

# Performance Optimization for Software Developers

with

# oneAPI & Parallelization on Intel<sup>®</sup> Architecture

Edmund Preiss  
Business Development Manager

Software and Advanced Technology Group (SATG)



intel<sup>®</sup>

# 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.

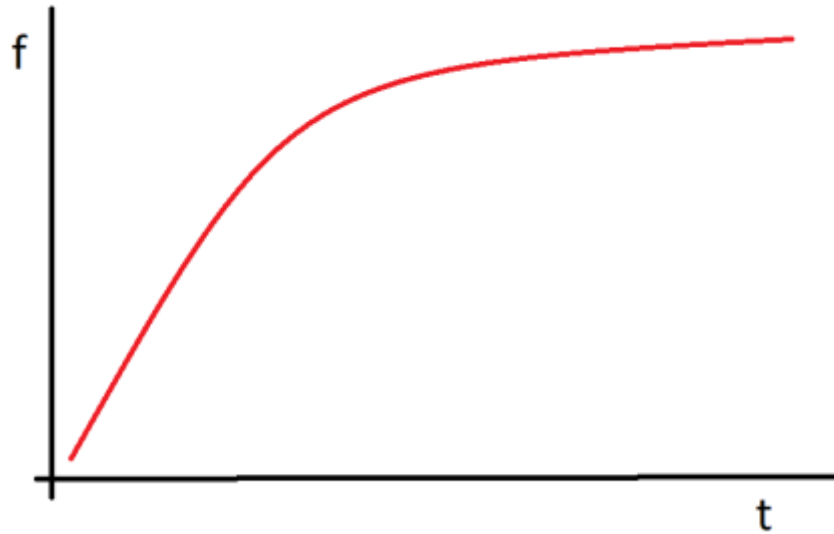
The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries.

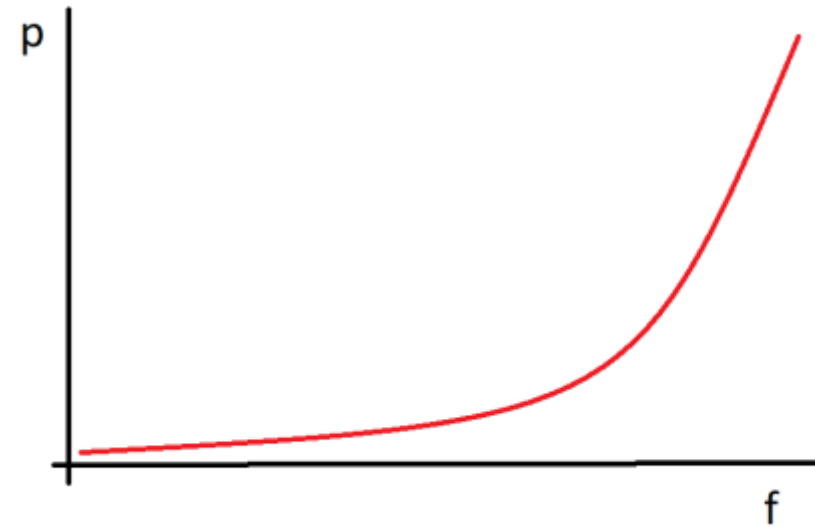
Other names and brands may be claimed as the property of others.

© Intel Corporation

# Welcome in the Parallelism Area !



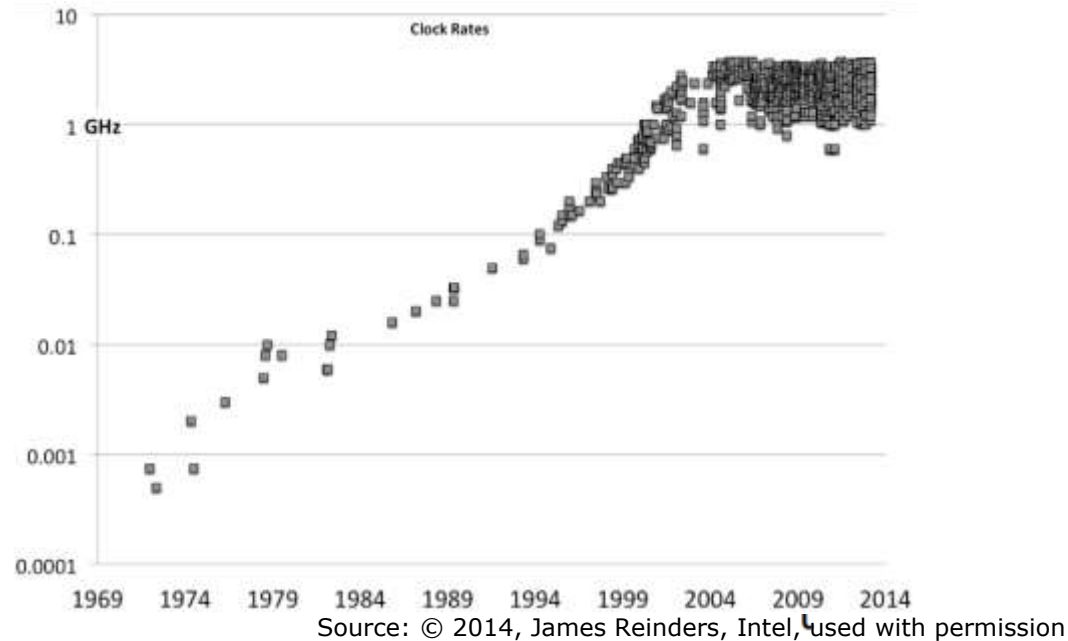
Microprocessor frequency over Time (history)



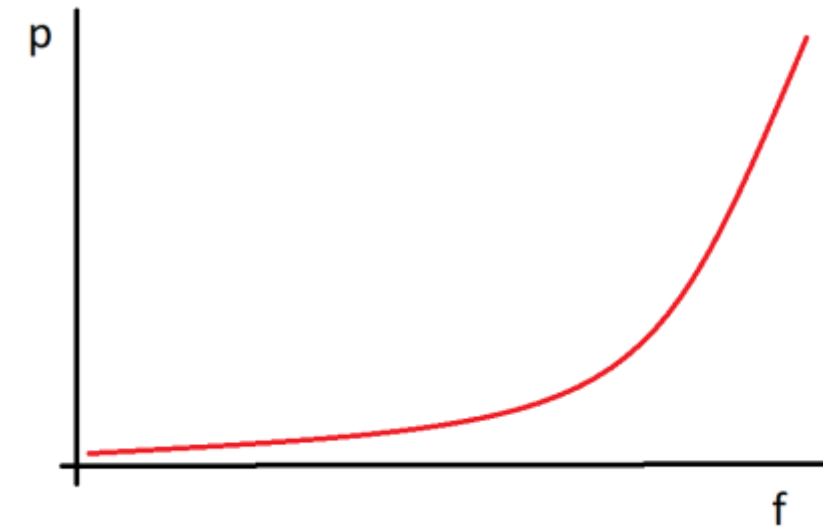
Microprocessor frequency versus Power consumption

- ... 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

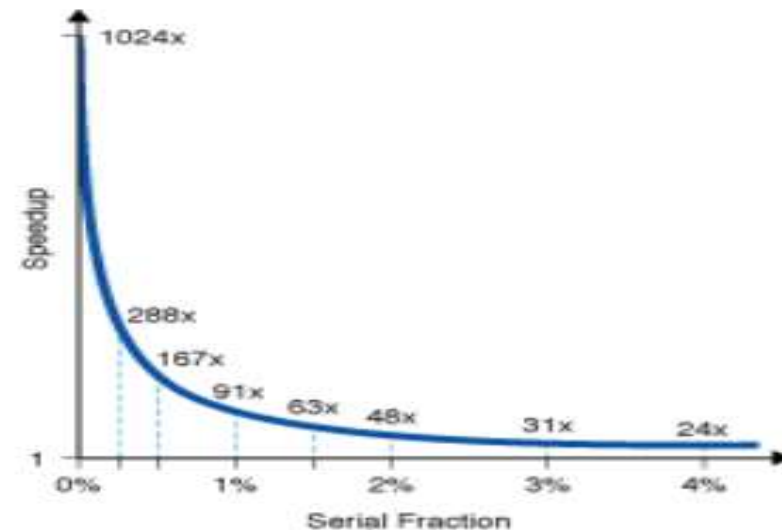
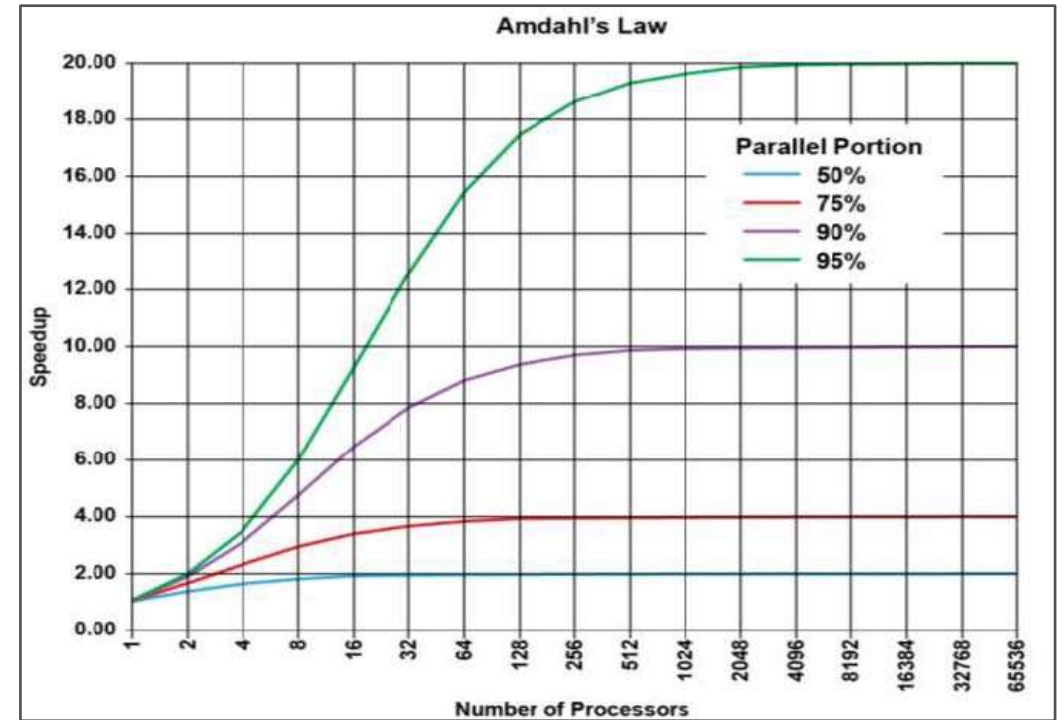
- ... performance is not only the computer architect's job anymore
- ... **performance increase is increasingly the job of the software developer**

# 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

$$\begin{aligned} \text{Speedup} &= (s + p) / (s + p / N) \\ &= 1 / (s + p / N) \end{aligned}$$

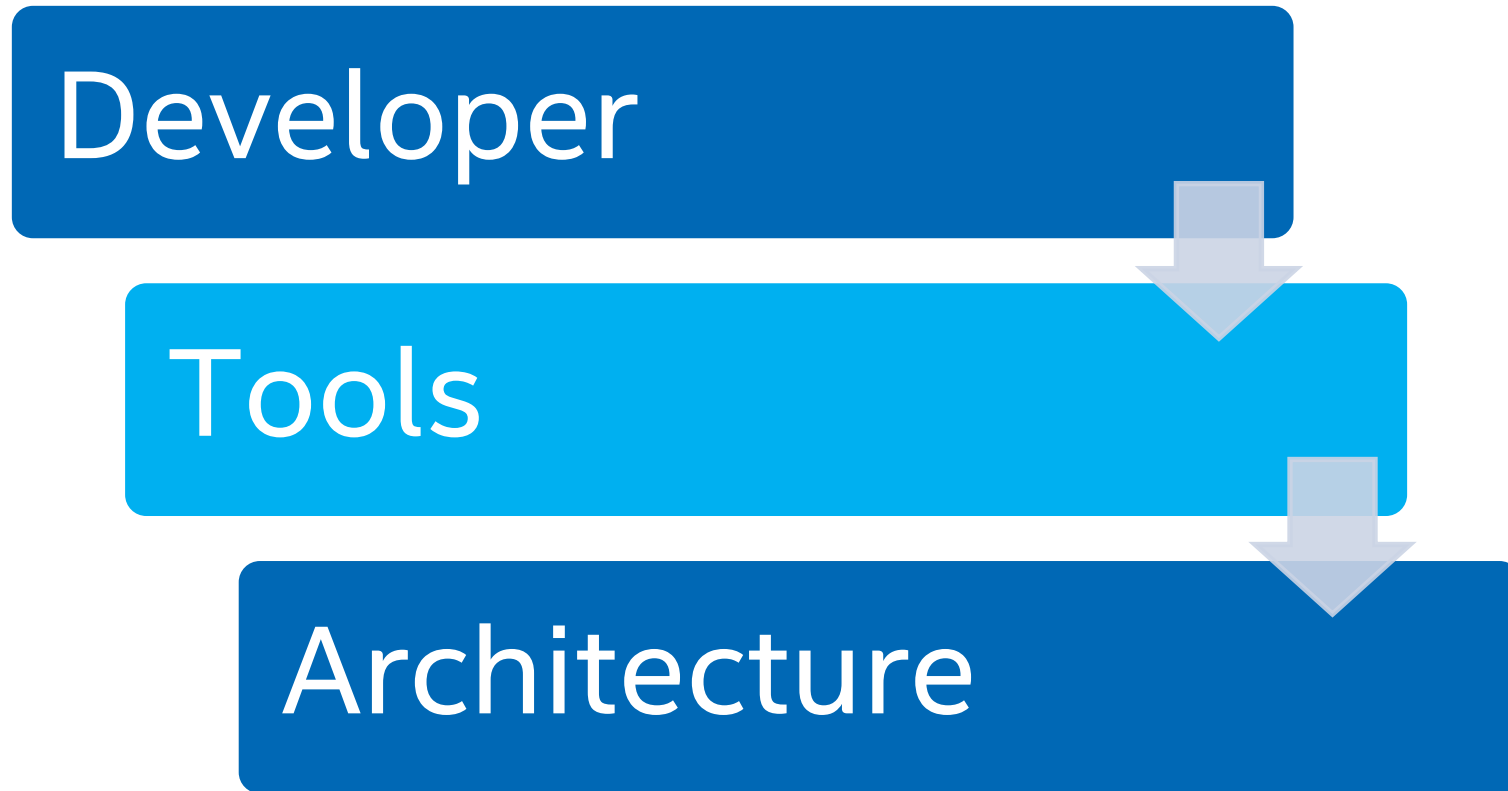


The 'Free Lunch' is over

“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



**oneAPI**

Open Industry  
Specification



**oneAPI**

Intel Product

# Programming Challenges

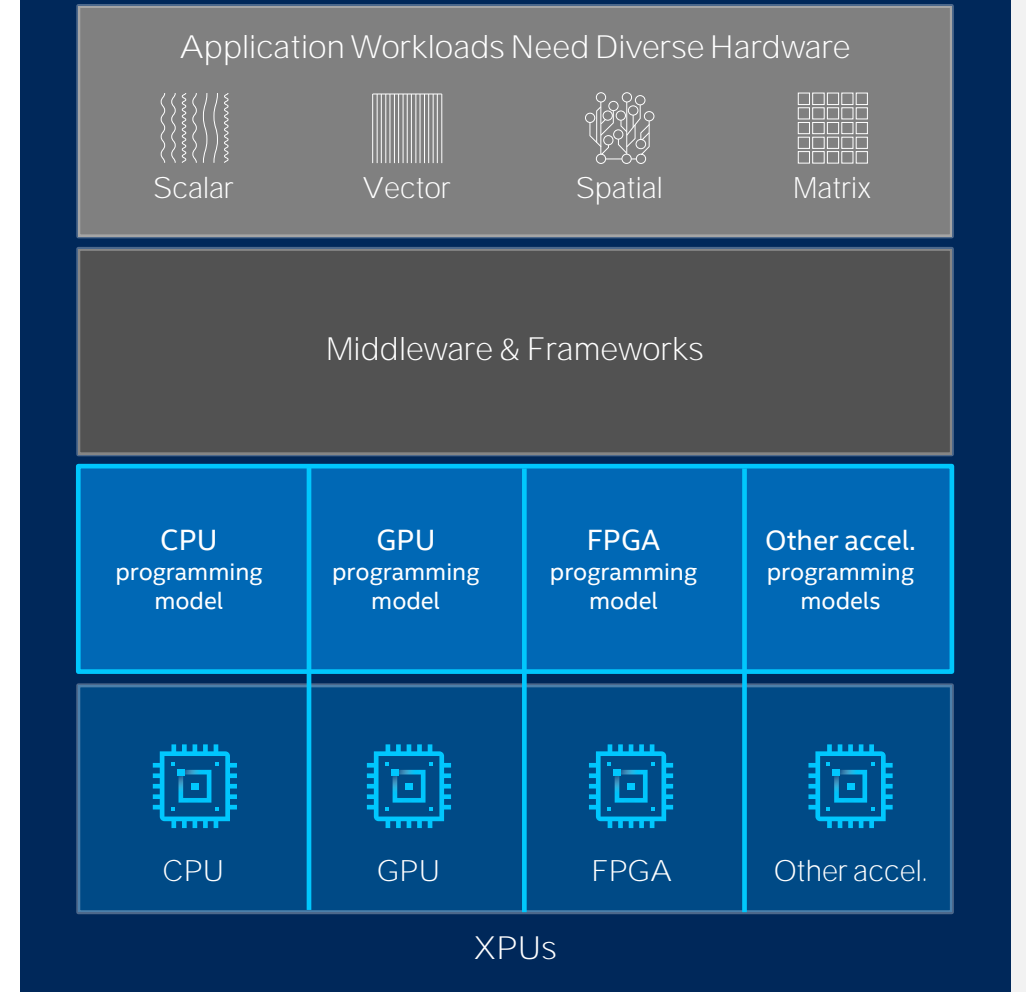
for Multiple Architectures

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



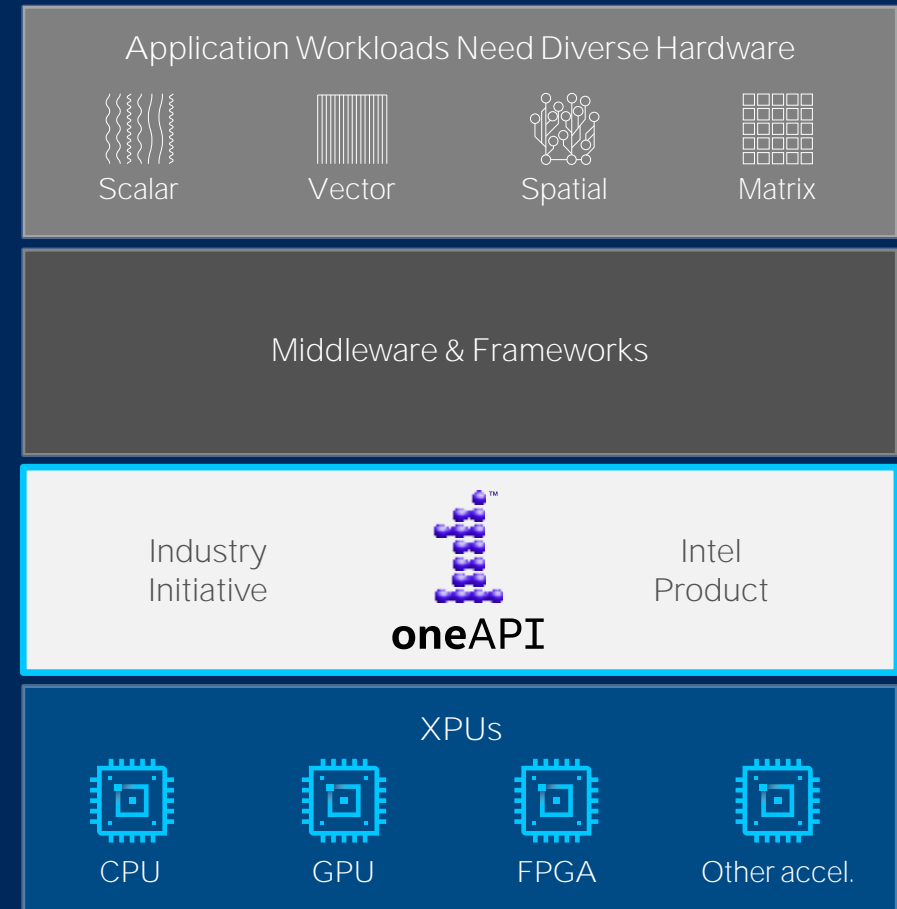
# Introducing oneAPI

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 domain-specific functions

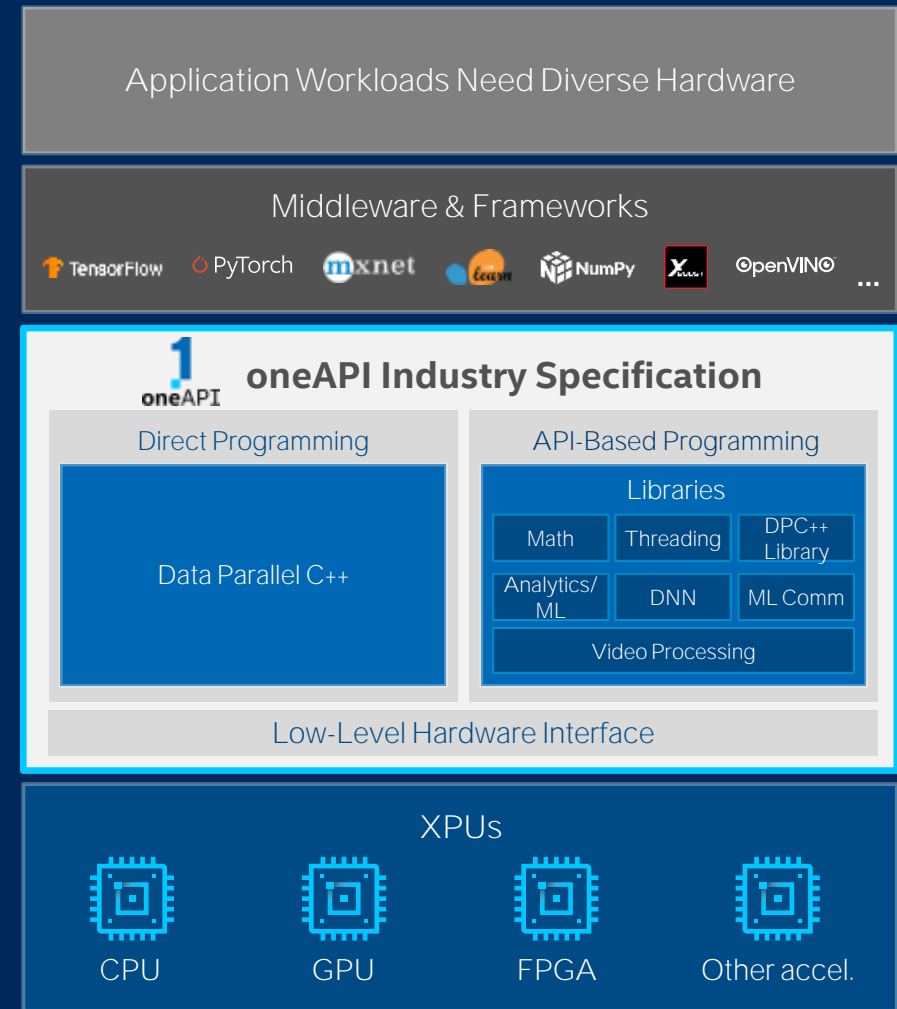
Low-level hardware abstraction layer

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++

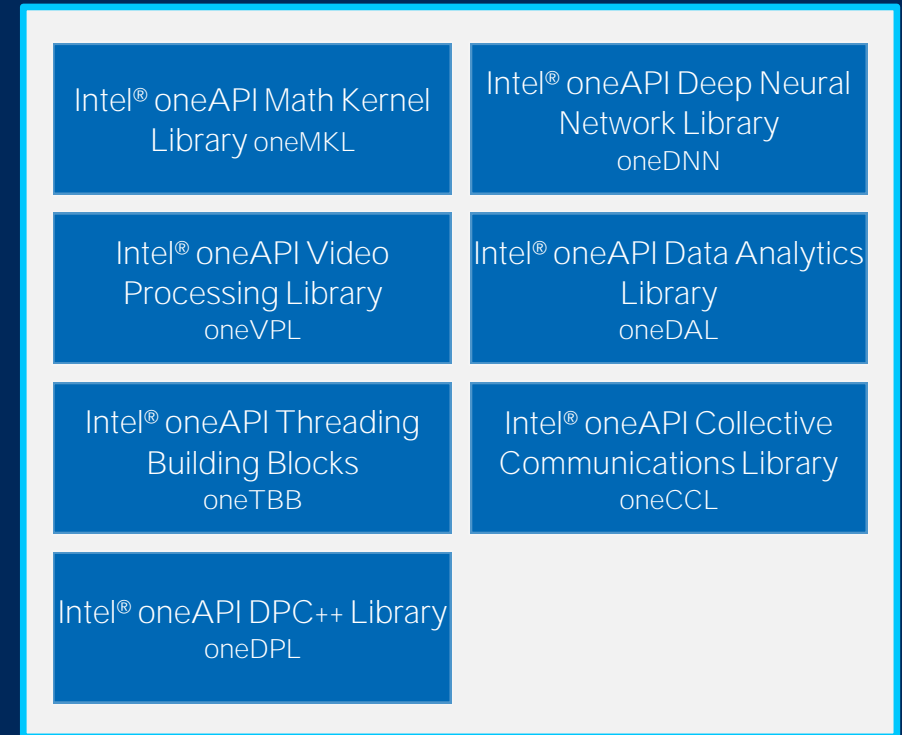
Community Extensions

Khronos SYCL

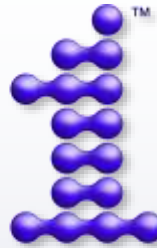
ISO C++

# Powerful oneAPI Libraries

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



# oneAPI – A Tools Development Framework



**oneAPI**

Open Industry  
Specification



**oneAPI**

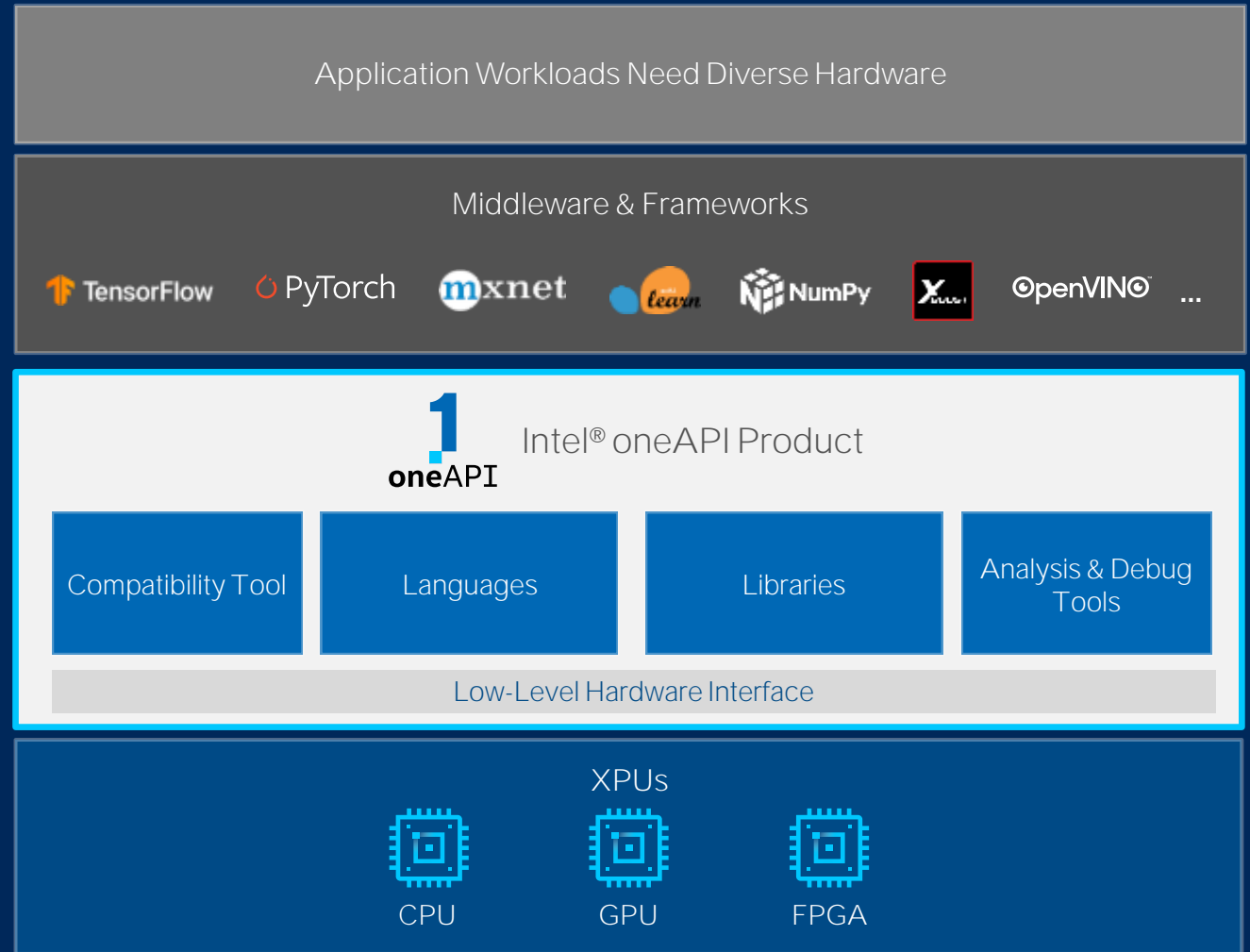
Intel Product

# Intel® oneAPI Product

Built on a Rich Heritage of Tools  
Based on Intel® Xeon® Processors,  
Now Expanded to XPU

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



[Available Now](#)



# Intel Compiler Transition: Classic to LLVM

## Start Your Migration Now

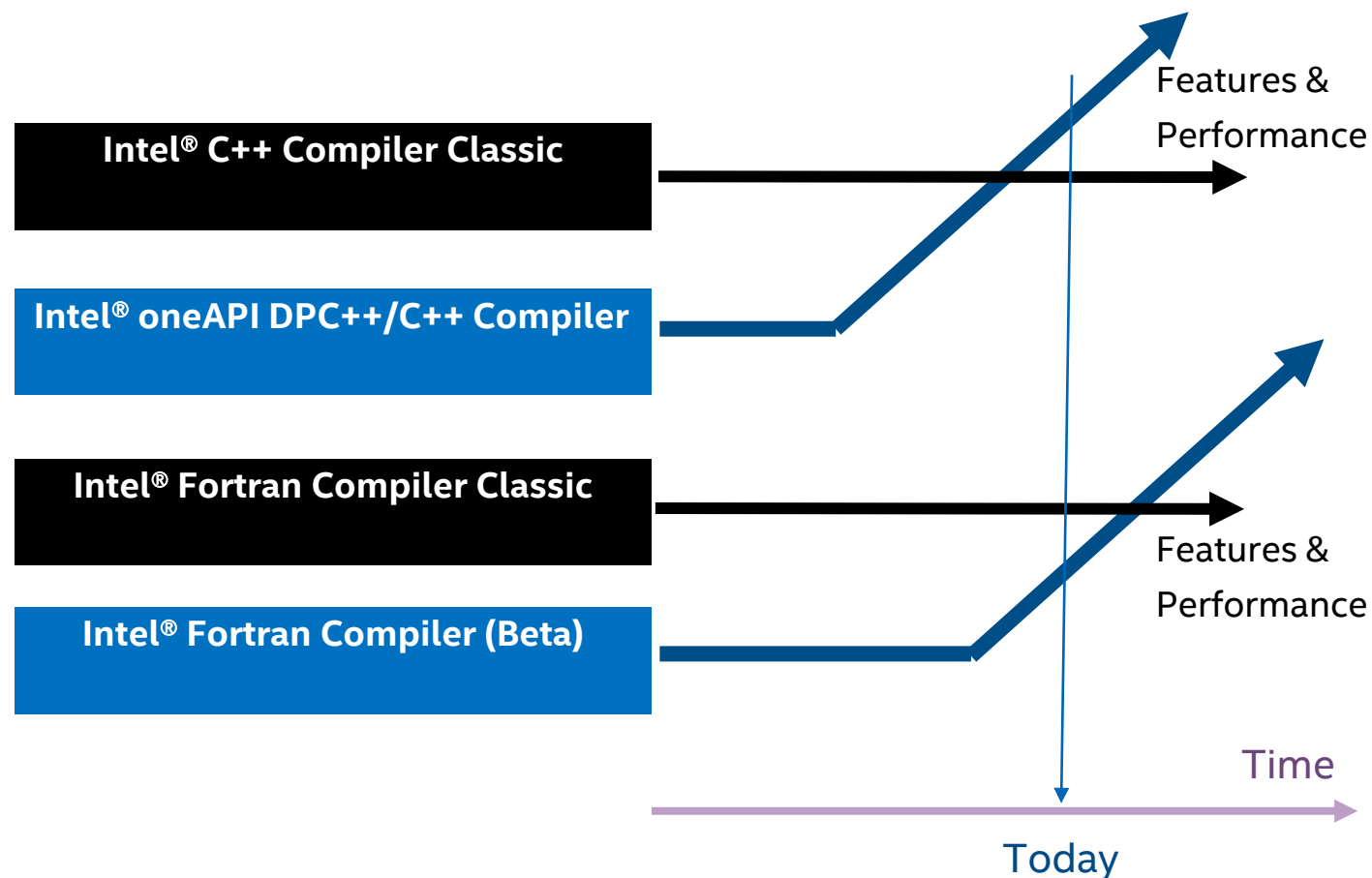
- Expand to XPU
- Modern LLVM Infrastructure

## Intel® oneAPI DPC++/C++ Compiler

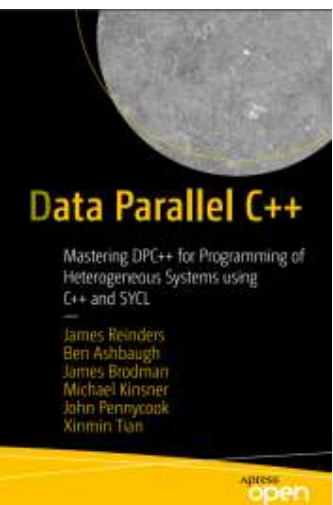
- Use for all new projects
- Migrate legacy projects

## Intel® Fortran Compiler (Beta)

- Test drive now & Provide feedback
- Prepare for migration



Performance, Quality, and Support Continues



# Intel® oneAPI Toolkits

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



## 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

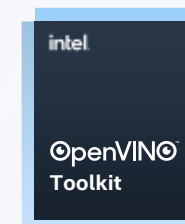
## Toolkits powered by oneAPI

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

# Streamlining Product Line



Composer Edition

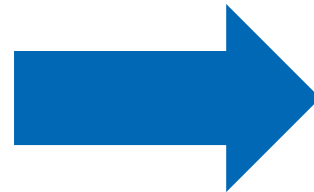
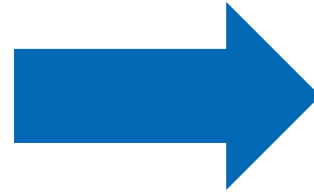


Professional Edition



Cluster Edition

CPU Focused (Core & Xeon)



Intel® oneAPI Base & HPC Toolkit Single-Node



Intel® oneAPI Base & HPC Toolkit Multi-Node

- CPU (Core & Xeon)
- GPU (Intel)
- FPGA (Intel)

# Intel® oneAPI Base & HPC Toolkit

## Intel® oneAPI Tools for HPC: Deliver Fast Applications that Scale

### What is it?

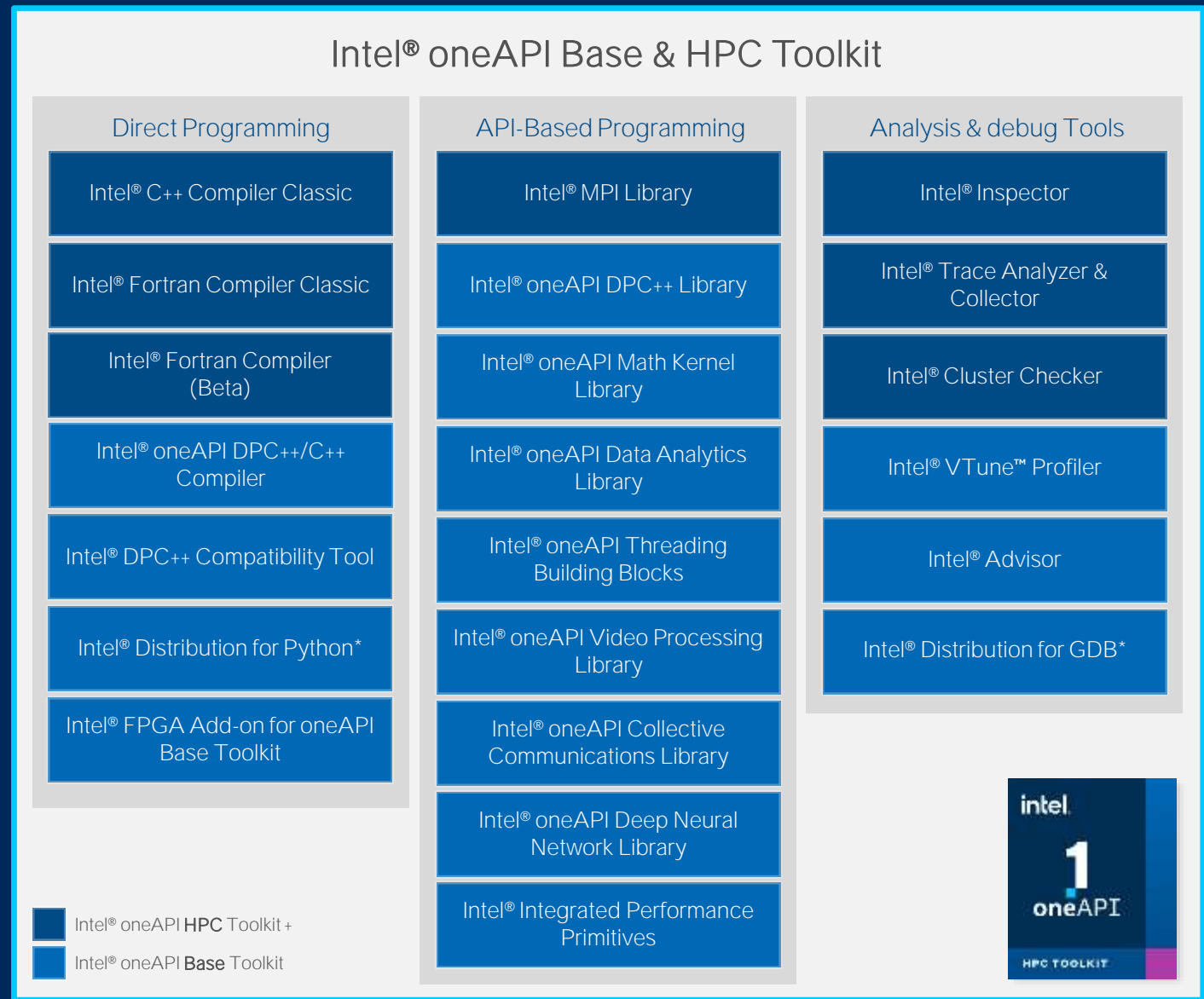
A toolkit that adds to the Intel® oneAPI Base Toolkit for building high-performance, scalable parallel code on C++, Fortran, OpenMP & MPI from enterprise to cloud, and HPC to AI applications.

### Who needs this product?

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

### Why is this important?

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



# Render Your Vision in Highest Fidelity Intel® oneAPI Rendering Toolkit

## Powerful Libraries for High-Fidelity Visualization Applications

- Deliver high-performance, high-fidelity visualization applications on Intel® 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 oneAPI Rendering & Ray Tracing Libraries

### Intel® Embree

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



### Intel® Open Image Denoise

AI-Accelerated Denoiser for Superior Visual Quality



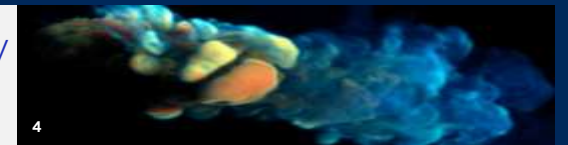
### Intel® OpenSWR

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



### Intel® Open Volume Kernel Library

Render & Simulate 3D Spatial Data Processing



### Intel® OSPRay

Scalable, Portable, Distributed Rendering API

### Intel® OSPRay Studio

Real-time rendering through a graphical user interface with this new scene graph application

### Intel® OSPRay for Hydra

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



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

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

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

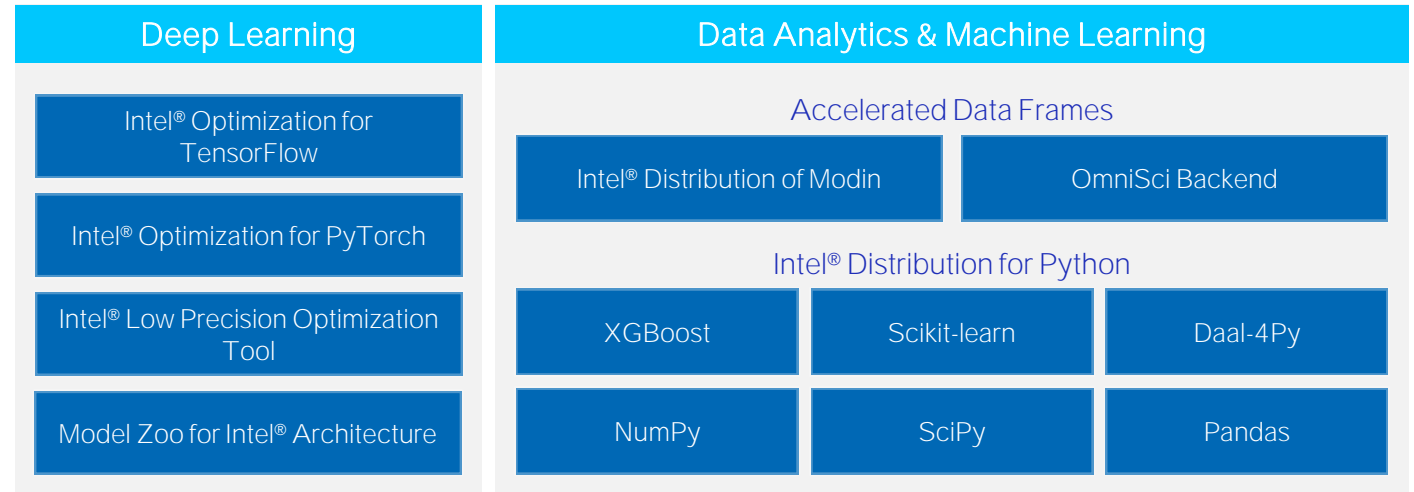
Accelerate end-to-end AI and data analytics pipelines with libraries optimized for Intel<sup>®</sup> architectures

## 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



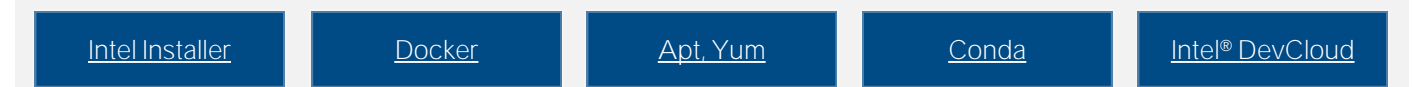
## Samples and End2End Workloads



Supported Hardware Architectures<sup>1</sup>

Hardware support varies by individual tool. Architecture support will be expanded over time. Other names and brands may be claimed as the property of others.

## Get the Toolkit [HERE](#) or via these locations



# Intel® Distribution of OpenVINO™ toolkit



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?

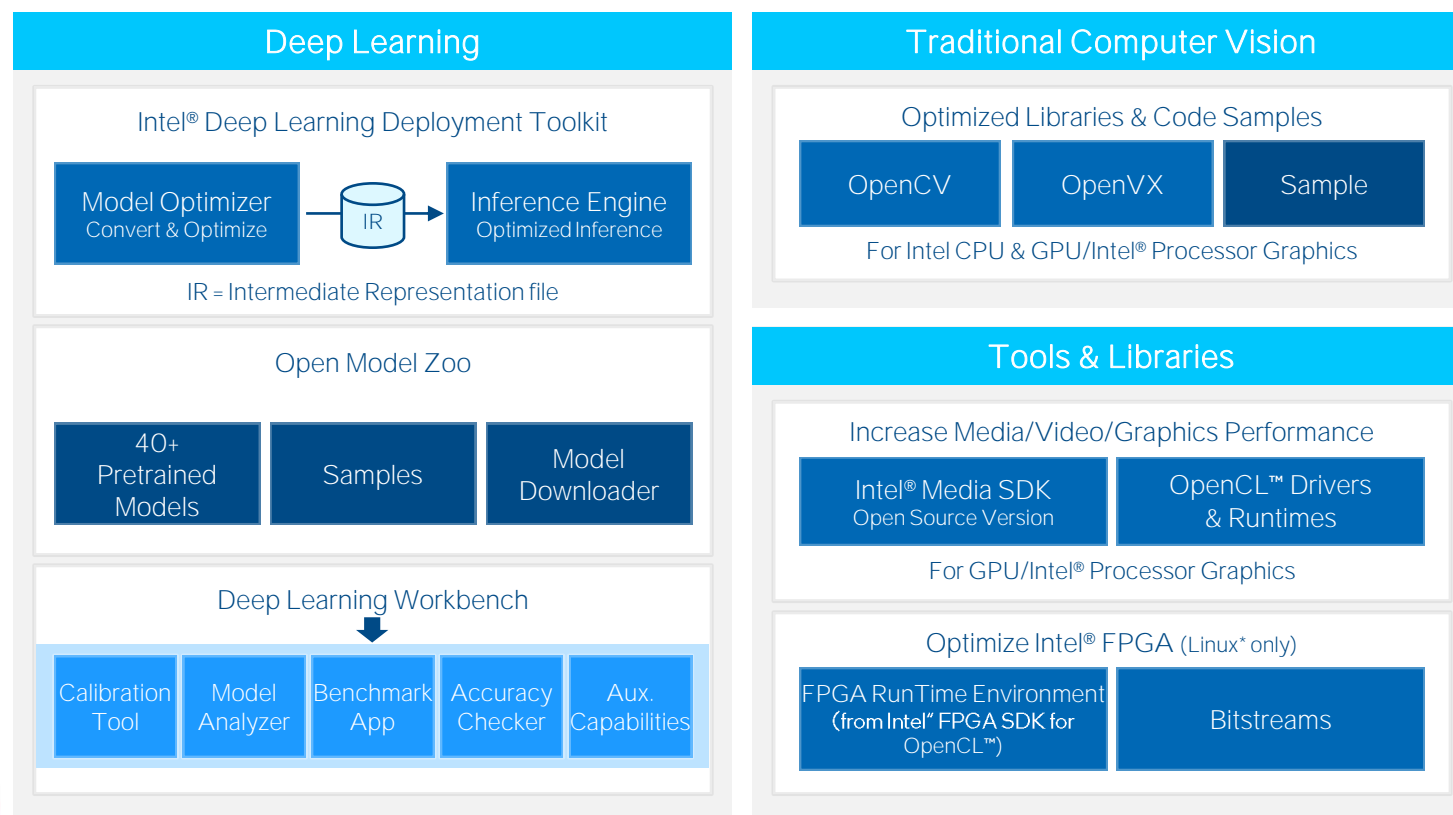
- Computer vision, deep learning software developers
- Data scientists
- OEMs, ISVs, System Integrators

## Usages

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



## Intel® Distribution of OpenVINO™ toolkit

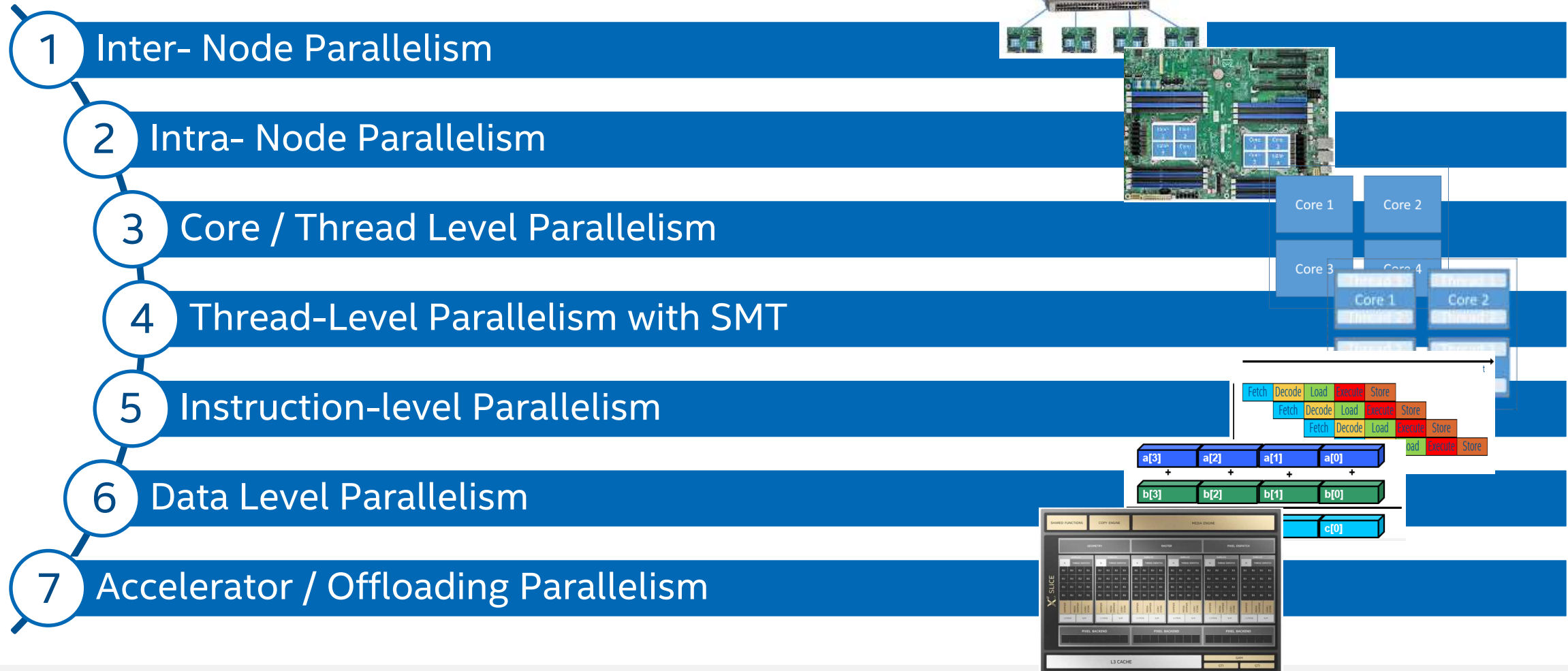


# Parallelism on the Intel Architecture

Enabled by Intel<sup>®</sup> oneAPI Products



# The Seven Levels of Parallelism



# Inter- Node Parallelism



# 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 communication-avoiding 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 😊

## Which Tools?

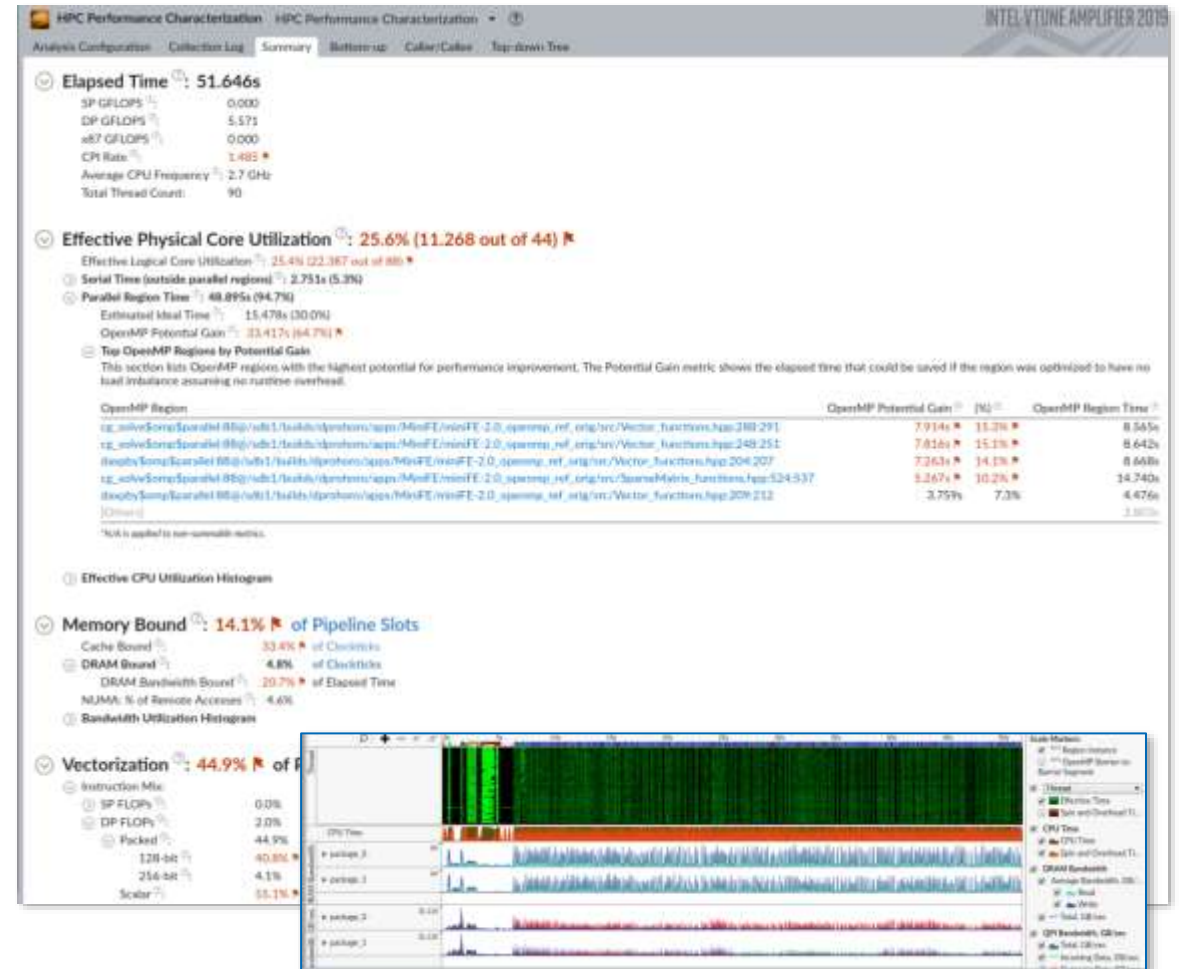
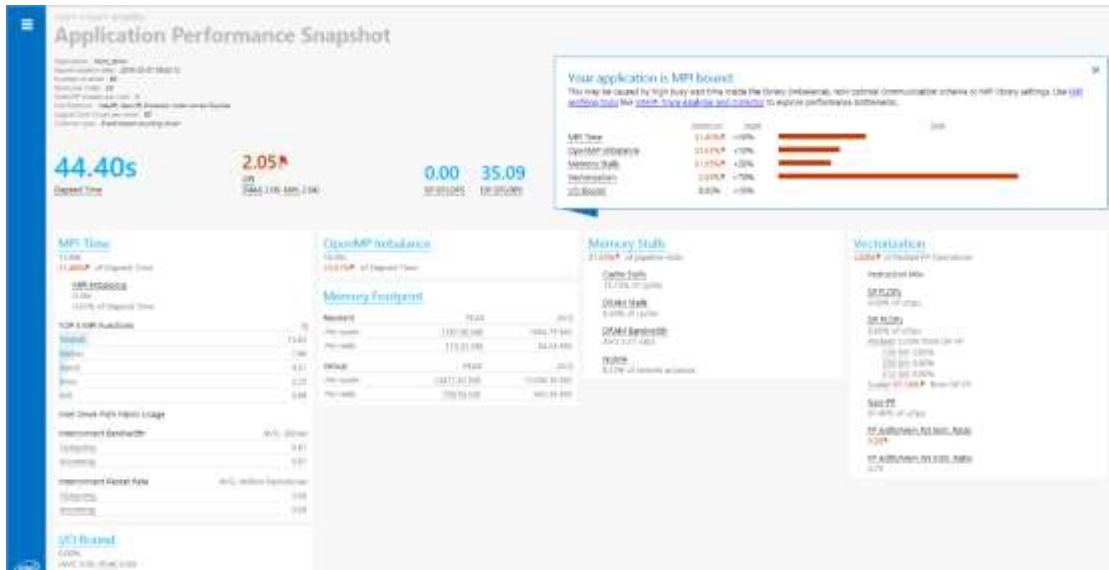
- Identify scalability issues with the **VTune™ Profiler** Application Performance Snapshot & the **Trace Analyzer and Collector**
- Fine Tune Collective Operations with **Intel MPI – autotuner**
- Where applicable
  - Math Kernel Library (oneMKL)
  - Data Analytics Library (oneDAL)
  - Collective Communications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

# Inter- Node Parallelism

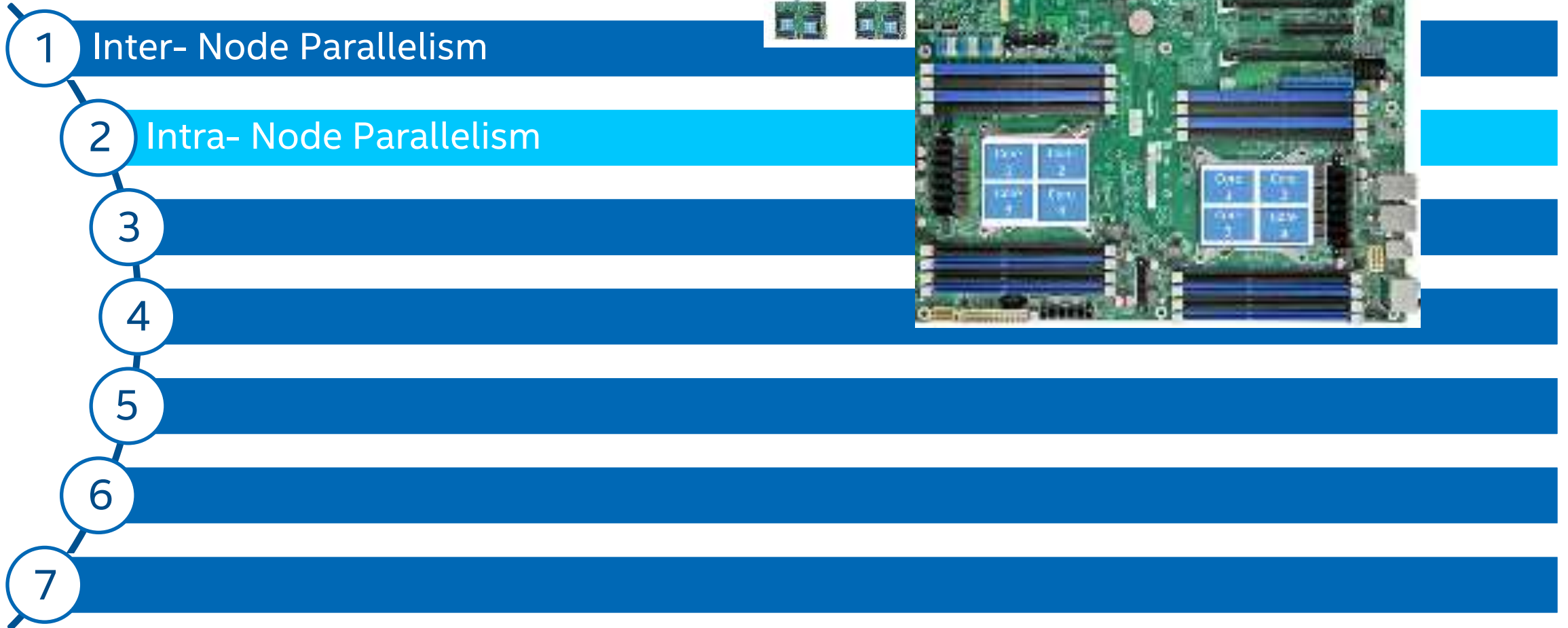
VTune HPC Analysis (right)

Application Performance

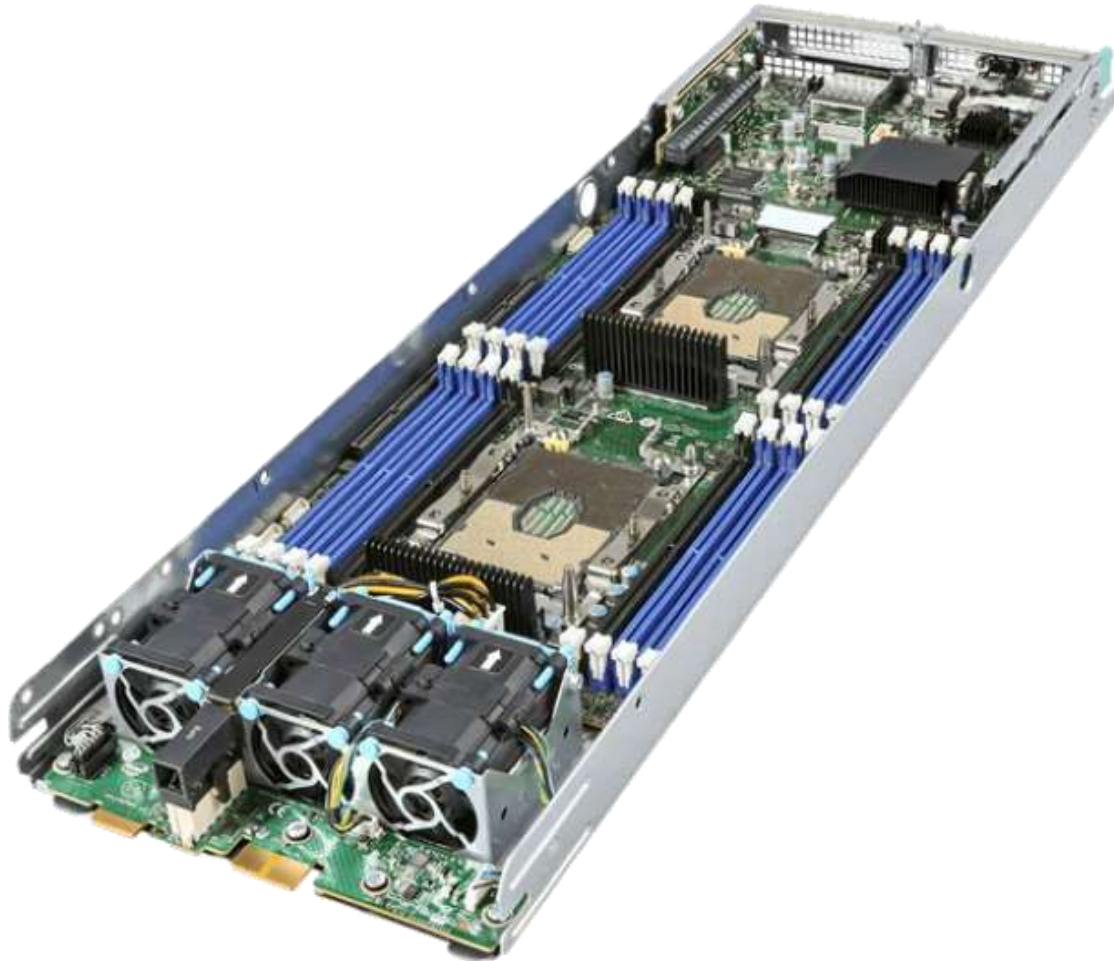
Snapshots HTML export (left)



# 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?

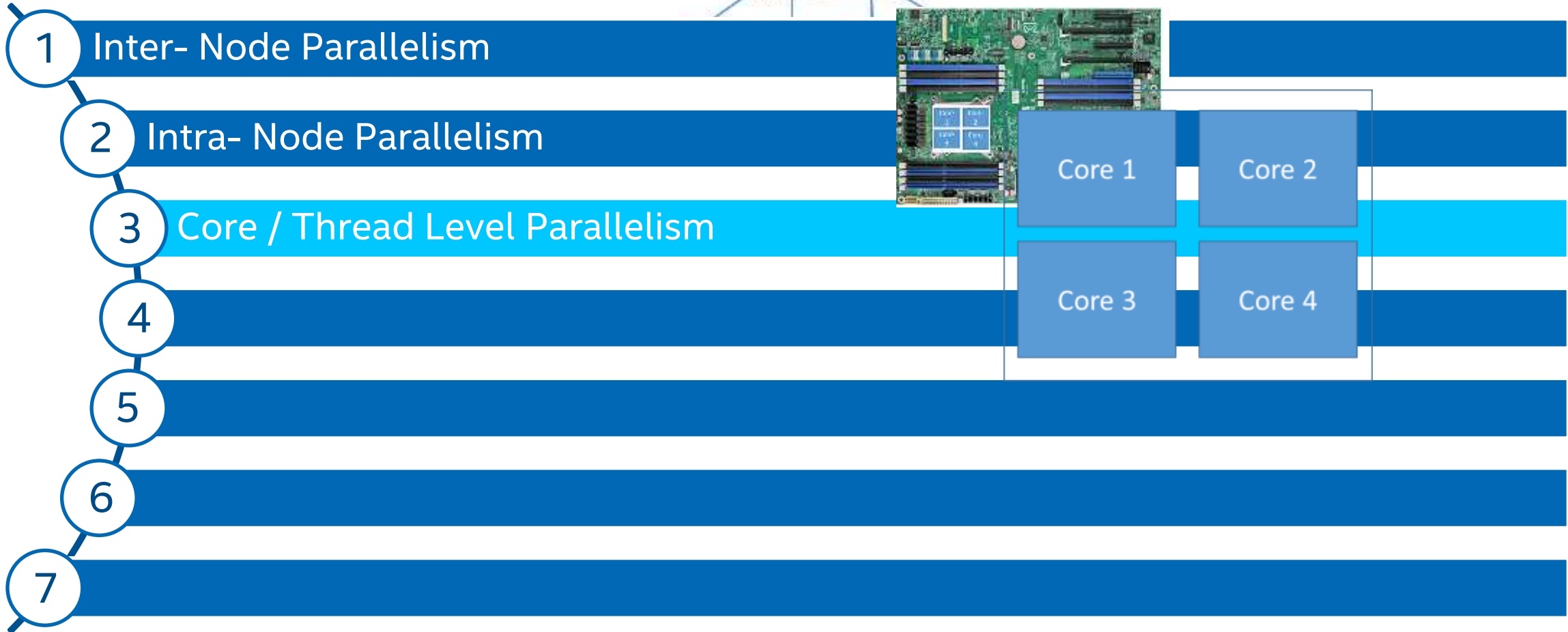
- Tune for best Memory Locality (NUMA optimization)
- Employ proper process pinning
- Avoid frequent involvement of Operating System (OS) Kernel e.g. by using scalable memory allocators

## Which Tools?

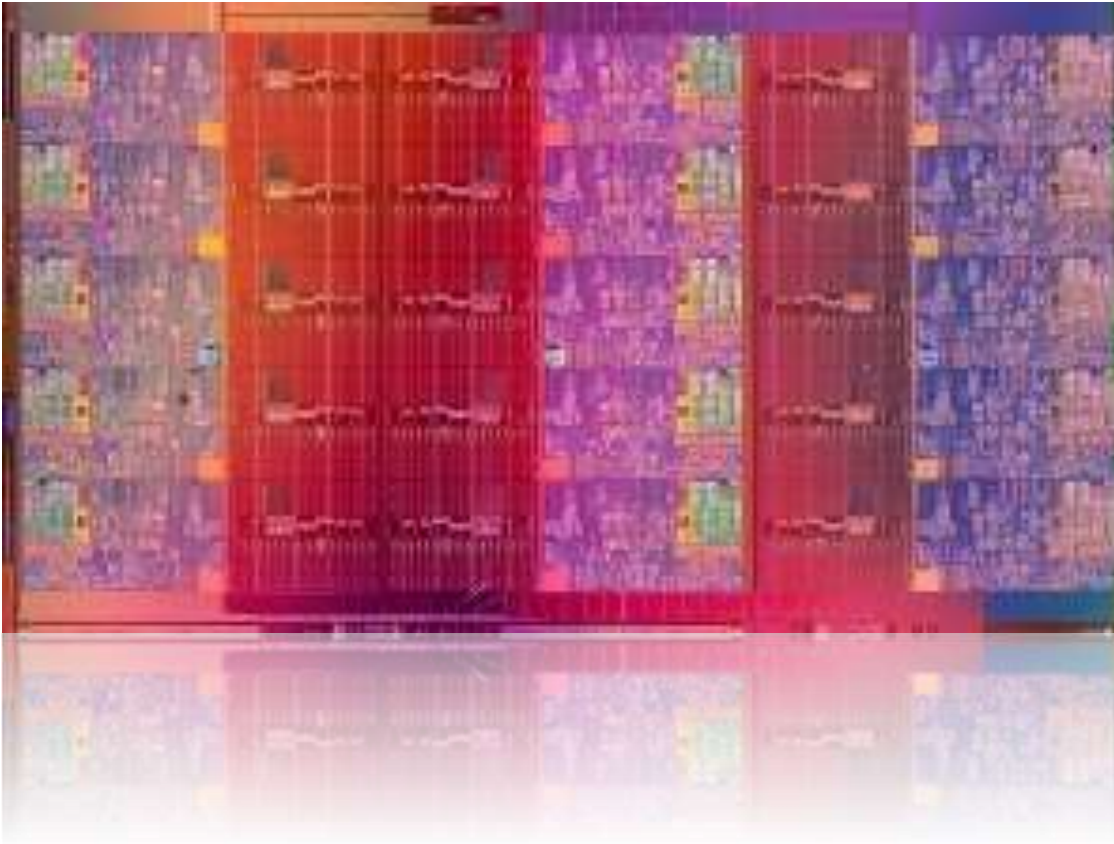
- Identify issues with the VTune™ Profiler
  - Hotspots
  - Memory Access
  - HPC Performance Characterization
- Identify issues with the VTune™ Profiler Application Performance Snapshot



# Core / Thread Level Parallelism



# 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

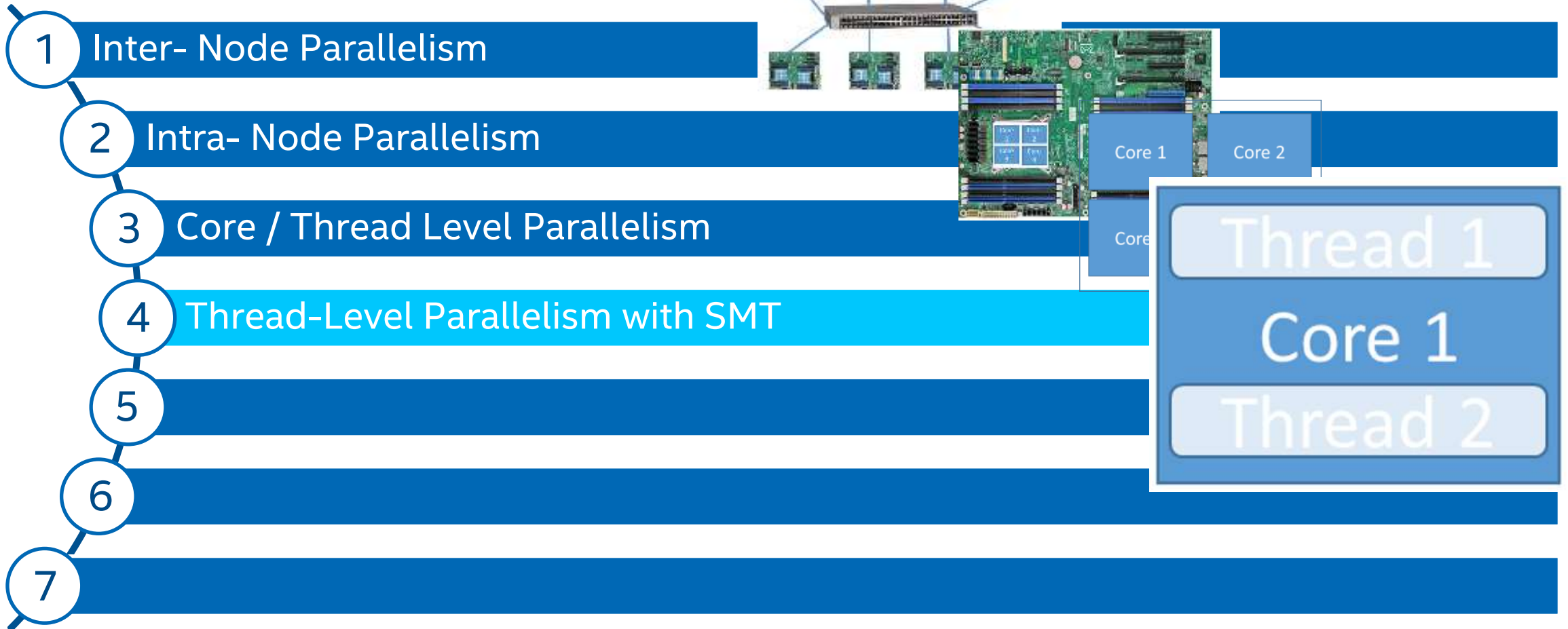
## What can I do?

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

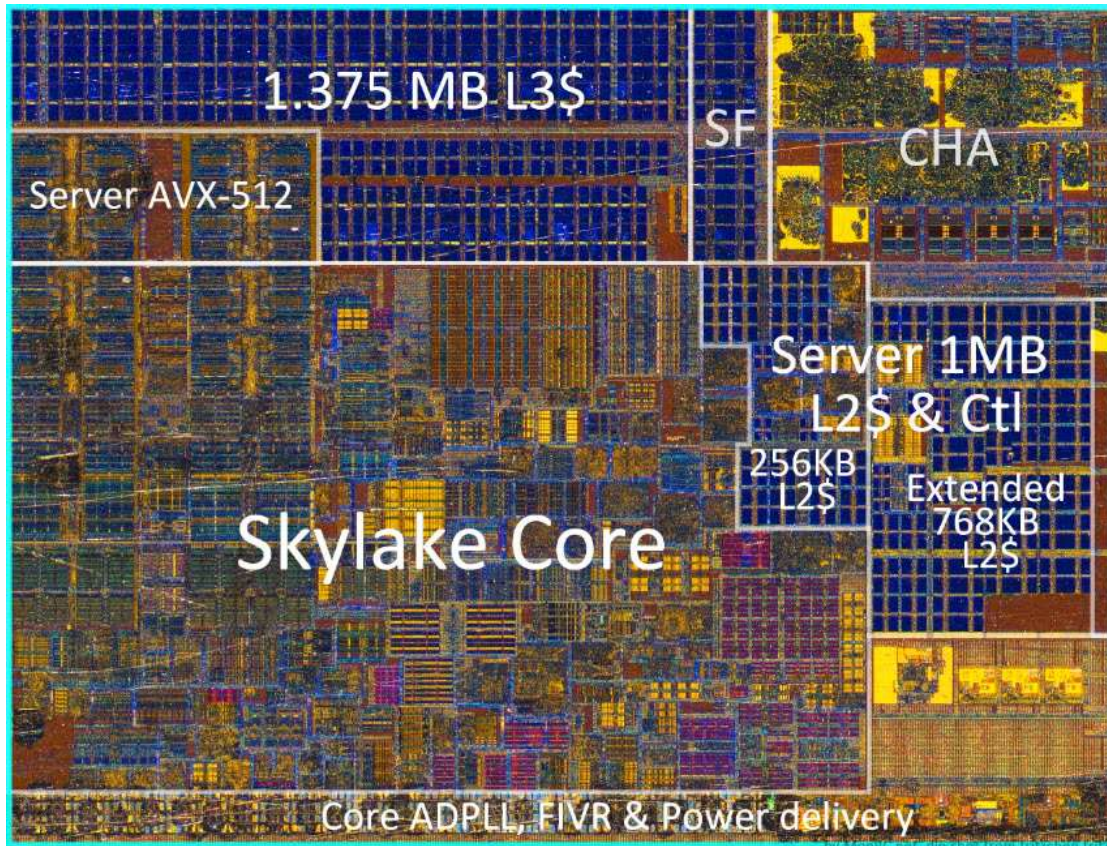
## Which Tools?

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

# Thread-Level Parallelism with Simultaneous MultiThreading (SMT)

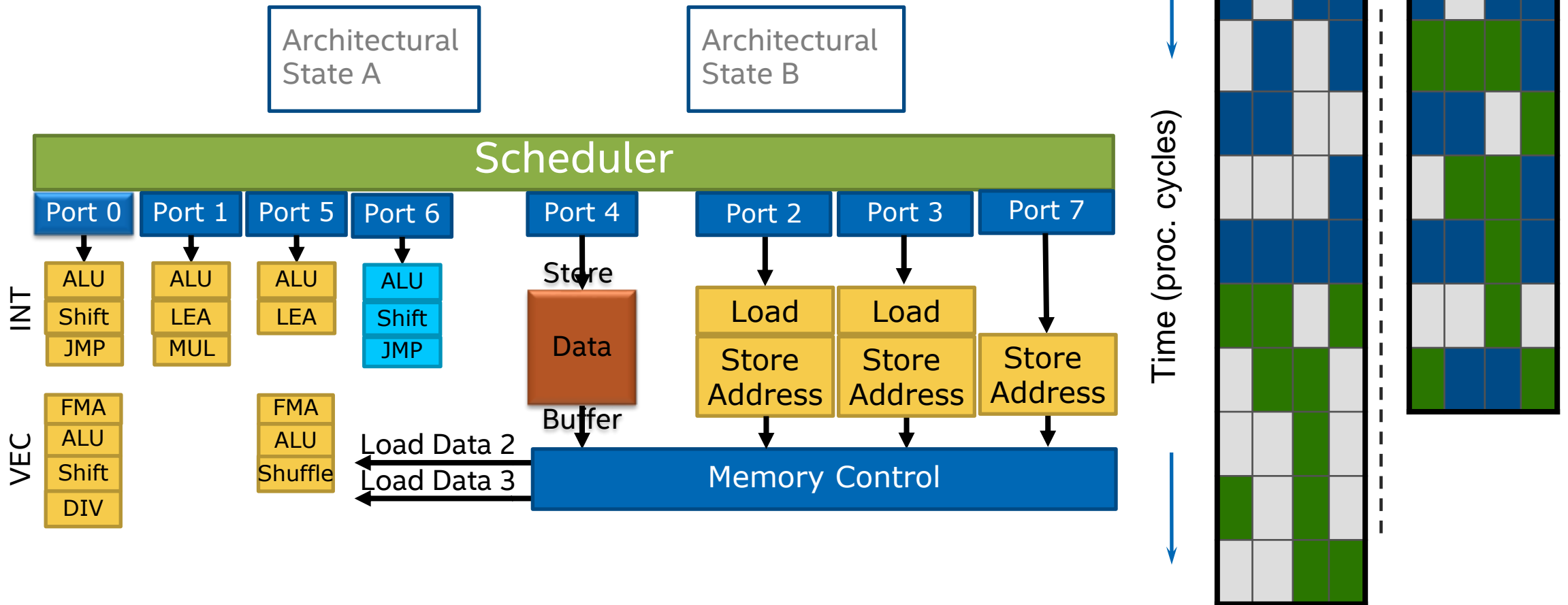


# Thread-Level Parallelism with Simultaneous MultiThreading (SMT)



- 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

# SMT also known as Hyperthreading



# Thread-Level Parallelism with Hyperthreading (SMT)

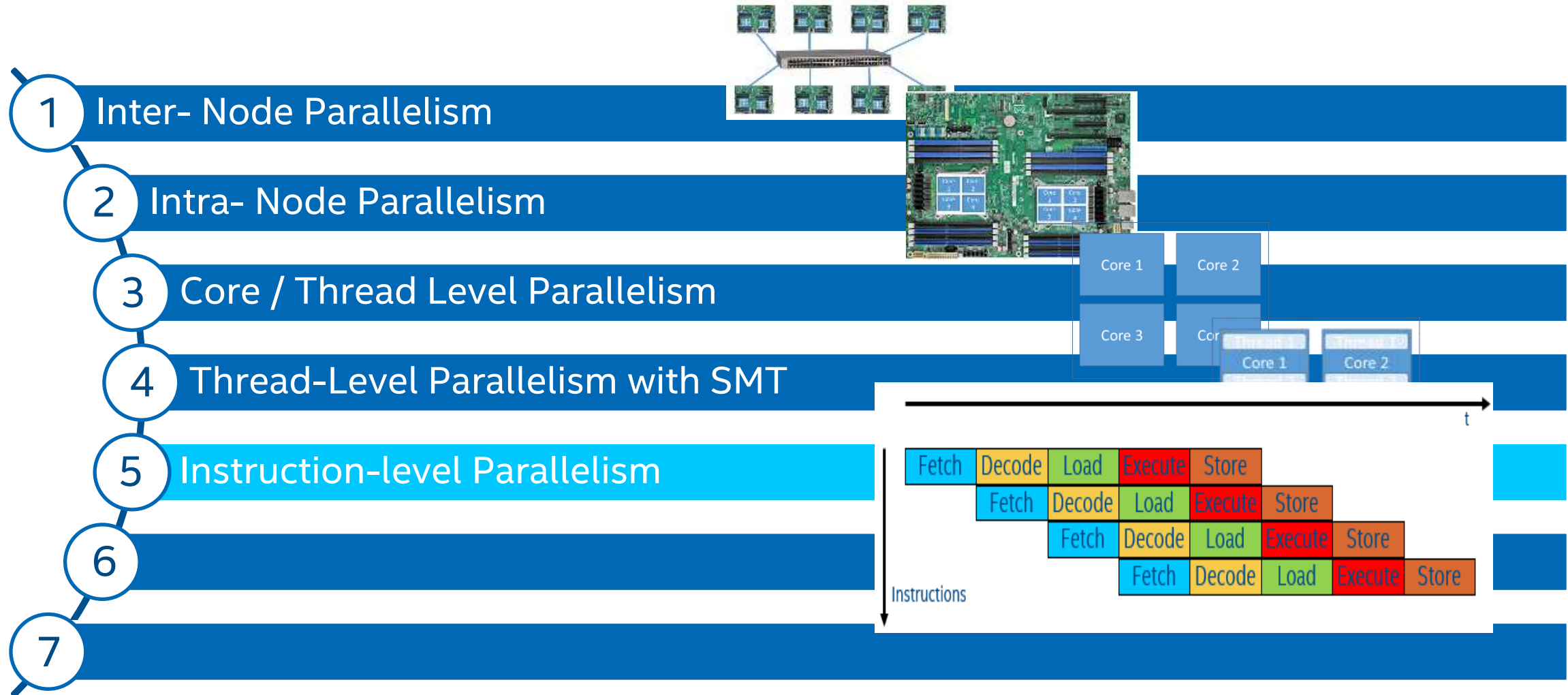
What can I do?

- 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

Which Tools?

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

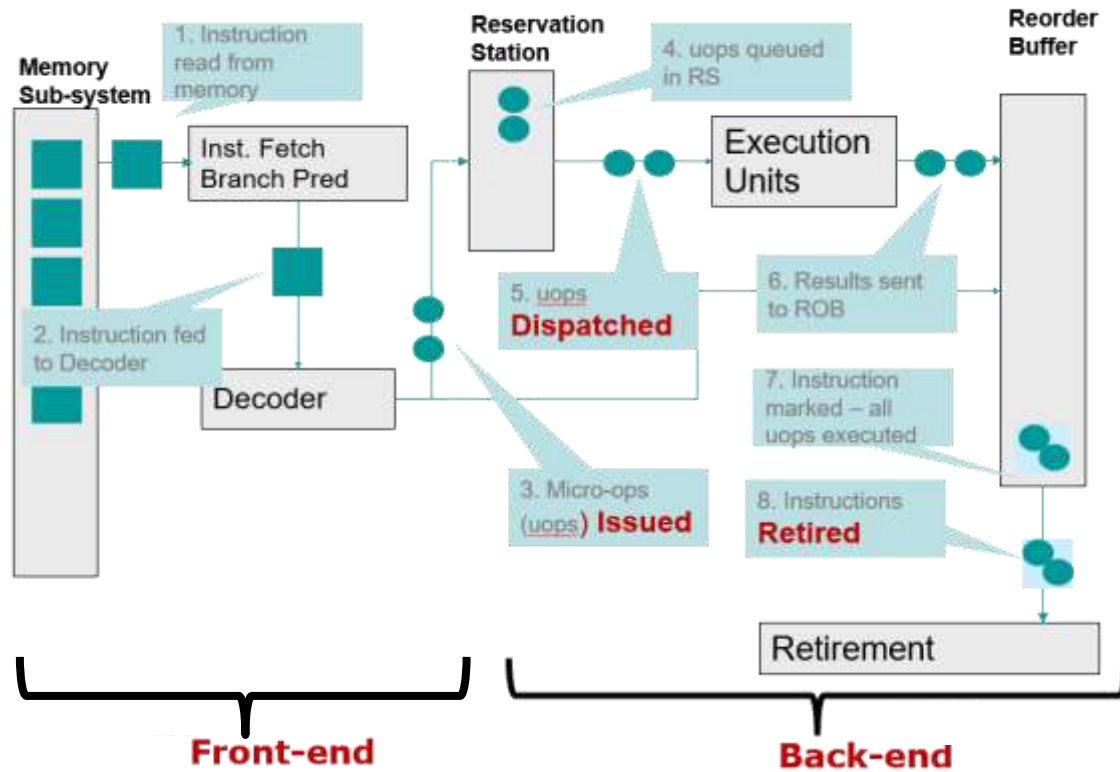
# Instruction-level Parallelism (ILP)





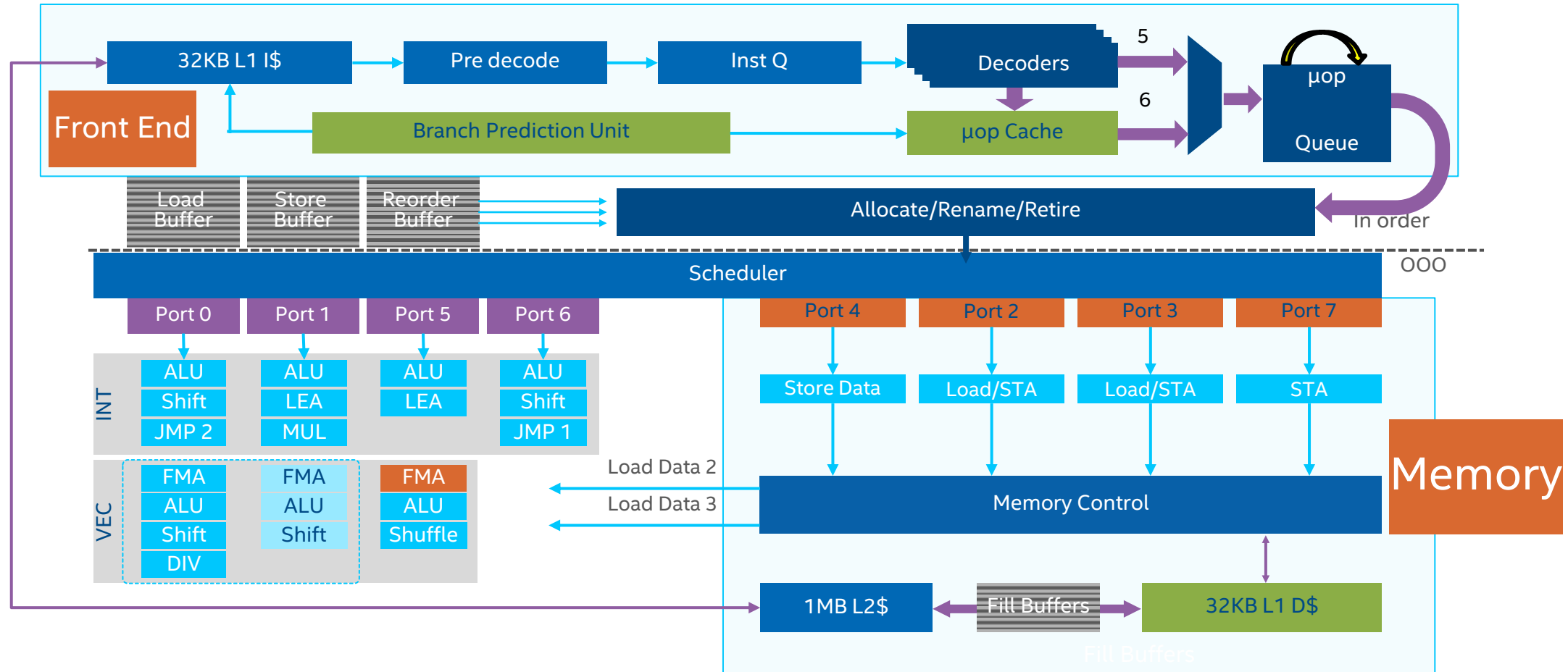
# Instruction-level Parallelism (ILP)

## 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)

# Instruction-level Parallelism (ILP)



# Instruction-level Parallelism (ILP)

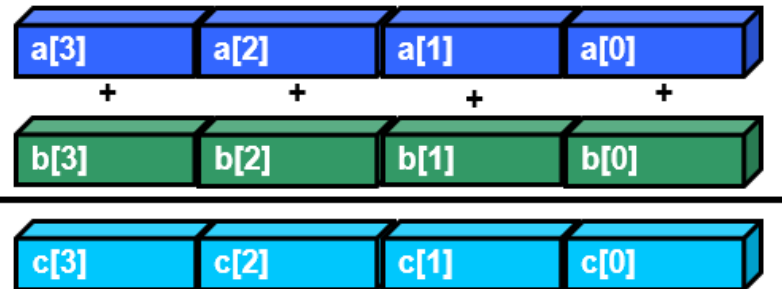
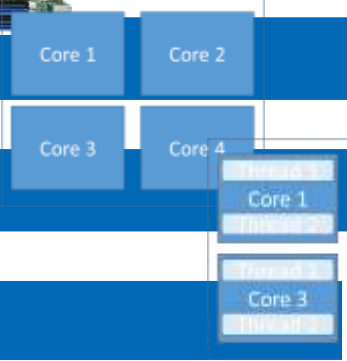
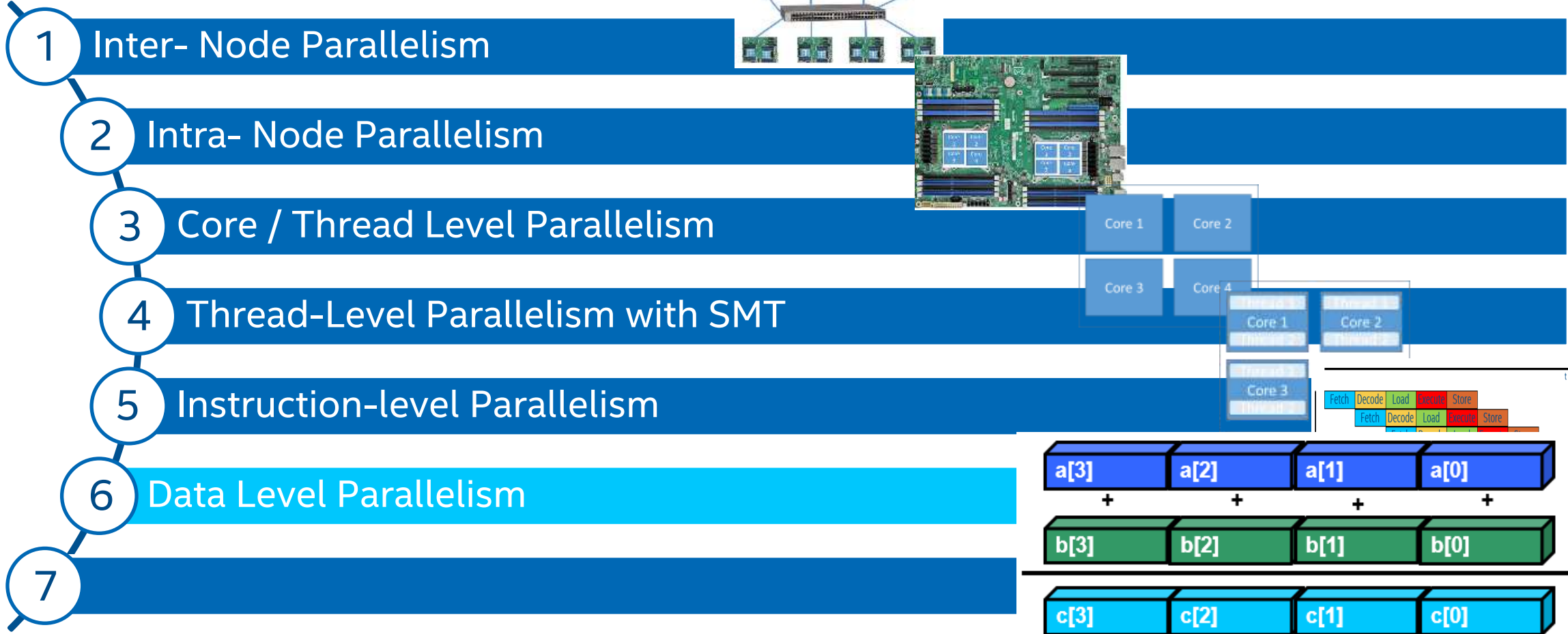
## What can I do?

- 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™ Profiler
  - Microarchitecture Analysis
- Intel® DPC++/C++ Compiler, Intel® C++ Compiler Classic, Intel® Fortran Compiler Classic, & Intel® Fortran Compiler (Beta)

# Data Level Parallelism

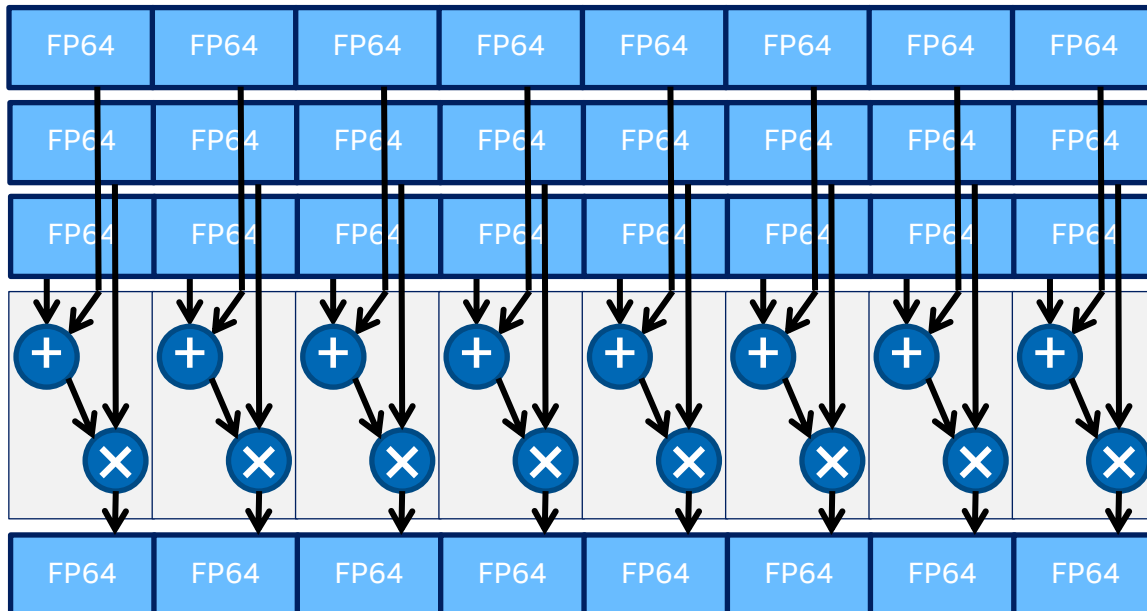


# Data Level Parallelism

Fused Multiply Add (FMA)

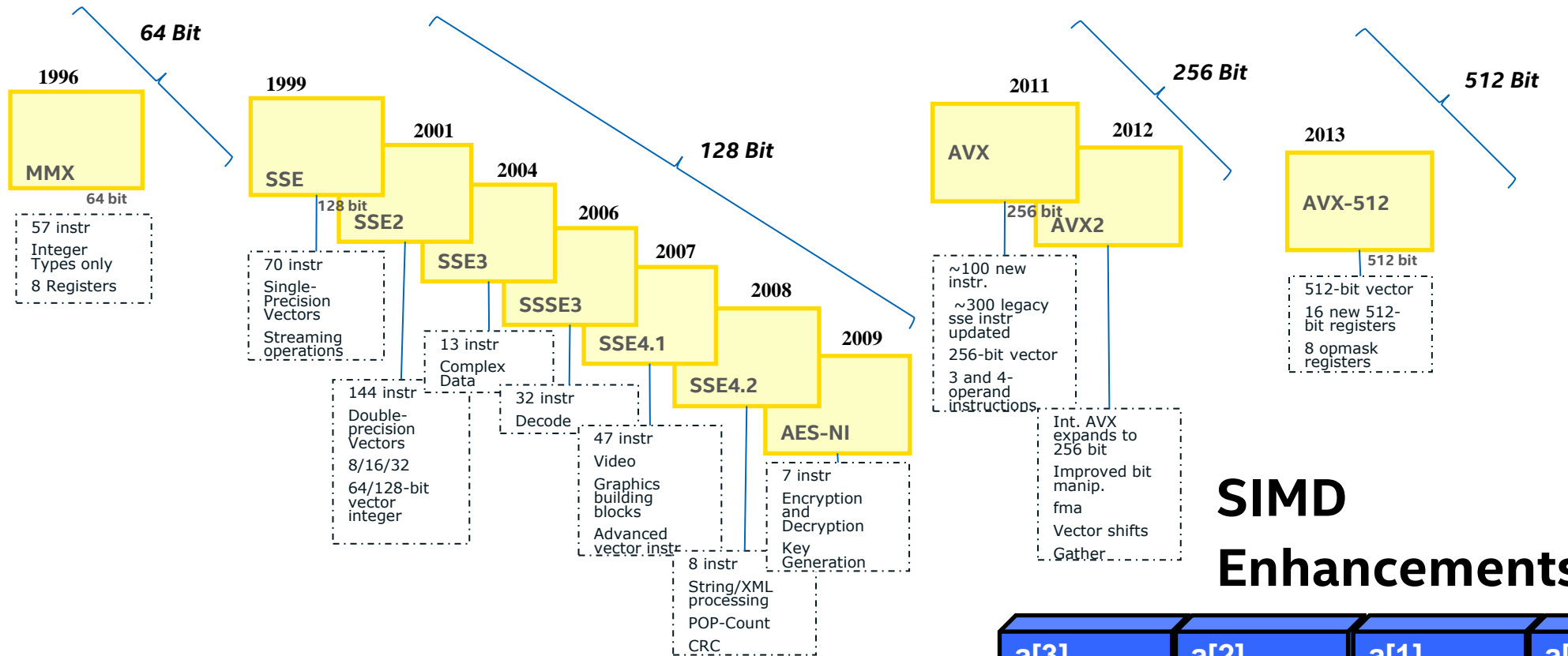
AVX-512 Vector Instruction

$$D = (A + C) \times B$$



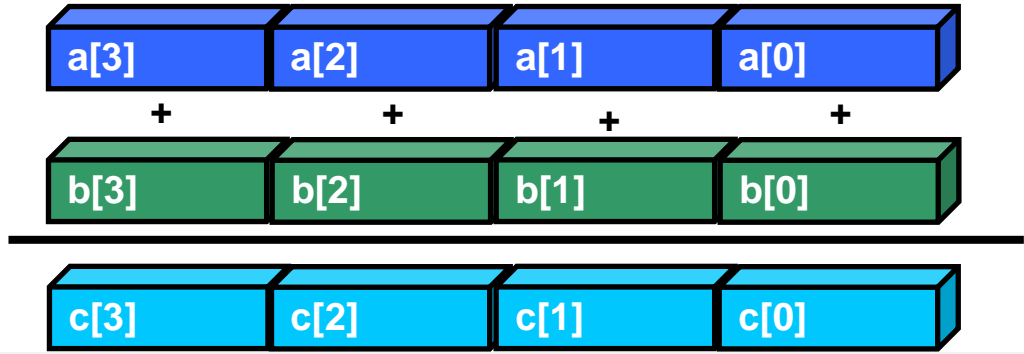
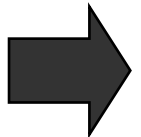
- X86 ISA Extensions
- Flynn's taxonomy: Single Instruction Multiple Data (SIMD)
- Code candidates: Loops – to split iterations in chunks of SIMD width – reducing the trip count
- Vectorization can happen implicitly with Compiler generating instructions

# Data Level Parallelism



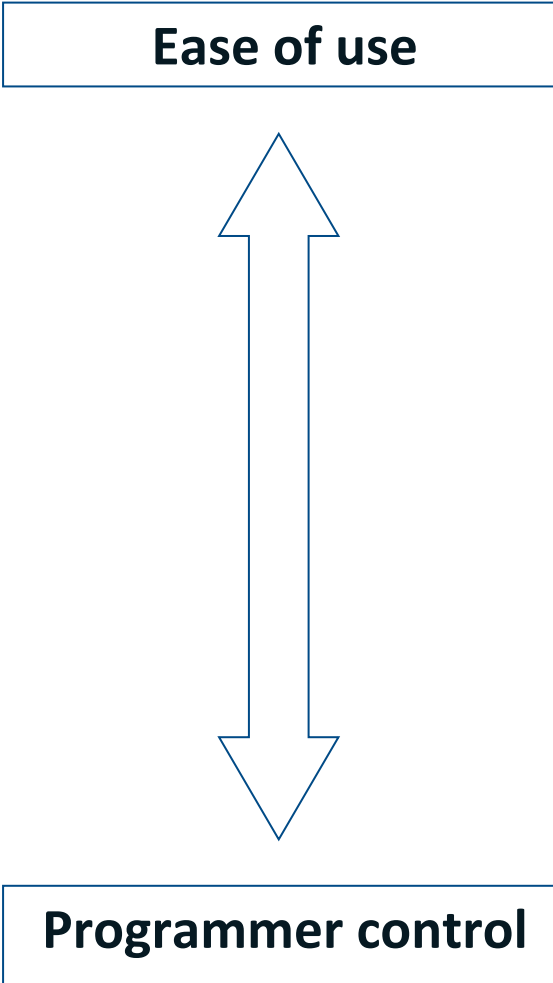
## SIMD Enhancements

```
for (i=0; i<MAX; i++)
    c[i]=a[i]+b[i];
```



# Data Level Parallelism

- 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)**



# Data Level Parallelism

## What can I do?

- Use the most advanced Vector ISA available on your machine – the compiler default is SSE2
- Add code pragmas to the non-vectorized 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

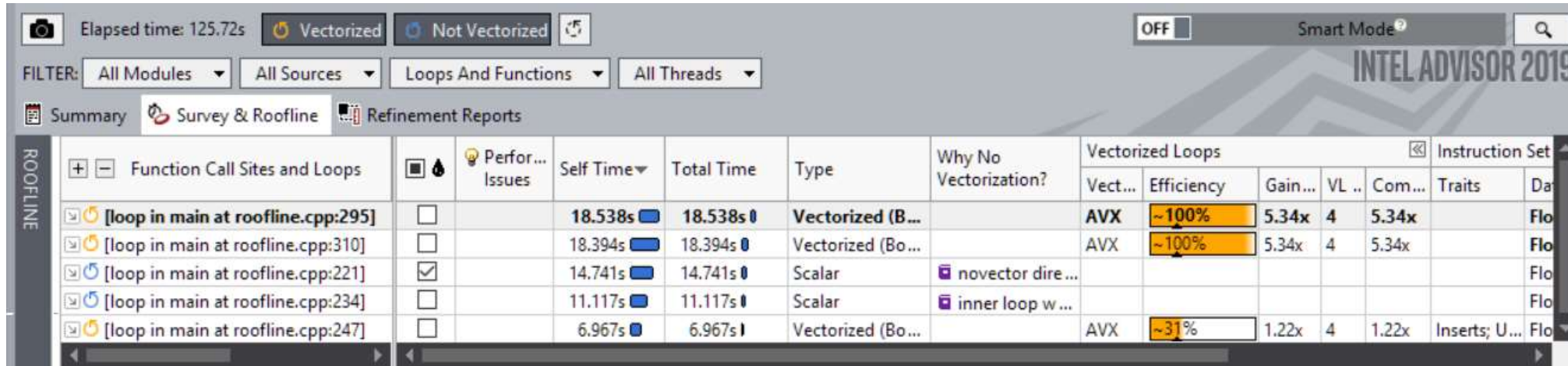
## Which Tools?

- Intel® DPC++/C++ Compiler, Intel® C++ Compiler Classic, Intel® Fortran Compiler Classic, & Intel® Fortran Compiler (Beta)
- Use the Intel® 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](#)



# Data Level Parallelism



Elapsed time: 125.72s | Vectorized | Not Vectorized | OFF | Smart Mode

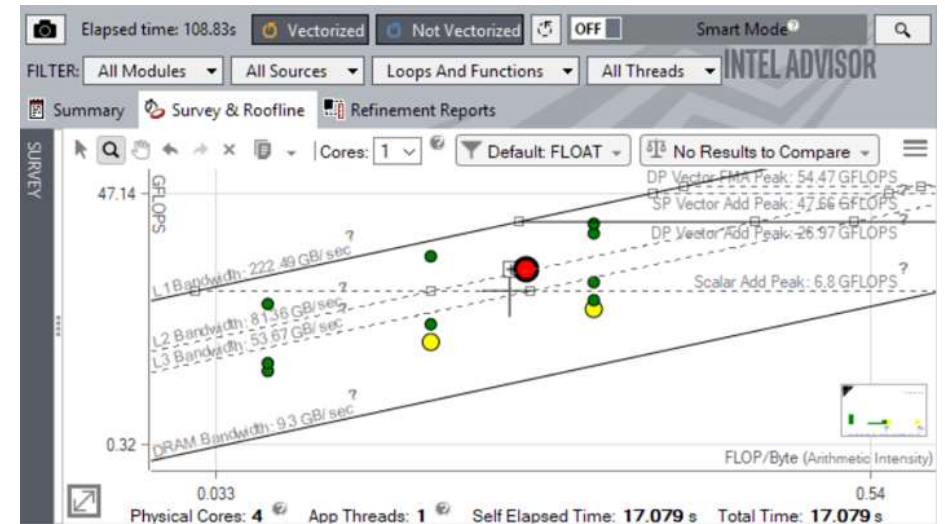
FILTER: All Modules | All Sources | Loops And Functions | All Threads

Summary | Survey & Roofline | Refinement Reports

Function Call Sites and Loops	Perfor... Issues	Self Time	Total Time	Type	Why No Vectorization?	Vectorized Loops				Instruction Set	
						Vect...	Efficiency	Gain...	VL ..	Com...	Traits
[loop in main at roofline.cpp:295]	<input type="checkbox"/>	18.538s	18.538s	Vectorized (B...		AVX	~100%	5.34x	4	5.34x	Flo
[loop in main at roofline.cpp:310]	<input type="checkbox"/>	18.394s	18.394s	Vectorized (Bo...		AVX	~100%	5.34x	4	5.34x	Flo
[loop in main at roofline.cpp:221]	<input checked="" type="checkbox"/>	14.741s	14.741s	Scalar	novector dire ...						Flo
[loop in main at roofline.cpp:234]	<input type="checkbox"/>	11.117s	11.117s	Scalar	inner loop w ...						Flo
[loop in main at roofline.cpp:247]	<input type="checkbox"/>	6.967s	6.967s	Vectorized (Bo...		AVX	~31%	1.22x	4	1.22x	Inserts; U... Flo

Advisor Vectorized Loop Summary (top-left)

Advisor Roofline (bottom right)



# Data Level Parallelism – legacy instructions



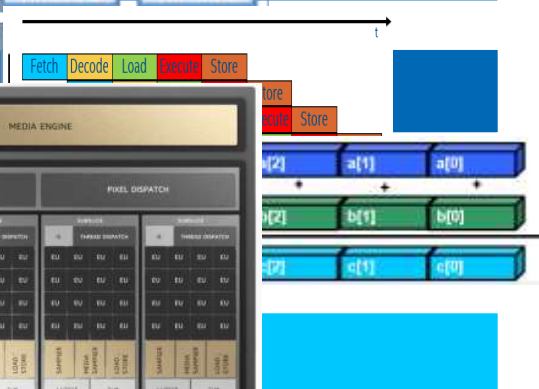
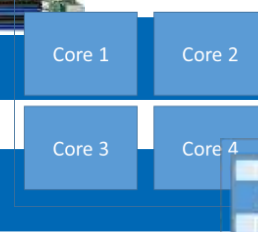
What is / was that?

## 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

# Accelerator / Offloading Parallelism

- 1 Inter- Node Parallelism
- 2 Intra- Node Parallelism
- 3 Core / Thread Level Parallelism
- 4 Thread-Level Parallelism with SMT
- 5 Instruction-level Parallelism
- 6 Data Level Parallelism
- 7 Accelerator / Offloading Parallelism



# Accelerator / Offloading Parallelism



- 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

# Accelerator / Offloading Parallelism

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;
}
```

## ■ Intel® oneAPI Accelerator Programming

- C/C++
  - Intel® DPC++ Compiler
  - Intel® C++ Compiler with OpenMP Offloading
- Fortran
  - Intel® Fortran Compiler with OpenMP Offloading

# Accelerator / Offloading Parallelism

What can I do?

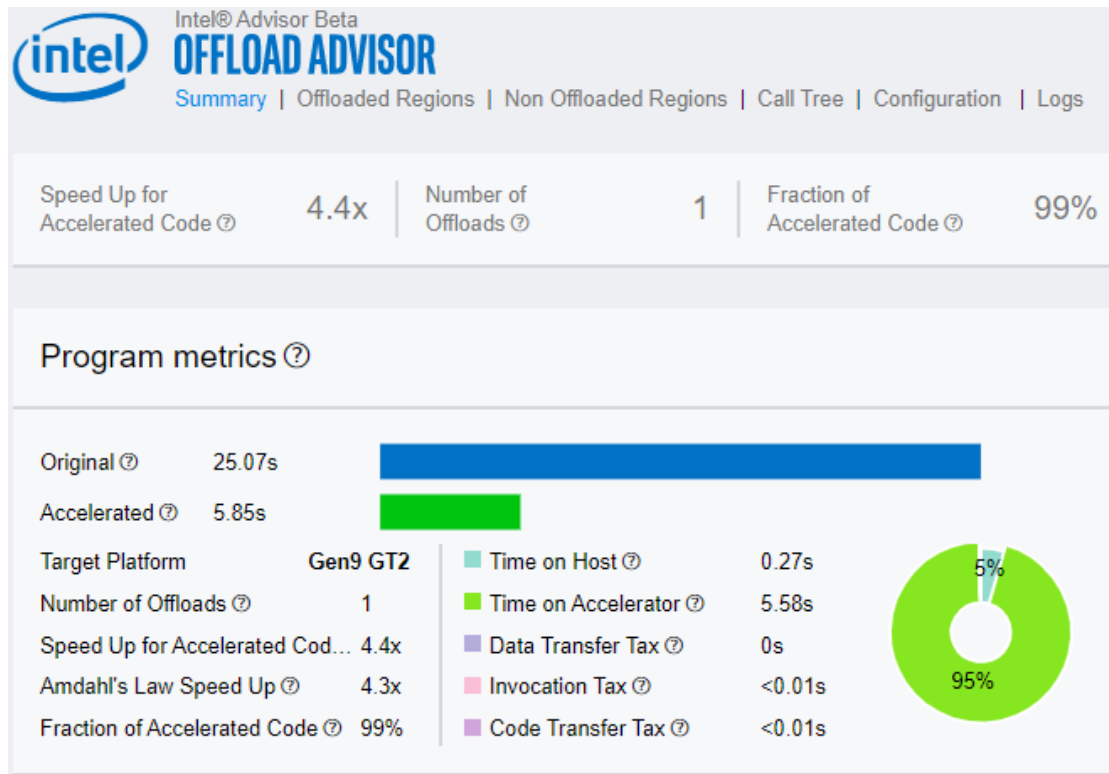
- Enable your code today for Intel Integrated Graphics GPUs and upcoming datacenter GPGPUs
- Reduce data transfers between host and device
- Minimize communication / synchronization in between individual threads

Which Tools? – Intel® oneAPI

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

# Accelerator / Offloading Parallelism

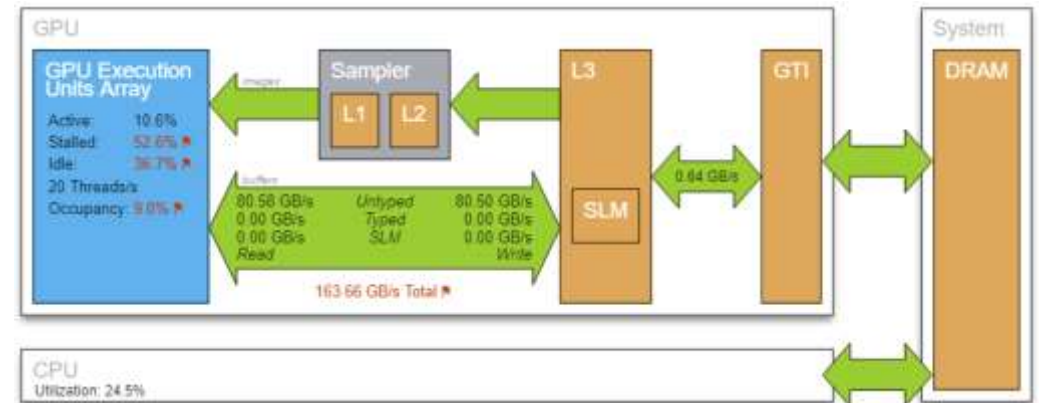
## Offload Advisor – Efficiency



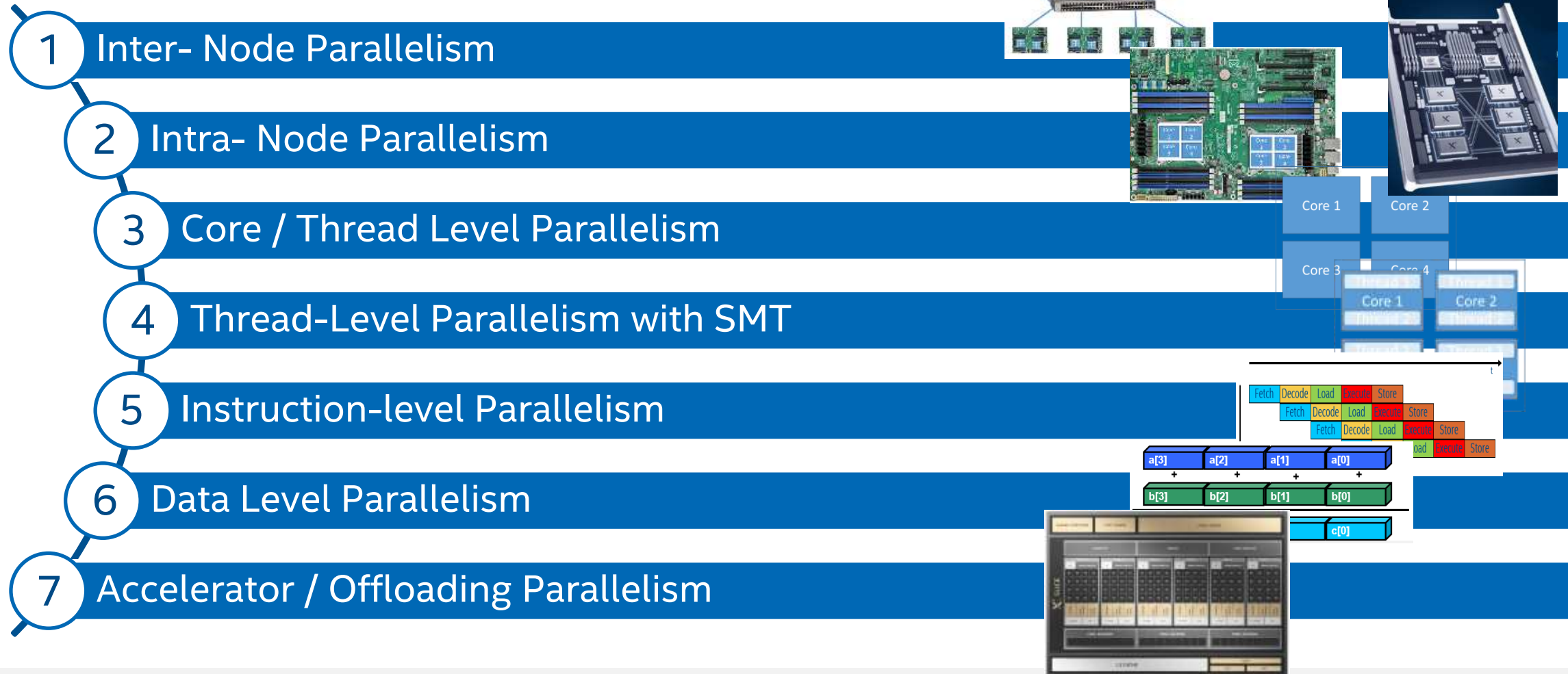
## VTune – GPU Offload Analysis

Source	GPU Instructions Executed by Instruction T...
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 dz = ptr[j].pos[2] - ptr[i].pos[2]	12,500,000
161	
162 distanceSqr = dx*dx + dy*dy + dz*dz	87,500,000
163 distanceInv = 1.0 / sqrt(distanceSqr)	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

Legend: GPU Instructions Executed by Instruction T...  
 Control Flow (Red), Send & Wait (Blue), Int32 & SP Float (Yellow), Int64 & DP Float (Orange), Other (Grey)



# The Seven Levels of Parallelism

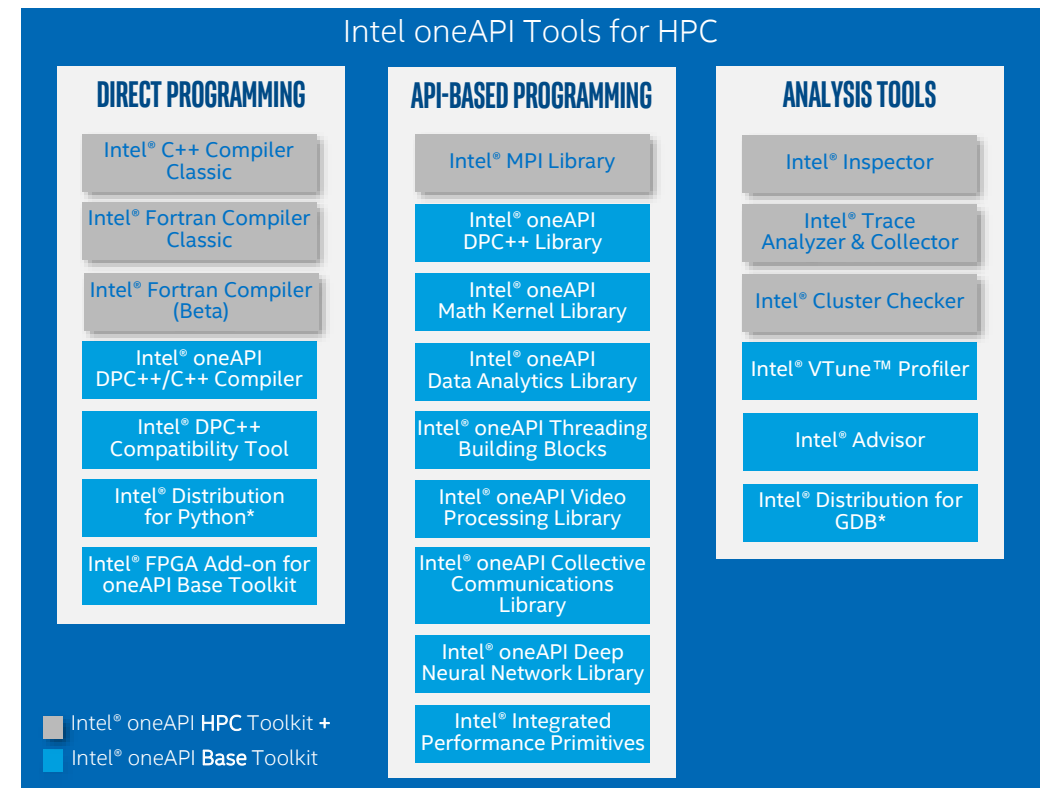




# The Seven Levels of Parallelism

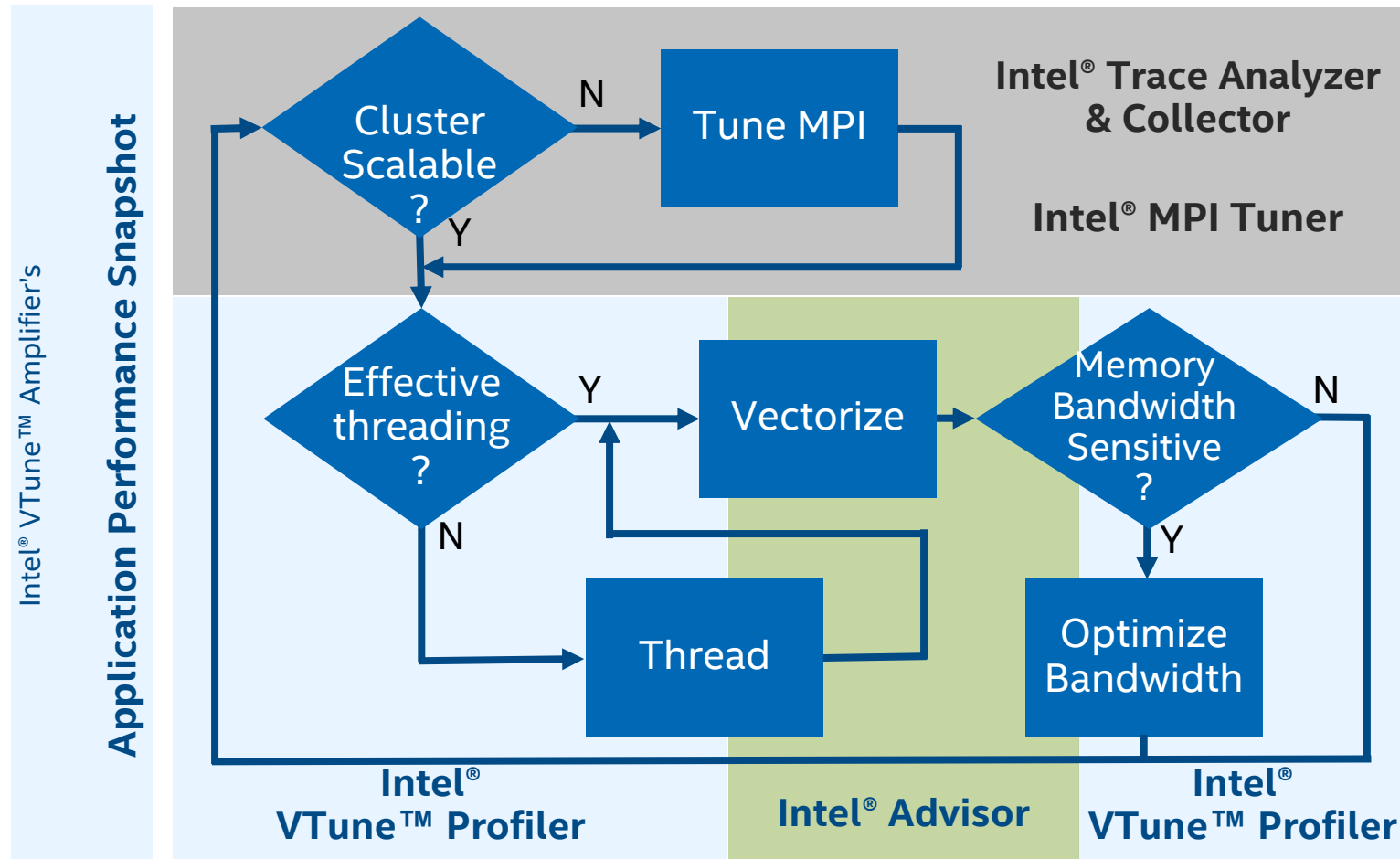
Parallelism on the Intel Architecture
Inter- Node Parallelism
Intra- Node Parallelism
Core / Thread Level Parallelism
Thread-Level Parallelism with SMT
Instruction-level Parallelism
Data Level Parallelism
Accelerator / Offloading Parallelism

Supported by the  
Intel® oneAPI Base & HPC Toolkits



# Performance Analysis Tools for Diagnosis

Intel® Parallel Studio



# Summary

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.

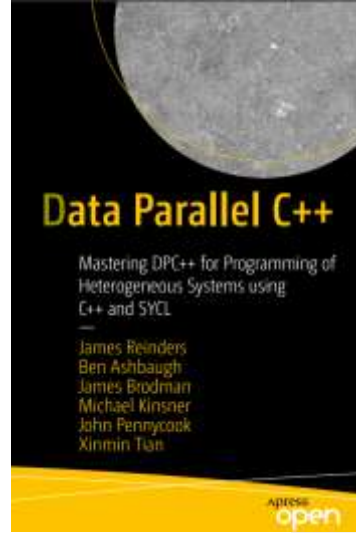
To achieve best CPU performance, make sure your programs are efficiently vectorized and are well parallelized.

Especially the latter one also counts for GPUs.

# (Recorded) Tools Webinars

- **Intel® 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® oneAPI Rendering Toolkit Webinar ( 1 day) :**
  - Rendering 08.06.2021:  
<https://www.ai-spektrum.de/veranstaltungen/intel-oneapi-rendering-toolkit-workshop.html>
  - Make use of simulated data and virtualize them in high fidelity photo realistic fashion
- **Intel® oneAPI AI Analytics Toolkit webinar – 1 day :**
  - AI webinar on 15.06.2021:  
<https://www.ai-spektrum.de/veranstaltungen/artificial-intelligence-on-intelr-platforms-using-intelr-oneapi-ai-analytics-toolkit-opensvino-workshop.html>
  - Intel performance optimized AI tools kits for classic and machine learning

# Selected Links : Intel Tools User's Manuals



- **Intel® 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® 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>

# 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'

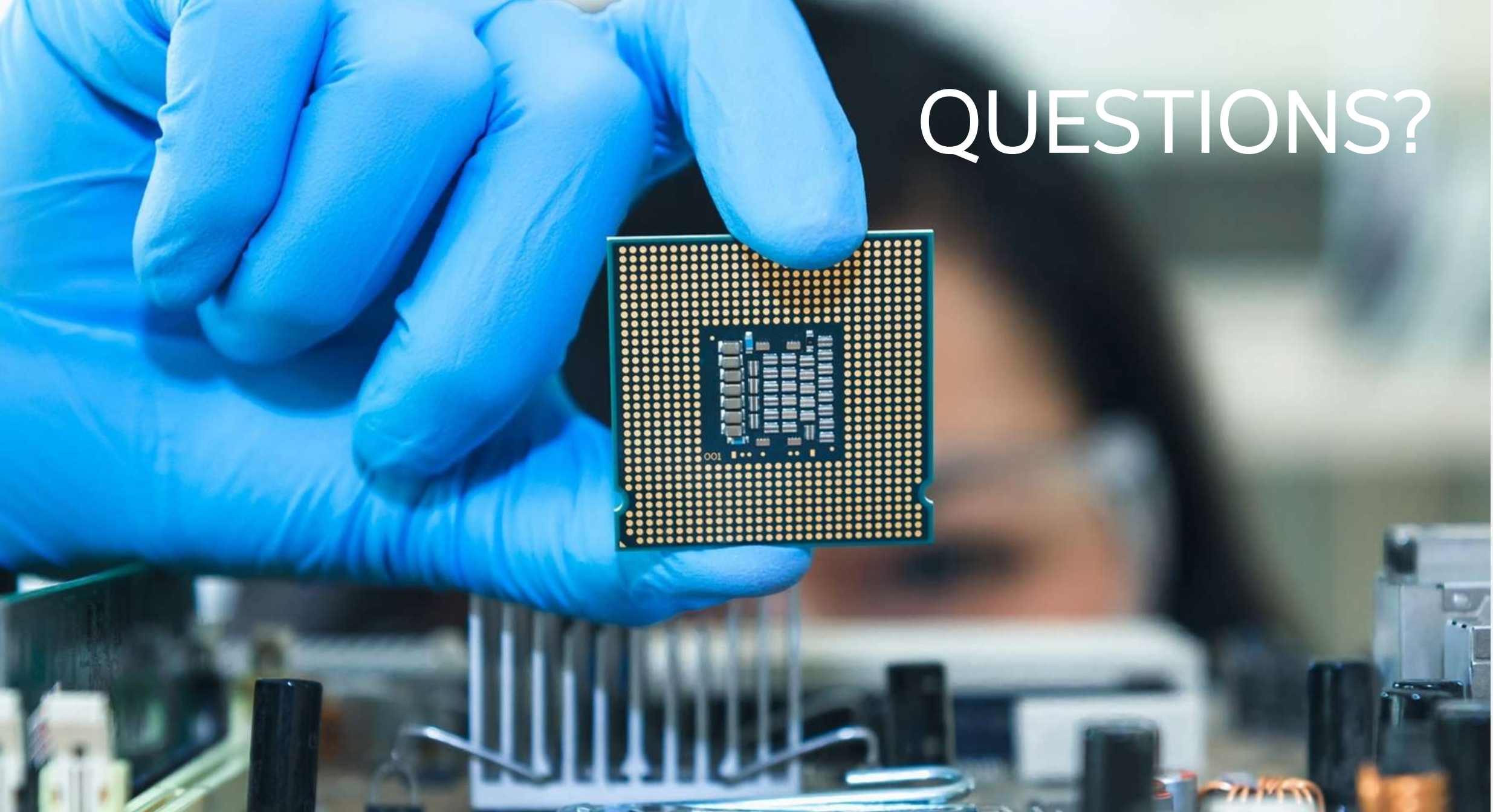
<https://www.intel.com/content/www/us/en/developer/community/parallel-universe-magazine/overview.html?s=Newest>

- Intel Developer Zone

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



# QUESTIONS?



intel®