>®</sup> oneAPI technical Webinar (2 days) co-hosted with Zuse Institute Berlin
- oneAPI Webinar on March 2<sup>nd</sup> and 3<sup>rd</sup>, 2021
  - https://www.zib.de/workshops/2021/oneapi
  - https://www.hlrn.de/doc/display/PUB/Joint+NHR@ZIB+-+INTEL+++oneAPI+Workshop
- Intel<sup>®</sup> oneAPI Rendering Tookit Webinar (1 day):
  - Rendering 08.06.2021: <u>https://www.ai-spektrum.de/veranstaltungen/intel-oneapi-rendering-toolkit-workshop.html</u>
  - Make use of simulated data and virtualize them in high fidelity photo realistic fashion

#### Intel<sup>®</sup> oneAPI AI Analytics Toolkit webinar – 1 day :

- AI webinar on 15.06.2021: <u>https://www.ai-spektrum.de/veranstaltungen/artificial-intelligence-on-intelr-platforms-using-intelr-oneapi-ai-analytics-toolkit-openvino-workshop.html</u>
- Intel performance optimized AI tools kits for classic and machine learning

## Selected Links : Intel Tools User's Manuals

#### • Intel<sup>®</sup> C++ Compiler Classic Developer Guide and Reference

https://www.intel.com/content/www/us/en/develop/documentation/cpp-compiler-developer-guide-and-reference/top.html

Intel Data Parallel C++

https://link.springer.com/content/pdf/10.1007%2F978-1-4842-5574-2.pdf

#### • Intel VTune Cookbook

This Cookbook introduces methodologies and use-case recipes to analyse the performance of your code with VTune Profiler, a tool that helps you identify ineffective algorithm and hardware usage and provides tuning advice.

https://www.intel.com/content/www/us/en/develop/documentation/vtune-cookbook/top.html

#### Intel<sup>®</sup> Advisor Cookbook

The Intel® Advisor Cookbook contains step-by-step instructions to help effectively use more cores, vectorization, or heterogeneous processing using Intel Advisor.

https://www.intel.com/content/www/us/en/develop/documentation/advisor-cookbook/top.html

#### • Intel Inspector User Guide(s) for Linux and for Windows

Get Started with Intel® Inspector -Windows\* OS . This document explains how to get started with the Intel® Inspector workflows. Linux : https://www.intel.com/content/www/us/en/develop/documentation/inspector-user-guide-linux/top.html

Windows :

https://www.intel.com/content/www/us/en/develop/documentation/get-started-with-inspector/top/windows.html

#### Microsoft Visual Studio\* Integration of Intel VTune

https://www.intel.com/content/www/us/en/develop/documentation/vtune-help/top/launch/microsoft-visual-studio-integration.html



Data Parallel C++

C++ and SYCL

Mastering DPC++ for Programming of Heterogeneous Systems using

## Other useful Links

• Intel Software Developer Manuals

These manuals describe the architecture and programming environment of the Intel® 64 and IA-32 architectures.

#### Intel® 64 and IA-32 Architectures Software Developer Manuals

• Intel Software Development Tools Magazine 'Intel Parallel Universe'

https://www.intel.com/content/www/us/en/developer/community/parallel-universe-magazine/overview.html?s=Newest

• Intel Developer Zone

https://www.intel.com/content/www/us/en/developer/overview.html



## QUESTIONS?



#