



**Hewlett Packard**  
Enterprise

# **The Journey To Exascale Computing And Beyond**

---

Seminar at the Politecnico di Milano

Alfio Lazzaro, HPE HPC/AI EMEA Research Lab, [alfio.lazzaro@hpe.com](mailto:alfio.lazzaro@hpe.com)

May 15, 2024

# Who I am

- PhD in Experimental High Energy Physics @ University of Milan (Italy), 2007
  - Working at the BaBar experiment @ SLAC laboratory (Menlo Park, CA)
- Since then, I have been working in High Performance Computing (HPC)
  - Numerical algorithms, optimizations, parallelization, computing devices...



2009-2012



2012-2014



2015-2016



University of  
Zurich<sup>UZH</sup>

2016-2018



Hewlett Packard  
Enterprise

2018-∞



**CSCS**

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre



# HPE HPC/AI EMEA Research Lab (part of the HP Labs)



**Partnering with leading organizations in the EMEA region to advance supercomputing R&D**

## Our Role

- Deep technical collaboration with industry, academia, and public sector
- Long term technical relationships surrounding research, co-design, and operational support
- Focus on new technologies, driving HPE products
- Create reusable PoCs & European IP

## Research Interests

- HPC, Cloud, AI, Quantum
- Data movement, analysis, and workflows
- Heterogeneous computing and novel accelerators
- Programming languages and models
- Compilers and mathematical optimisation
- Performance portability, security, and containerisation
- Energy efficiency and sustainability

## Engagement Models

- Centres of Excellence
- Advanced Collaboration Centres
- Value-add projects
- Joint-funded research projects
- Nationally/internationally funded research projects
- Ph.D. and Placements

[https://www.hpe.com/emea\\_europe/en/compute/hpc/emea-research-lab.html](https://www.hpe.com/emea_europe/en/compute/hpc/emea-research-lab.html)

## Disclaimer

- I'm speaking solely for myself, not for my company or as a formal representative of my company

- **Spoiler Alert:** This is NOT a technical presentation (at least as I could image)

**the 10-Megabyte Computer System**



**Only \$5995**  
COMPLETE

**New From IMSAI®**

- 10-Megabyte Hard Disk
- 5 1/4" Dual-Density Floppy Disk Back-up
- 8-Bit Microprocessor (Optional 16-bit Microprocessor)
- Memory-Mapped Video Display Board
- Disk Controller
- Standard 64K RAM (Optional 256K RAM)
- 10-Slot S-100 Motherboard
- 28-Amp Power Supply
- 12" Monitor
- Standard Intelligent 82-Key ASCII Keyboard (Optional Intelligent 86-Key ASCII Extended Keyboard)
- 132-Column Dot-Matrix Printer
- CP/M\* Operating System

*You Read It Right ...  
All for \$5995!*

**IMSAI®** ...Thinking ahead for the 80's

415/635-7615 Computer Division of the Fischer-Freitas Corporation  
910 81st Avenue, Bldg. 14 • Oakland, CA 94621

\*CP/M is a trademark of Digital Research. Im Sai is a trademark of the Fischer-Freitas Corporation



# Outline

---

- Introduction
  - HPC and Supercomputing
  - Performance trend over the years
- The Exascale Era of Computing
  - Hardware and Software challenges
- Conclusion



# Introduction

---

*Where is the 'any' key?*

– Homer Simpson (in response to the message “Press any key”, E7 S7)



# High Performance Computing

- *“High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business.”*

<https://insidehpc.com/hpc-basic-training/what-is-hpc/>

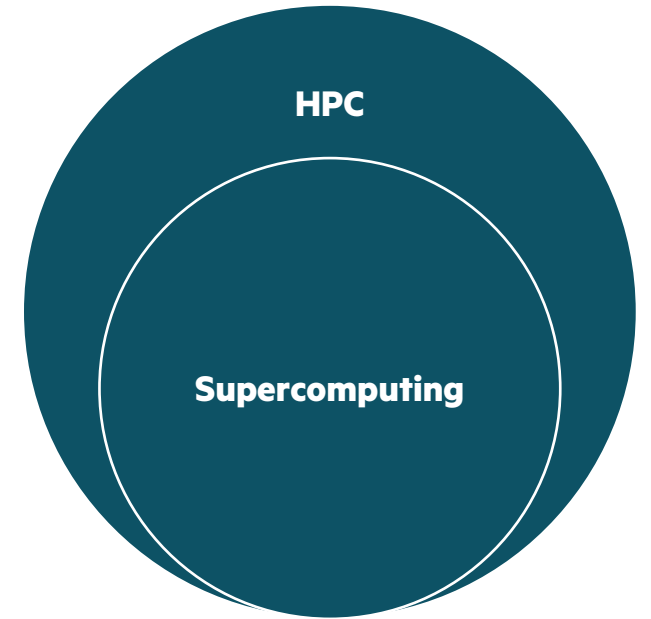
- Several performance metrics can be optimized
  - Latency: faster and faster application execution
  - Throughput: more and more work done
  - Energy: consume less energy to carry out a given task (Sustainable HPC)

- HPC is everywhere there is a computing system!



# Supercomputing

- Supercomputing uses HPC techniques to build a single tightly-coupled system
  - A (complex and expensive) system which acts as a whole to perform an HPC single simulation
  - Write software that can take advantage of the supercomputer performance, e.g. employing parallel execution
- All system hardware/software aspects are important:
  - Select best microprocessor and accelerator
  - Surround it with a bandwidth-rich environment: local memory, interconnection network
  - Scale the system
    - Provide scalable programming and performance tools
    - Provide scalable I/O
    - Eliminate operating system interference (OS jitter)
    - Design in reliability and resiliency
    - Provide scalable system management



“Anyone can build a fast CPU.  
The trick is to build a fast system.”  
— Seymour Cray

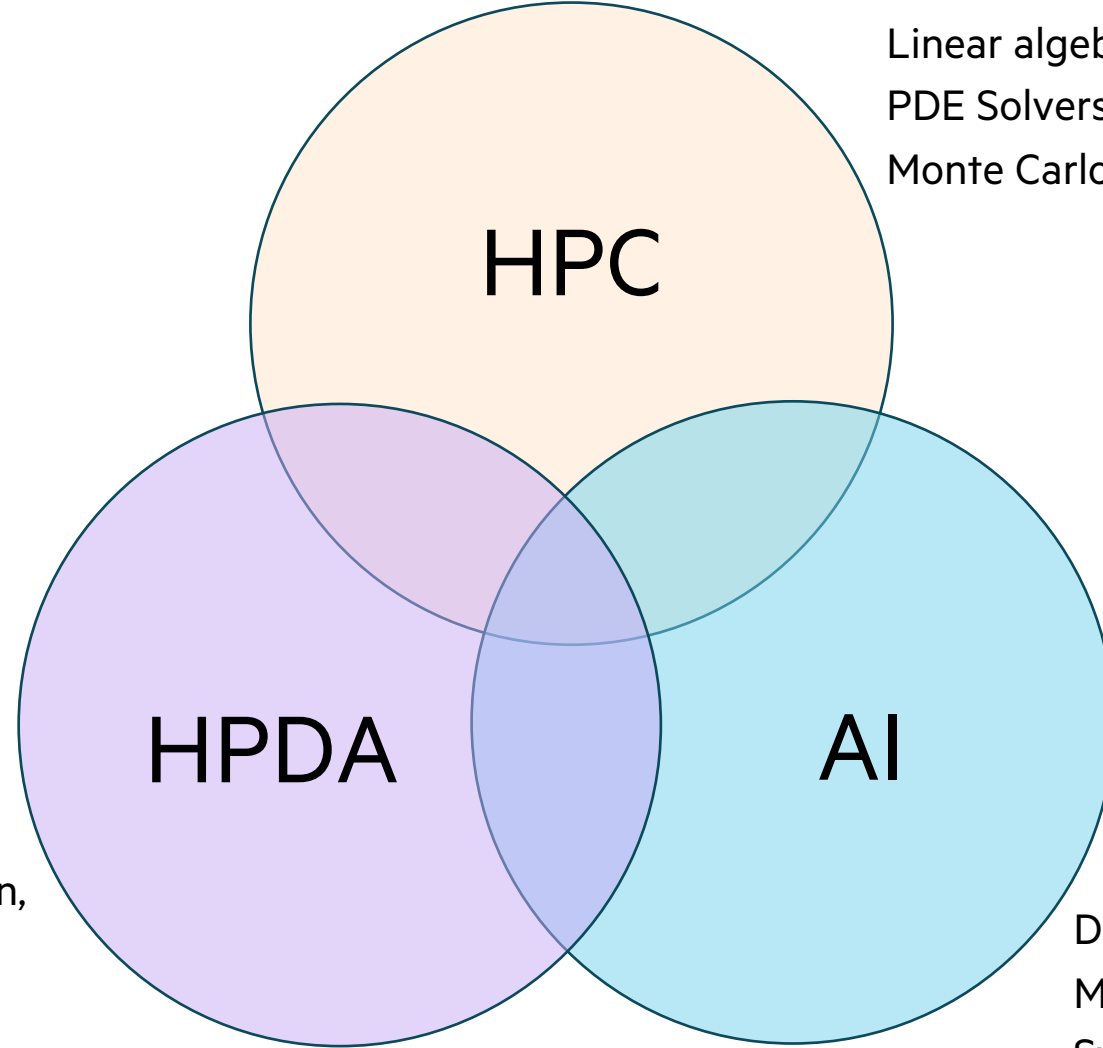




# Focus on Intensive Numerical Computations

- Traditionally, HPC has been embraced by research scientists who needed to perform complex mathematical calculations, which requires **intensive numerical computations (Floating-Point operations, Flop)**
  - Double-precision (DP) operations
- Nowadays HPC is gaining the attention of a wider number of enterprises spanning an array of fields
  - Not discussed in the rest of the presentation (lack of time)

Monitoring, classification,  
anomaly detection,  
hypothesis testing,  
clustering, optimization,  
signal processing, etc.

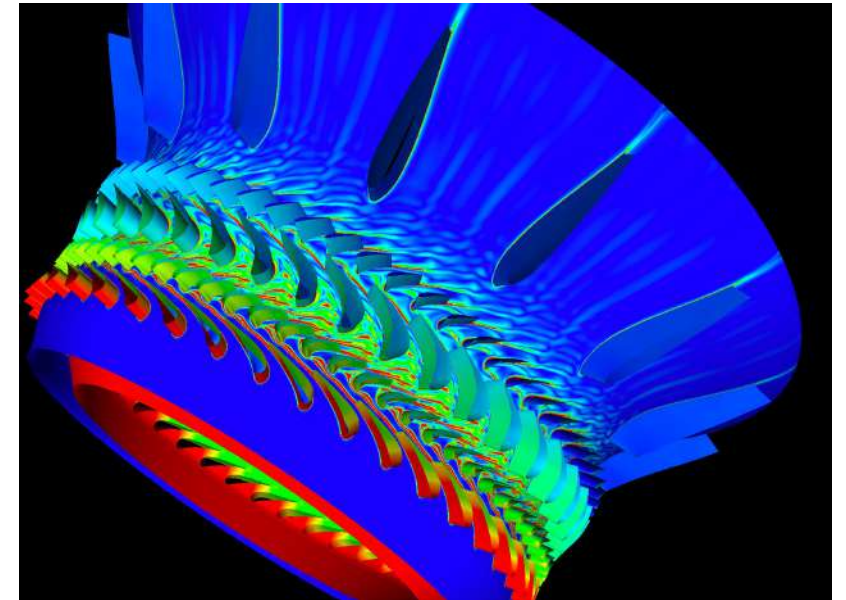
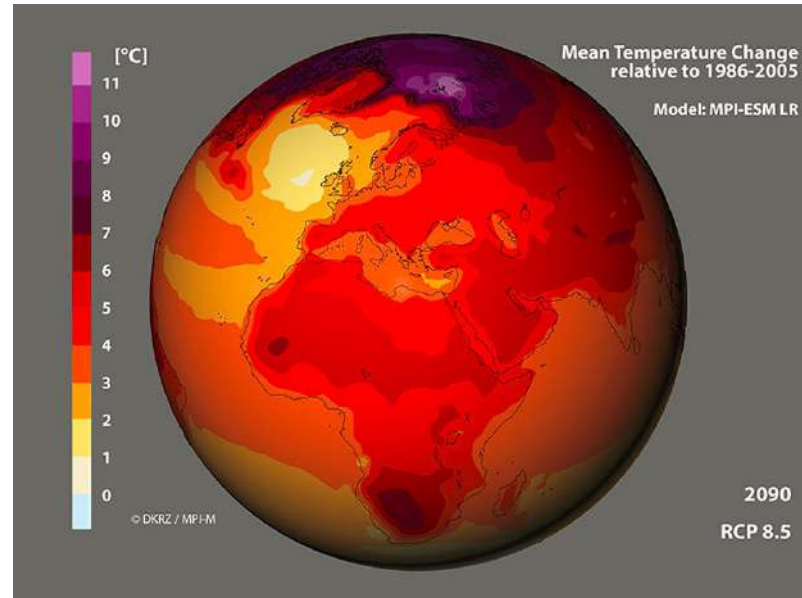
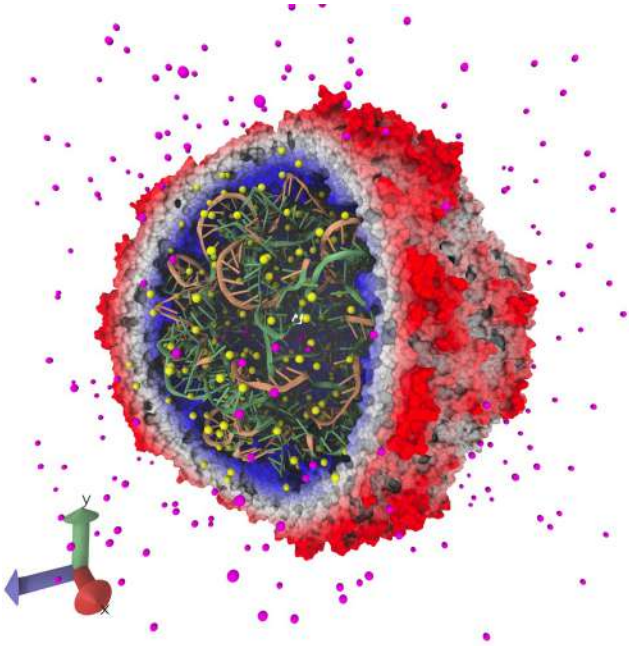


CFD Simulations  
Linear algebra  
PDE Solvers,  
Monte Carlo, etc.

Deep learning,  
Machine learning,  
System modelling, etc.

# Why we need (more and more) Supercomputing

- Simulation is the third pillar of the science research (with experiment and theory)
  - Quantum mechanics, weather forecasting, climate research, oil and gas exploration, cosmology, biology, fluid dynamics, ...
  - More and more complex systems to study



- Recent convergence of Artificial Intelligence and Supercomputing

# Supercomputers

---

*#3 pencils and quadrille pads.*

- Seymour Cray (when asked what CAD tools he used to design the Cray-1 supercomputer), 1980



# The Beginning

- The first supercomputer (1964): CDC6600
  - Realized by Seymour Cray (1925-1996), *the father of supercomputing*
  - Single CPU@10MHz, 1-3 MFlop/s
  - 8 M\$ cost (~60M\$ in today's money)
- In 1972 Cray started his own company and released the first supercomputer in 1976: Cray-1
  - Fastest system for 5 years, 10x faster than competing systems
  - Single processor @80MHz, 160 MFlop/s
  - 5-8 M\$ (~25M\$ in today's money)



# The Cray Era: Vector Processors

A single instruction can operate on one-dimensional arrays of data called *vectors*, at the same time

1982

- **Cray X-MP**

- First multi-processors supercomputer, 2 vector processors @105MHz, up to 400 MFlop/s
- 16 MB memory

1988

- **Cray Y-MP**

- 2, 4, or 8 vector processors @167MHz, 333 MFlop/s each (up to 2.6GFlop/s)
- Up to 512 MB memory

1991

- **Cray C90**

- 3x faster than Y-MP vector processor, 2x more processors, i.e. up to 15.6GFlop/s



## Massively Parallel Era (>1990)

---

- Vector processing did not allow to scaling to hundred processors systems
  - Custom processors were not feasible
- Shift towards **commodity-based processors, i.e. PC technology**
  - *The attack of Killer Micros*, Eugene Brook's talk at Supercomputing90
  - Mainly driven by cost – computing went mainstream
- Thousands of “small” (and relatively “cheap”) **compute units (nodes)** connected together
  - *“If you were plowing a field, which would you rather use: two strong oxen or 1,024 chickens?”*, Seymour Cray's joke
  - **Distributed memory**: each node has its own memory attached
  - Use special **networks** to connects nodes

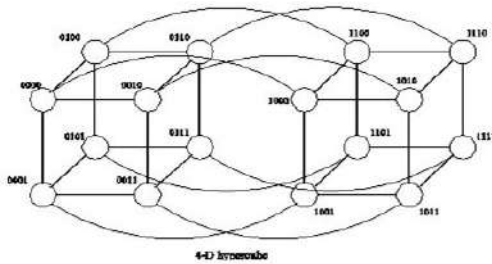




# Network Complexity

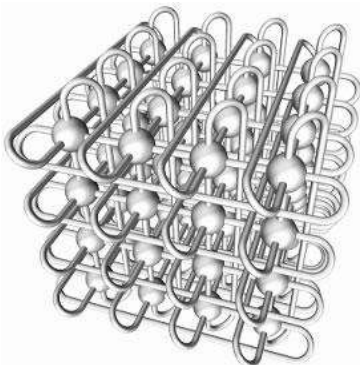
- nD Hypercube

- Number of outgoing ports scales with the log of the machine size
- Difficult to scale out



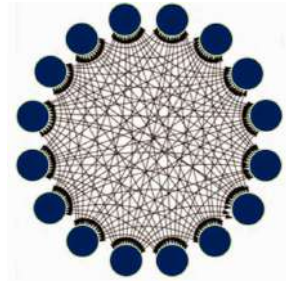
- nD Torus

- Nearest neighbor



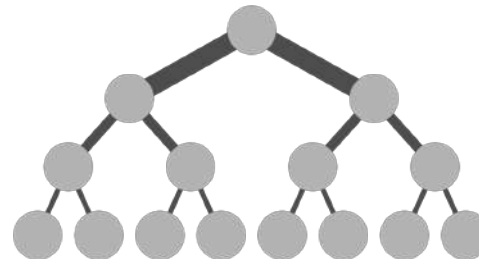
- Dragonfly

- Hierarchical design
- All-to-all connectivity between groups



- Fat Tree

- Increases bandwidth for the higher levels of the tree

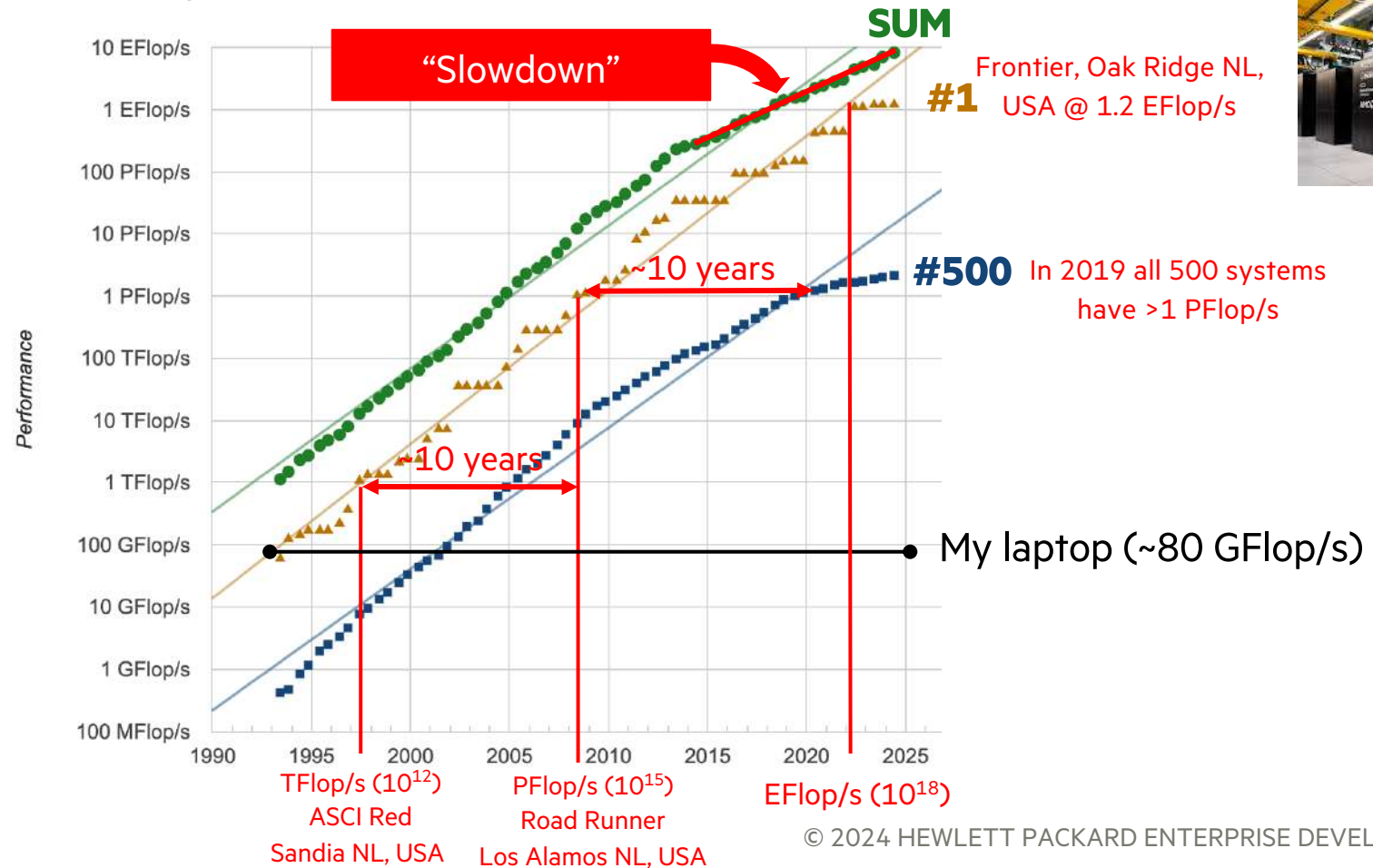


## Challenging task

- Connect thousands of nodes
- Topology mapping of the application
- Hardware instability

# Performance Trend

- Top500.org: list of the 500 fastest supercomputers (update twice a year)
  - Ranking based on the (historical) HPL benchmark (DP Flop/s)
    - Numerical linear algebra solver (compute intensive)



# Top-10 list

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	<b>Frontier</b> - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,206.00	1,714.81	22,786
2	<b>Aurora</b> - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
3	<b>Eagle</b> - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
4	<b>Supercomputer Fugaku</b> - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
5	<b>LUMI</b> - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107

1<sup>st</sup> System in Europe (since 2022),  
Same HPE Cray system as Frontier

- Rmax: HPL benchmark performance
- Rpeak: theoretical aggregate max performance
- **Rmax/Rpeak: performance efficiency (%)**
- **Rmax/Power: energy efficiency (GFlop/s/W)**

6	<b>Alps</b> - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	1,305,600	270.00	353.75	5,194
7	<b>Leonardo</b> - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, EVIDEN EuroHPC/CINECA Italy	1,824,768	241.20	306.31	7,494
8	<b>MareNostrum 5 ACC</b> - BullSequana XH3000, Xeon Platinum 8460Y+ 32C 2.3GHz, NVIDIA H100 64GB, Infiniband NDR, EVIDEN EuroHPC/BSC Spain	663,040	175.30	249.44	4,159
9	<b>Summit</b> - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096
10	<b>Eos NVIDIA DGX SuperPOD</b> - NVIDIA DGX H100, Xeon Platinum 8480C 56C 3.8GHz, NVIDIA H100, Infiniband NDR400, Nvidia NVIDIA Corporation United States	485,888	121.40	188.65	

1<sup>st</sup> System in Italy (since 2022)



# Energy Concern

- **Top500.org/green500:** list of the 500 most energy efficient supercomputers
  - Still based on HPL benchmark, now focusing on energy efficiency
- Frontier test system at the 7<sup>th</sup> position
- Clear pattern appears: all efficient systems have Graphics Processor Units (GPU) co-processors!

Rank	TOP500 Rank	System	Cores	Rmax (PFlop/s)	Power (kW)	Energy Efficiency (GFlops/watts)
1	189	JEDI - BullSequana XH3000, Grace Hopper Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, ParTec/EVIDEN EuroHPC/FZJ Germany	19,584	4.50	67	72.733
2	128	Isambard-AI phase 1 - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE University of Bristol United Kingdom	34,272	7.42	117	68.835
3	55	Helios GPU - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Cyfronet Poland	89,760	19.14	317	66.948
4	329	Henri - ThinkSystem SR670 V2, Intel Xeon Platinum 8362 32C 2.8GHz, NVIDIA H100 80GB PCIe, Infiniband HDR, Lenovo Flatiron Institute United States	8,288	2.88	44	65.396
5	71	preAlps - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	81,600	15.47	240	64.381
6	299	HoreKa-Teal - ThinkSystem SD665-N V3, AMD EPYC 9354 32C 3.25GHz, Nvidia H100 94Gb SXM5, Infiniband NDR200, Lenovo Karlsruher Institut für Technologie (KIT) Germany	13,616	3.12	50	62.964
7	54	Frontier TDS - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	120,832	19.20	309	62.684

# The Exascale Era of Computing

---

*In the future, computers may weigh no more than 1.5 tons.*

– Popular mechanics, 1949



# Exascale Estimates

- After the first Petascale system came online (2009), people started to discuss about an Exascale system
  - The International Exascale Software Project (IESP)
- The IESP Roadmap was published in 2011
  - Dongarra J., Beckman P., et al. *The International Exascale Software Roadmap. International Journal of High Performance Computer Applications*, 25(1), 2011

	2009	Pre-exascale	Exascale estimates
System peak (PFlop/s)	2	100-200	1000
Power (MW)	6	~ 15	~ 20
Node concurrency	12	O(100)	O(1k-10k)
System size (nodes)	18,700	>50,000	O(100k-1M)
Mean time to interrupt (days)	>1	O(1)	O(0.1)

DOE  
Constraint

Source: <https://www.hpcwire.com/2021/07/14/frontier-to-meet-20mw-exascale-power-target-set-by-darpa-in-2008/>

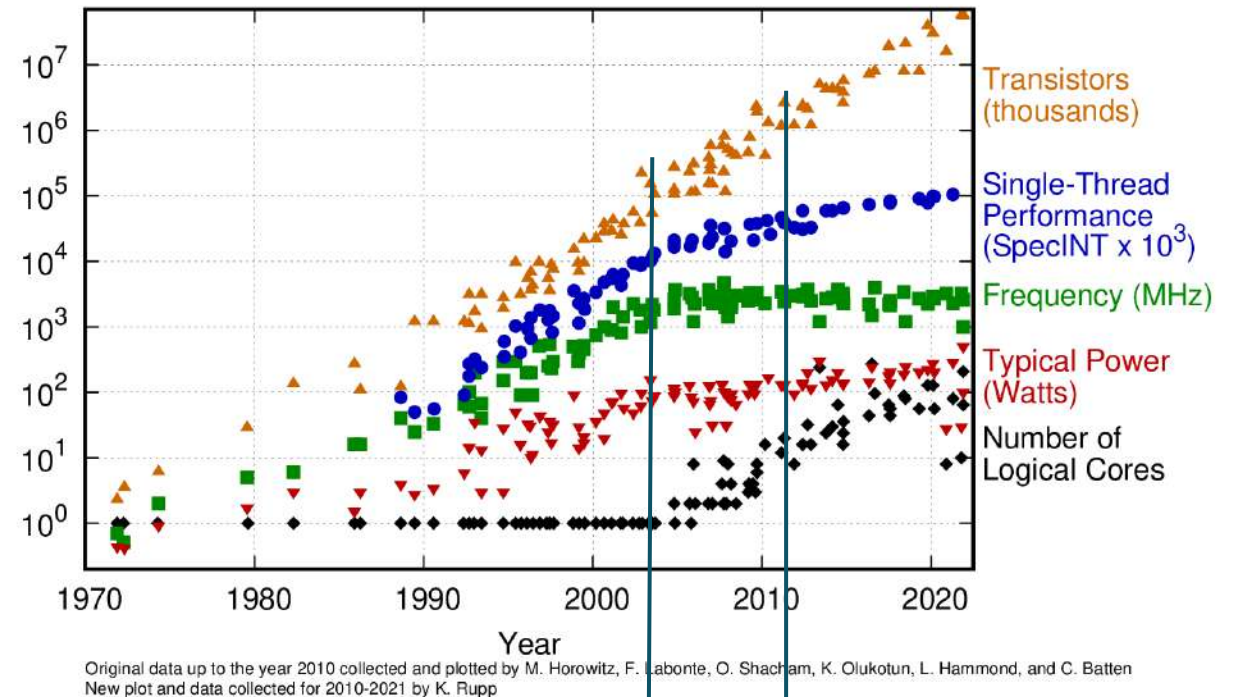




# Technology Trend

- Moore's law has been driven the performance improvement in the last 50 years
  - 3nm in current production (well, this dimension is meaningless...)
  - Current model up to ~1nm (?), probably another 4 years before reaching this limit
- Power density wall: hard to increase performance per core without large energy investment (Dennard scaling)
  - Multi-core CPUs (>2004)
  - Many-core accelerators, eg. GPUs (>2012)

50 Years of Microprocessor Trend Data

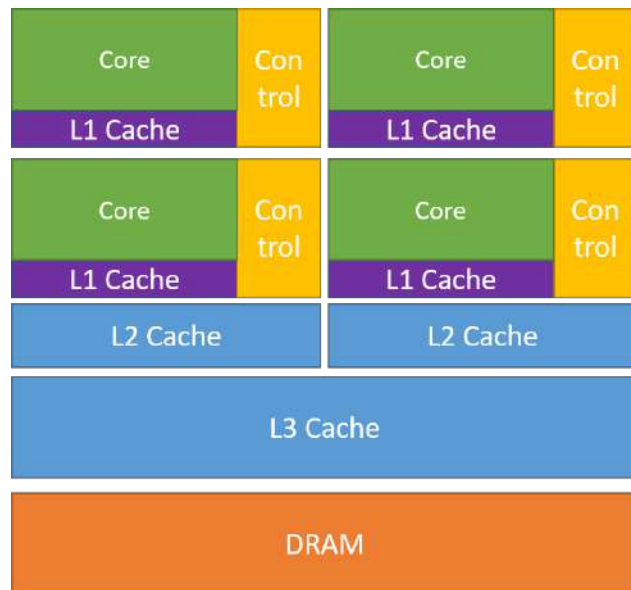


multi-core CPUs

many-core accelerators

# Accelerators

- Specialized parallel hardware for floating point operations
  - Based on massive parallel architectures
  - Co-processors for traditional CPUs (which are used for generic workloads)
  - GPUs have been the most common accelerators during the last few years



CPU



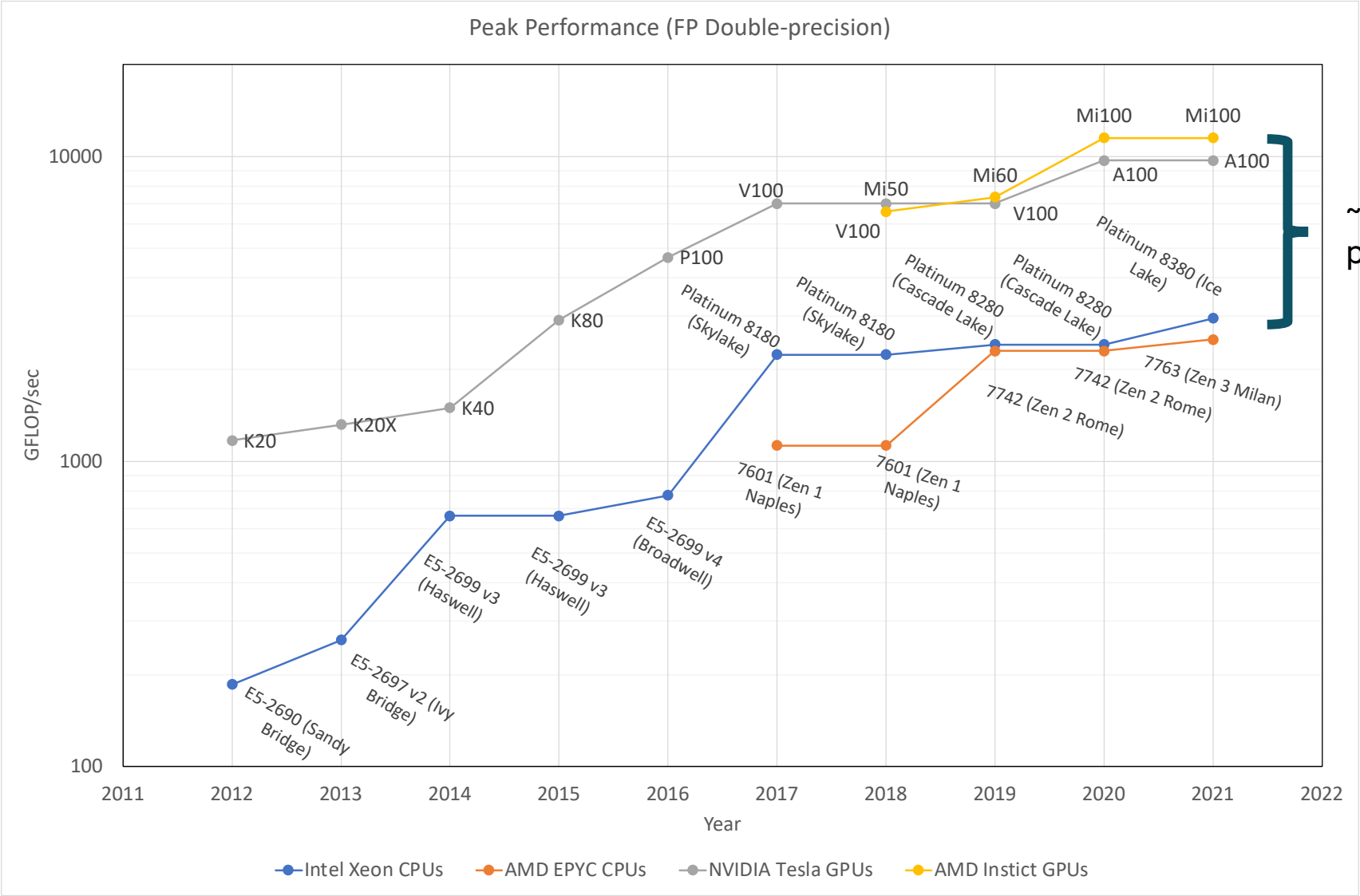
GPU

Source: CUDA C++  
Programming Guide

- Low compute density
- Large caches
- Optimized for serial operations
- Latency optimized

- High compute density
- High computations per Memory Access
- Built for parallel operations
- Throughput optimized

# CPU VS GPU Performance Trend

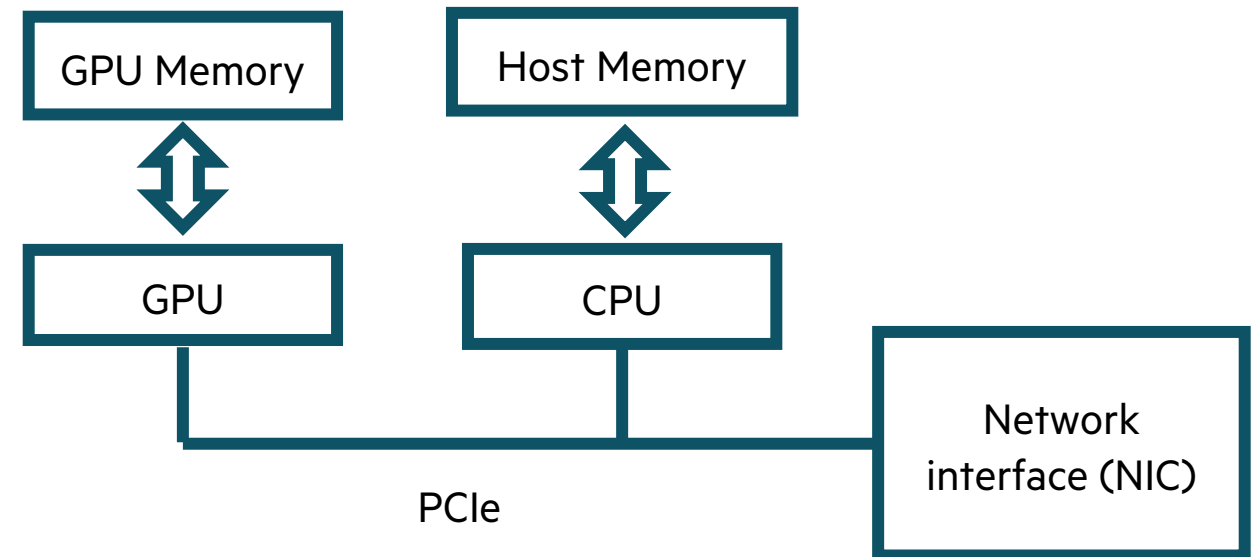


~4x peak-performance ratio



## Typical node layout: Single GPU

- GPU connected to CPU via PCIe
  - GPU and Host memory do not share the same address space
  - Must copy data from/to the host memory to/from the GPU (device) memory



# AMD Mi250 GPU: Chiplet Multichip Architecture

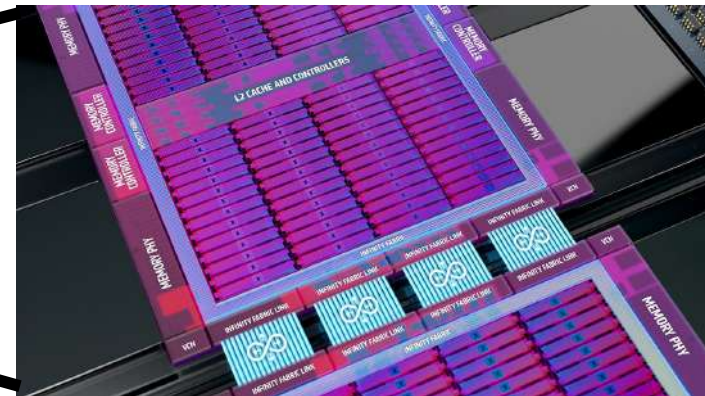
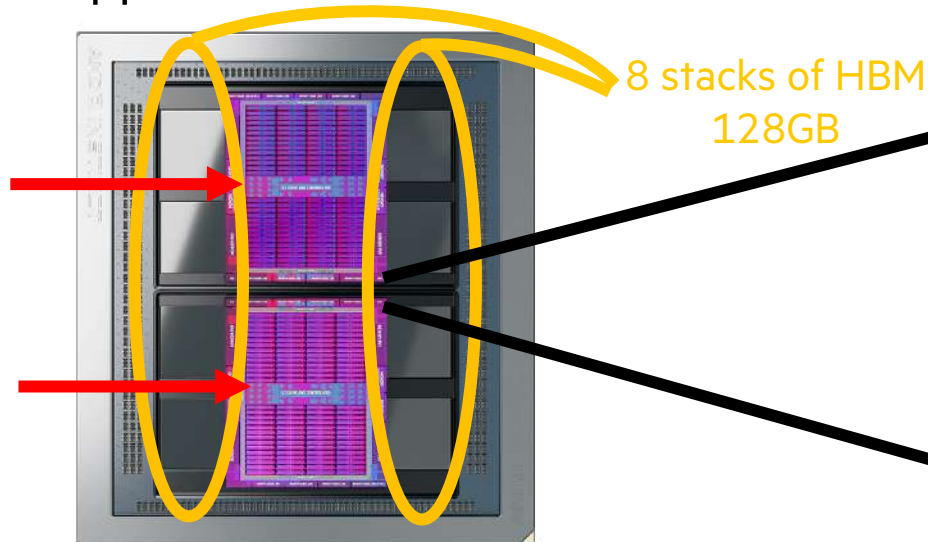
- **Extend chips complexity**

- Monolithic chips: all components into a single printed piece of silicon
- Chiplets break things up into smaller components that are then combined into a larger processor via an interposed in a single package
  - Mix-and-match “LEGO-like” assembly

➔ **Reduce costs**

- AMD using this approach on the Zen CPUs and **Mi250 GPUs**

Based on 2 Mi100  
(GCD, Graphics  
Compute Die)

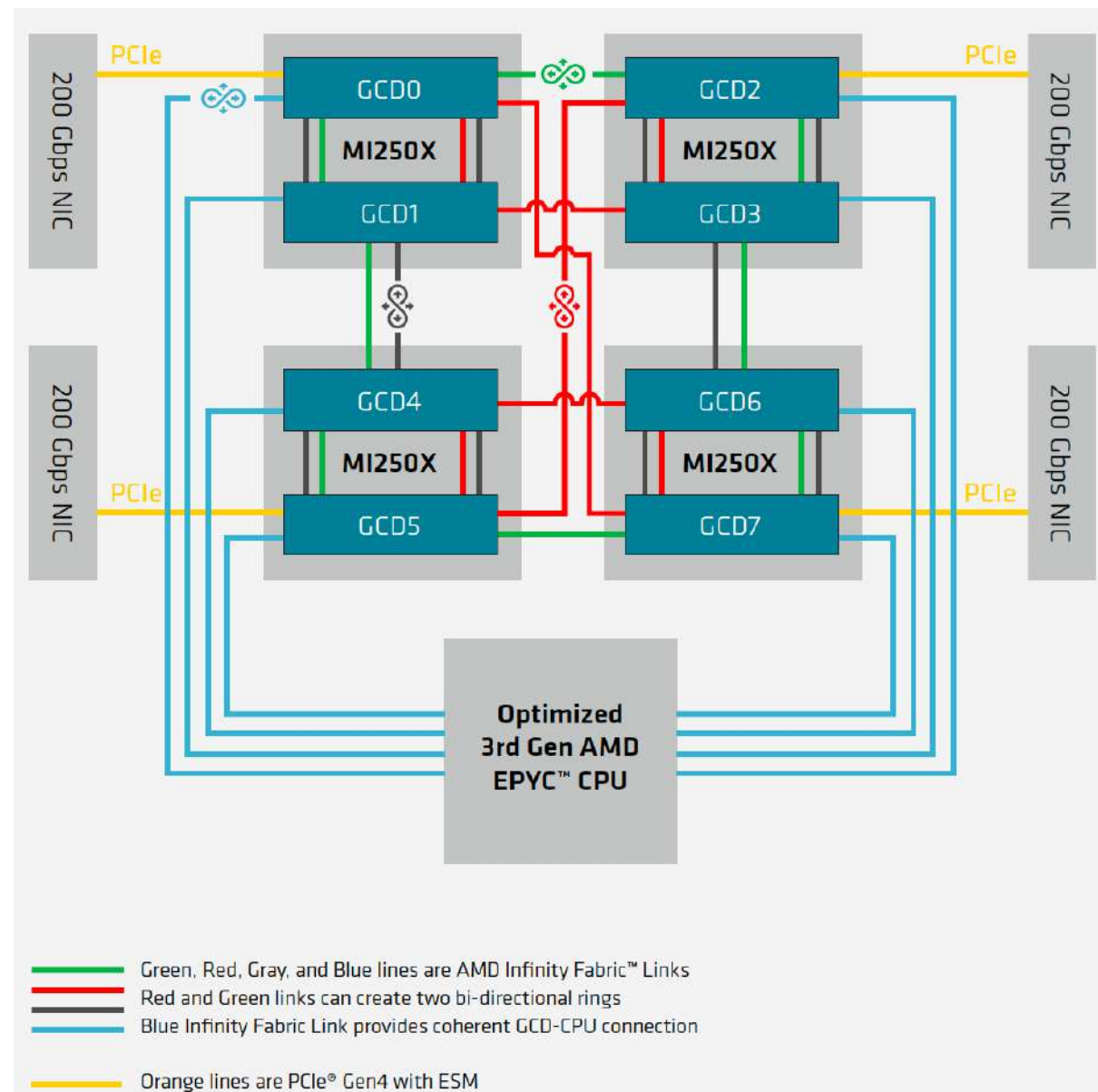


4 links for fast  
GCDs connection  
(up to 400GB/s of  
bidirectional  
bandwidth)

Source: <https://www.amd.com/en/technologies/cdna2>

# HPE-AMD Multi-GPUs Node Layout

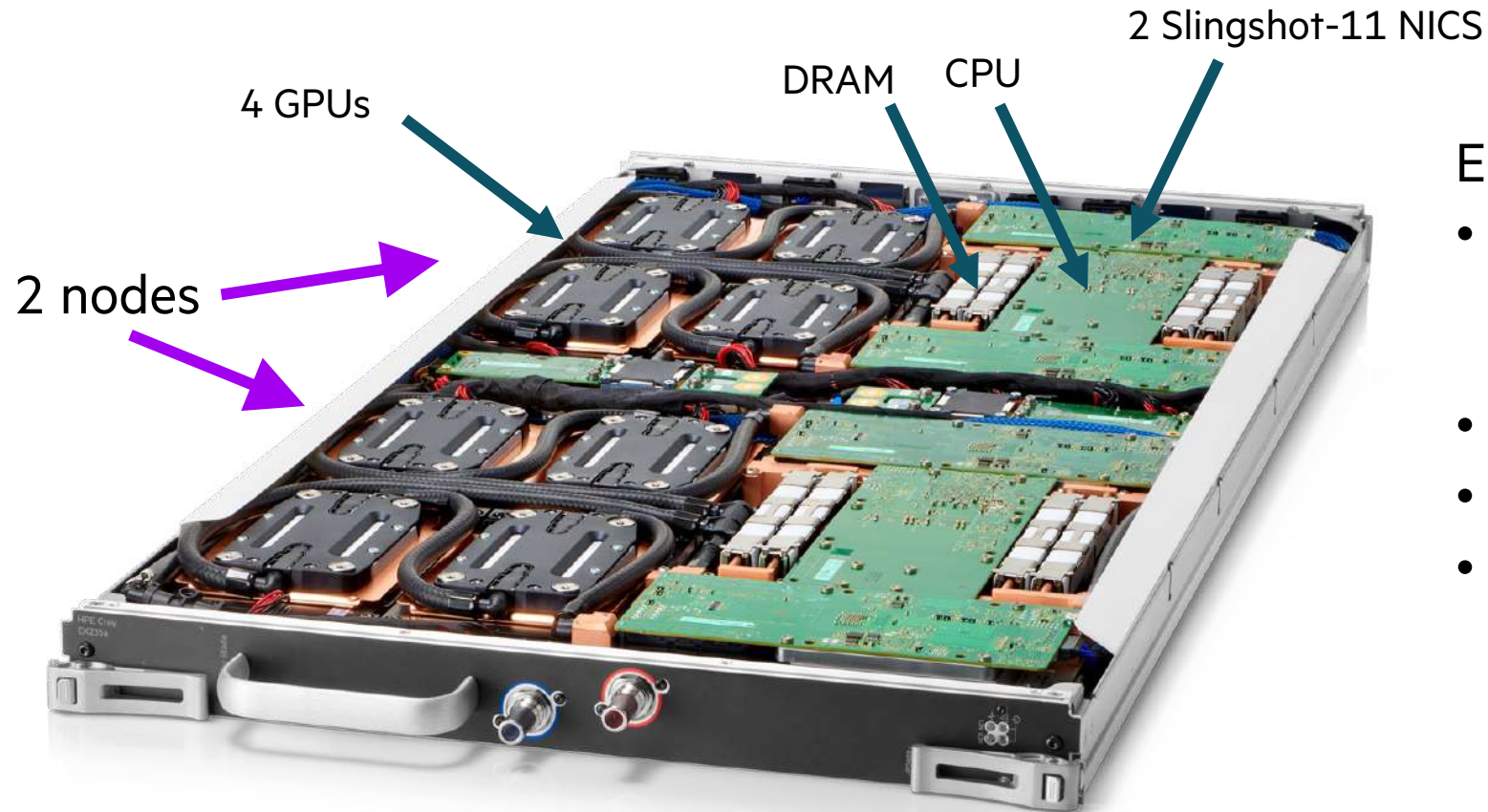
- Fast connections between GPUs, CPU-GPUs, GPU-NICs
- AMD node layout: 4 Mi250 GPUs
- Mi250 specs:
  - 110 Compute Units (CU) per each die = 220 CUs
  - 64 SIMD threads per each CU = 14080 Stream Processors
  - Peak FP64/FP32 Vector: 47.9 TFLOPS
  - Peak FP64/FP32 Matrix: 95.7 TFLOPS
  - Total L2 cache: 8 MB per each die (64kB per CU)
  - Frequency: up to 1700 MHz
  - Max power: up to 560 Watts



Source: <https://www.amd.com/system/files/documents/amd-cdna2-white-paper.pdf>



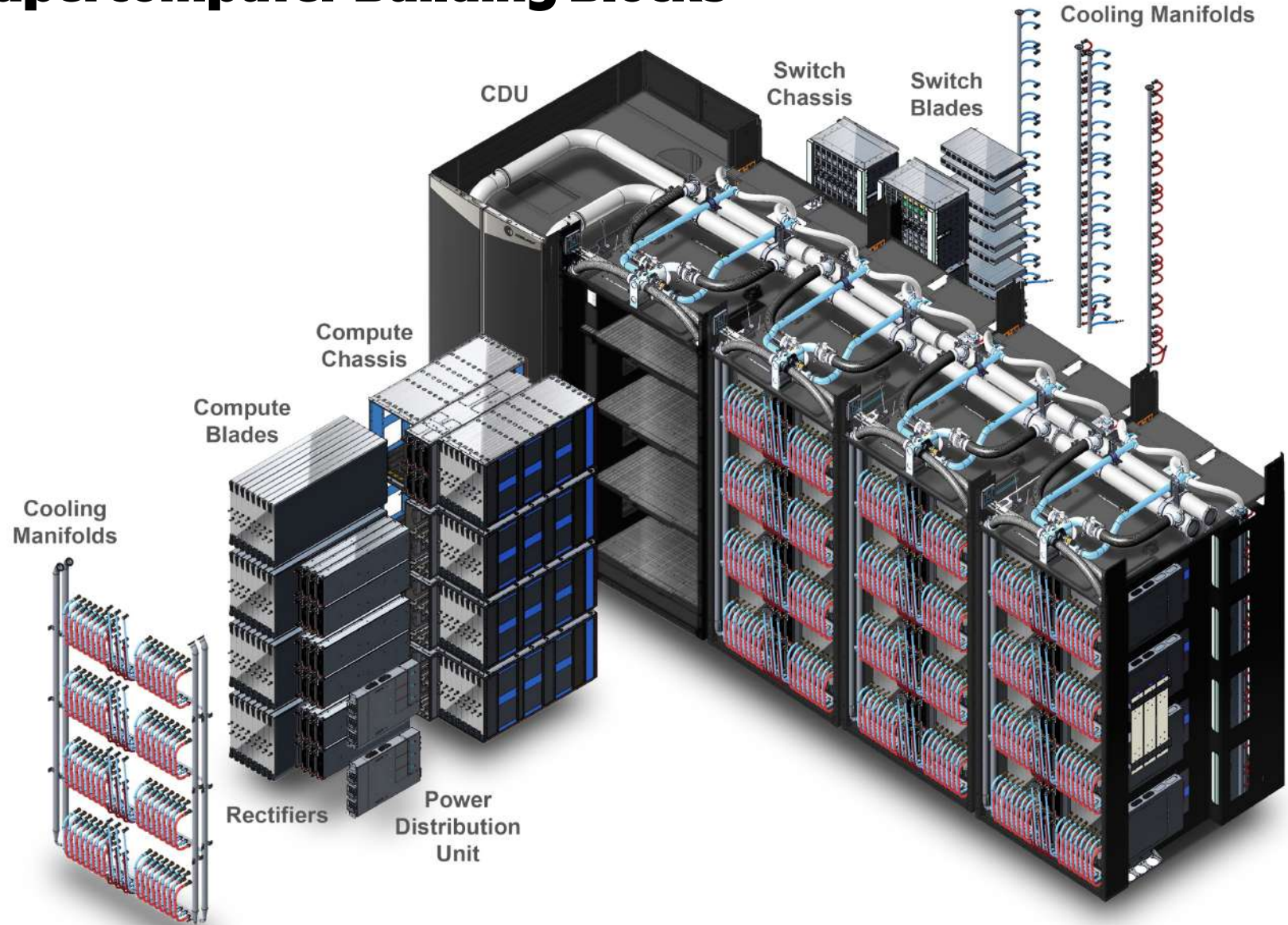
# Frontier Compute Blade Architecture



Each node:

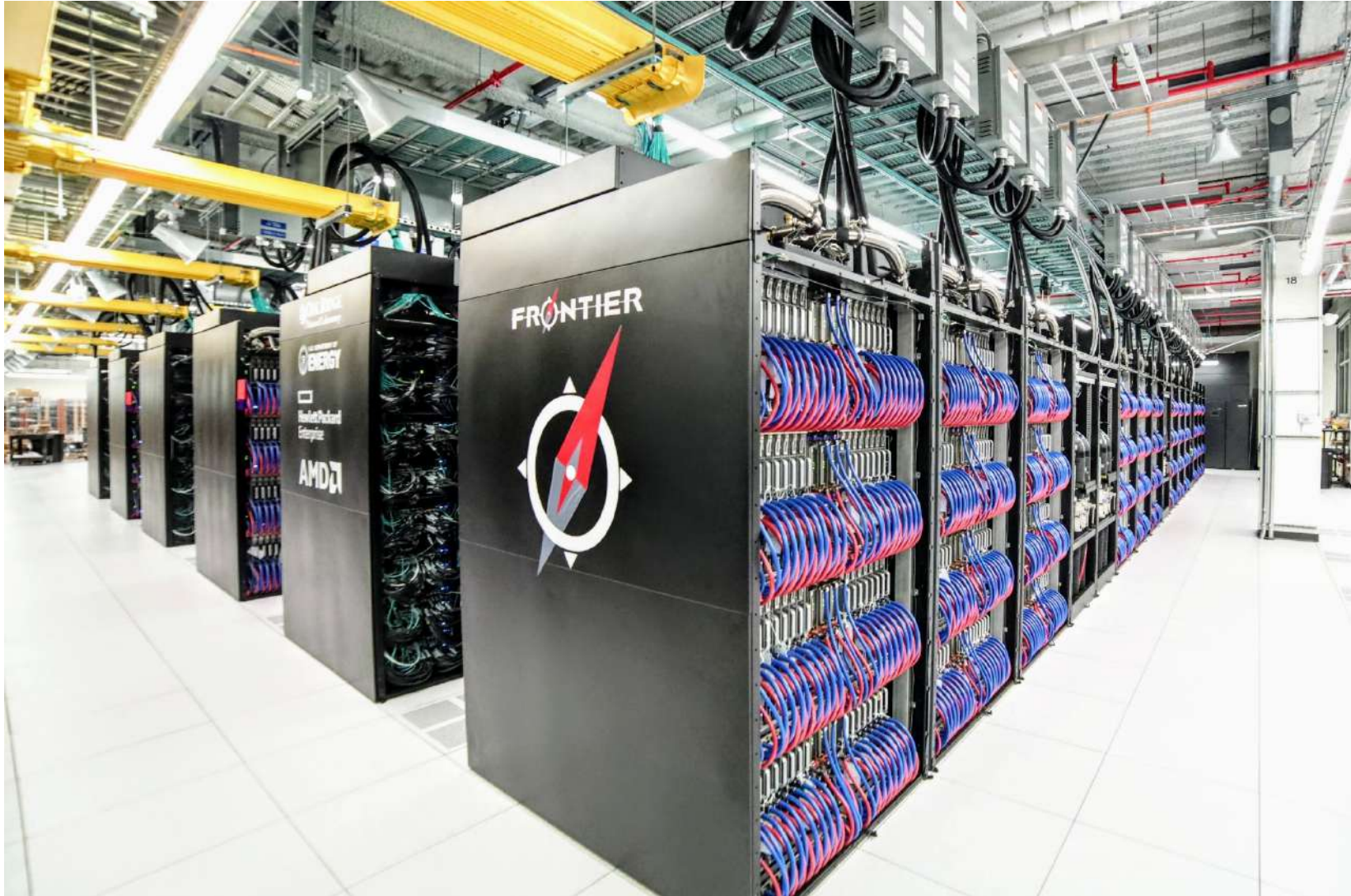
- AMD EPYC 7A53 “Optimized 3rd Gen EPYC” 64-Core Processor, 2.00 GHz
- 512 GB DDR4 memory
- 4x AMD MI-250X GPU
- Each GPU connected to a Slingshot 200Gb/s NIC (Dragonfly topology)

# Frontier Supercomputer Building Blocks





# Frontier in Numbers



- #1 in the world @ 1.2 EFlop/s
- 74 HPE Cray EX cabinets
- 9,408 AMD EPYC CPUs, 37,632 AMD GPUs, 37,632 Slingshot NICs
- 700 petabytes of storage capacity
- 90 miles of HPE Slingshot networking cables

# Exascale Estimates VS Frontier Numbers

	2009	Pre-exascale	Exascale estimates	Frontier
System peak (PFlop/s)	2	100-200	1000	1206
Power (MW)	6	~ 15	~ 20	22.786 (<20 MW per exaflop)
Node concurrency	12	O(100)	O(1k-10k)	O(100k)
System size (nodes)	18,700	>50,000	O(100k-1M)	9,408
Mean time to interrupt (days)	>1	O(1)	O(0.1)	O(0.3)

Source: <https://www.hpcwire.com/2021/07/14/frontier-to-meet-20mw-exascale-power-target-set-by-darpa-in-2008/>

Quoting Al Geist (ORNL): “In the end, what we found out was that exascale didn’t require this exotic technology that came out in the 2008 report. We didn’t need special architectures, we didn’t even need new programming paradigms. It turned out to be very incremental steps, not a giant leap like we thought it was going to take to get to Frontier.”



# Programming Models

---

*If you build it, they will come.*

– Quote from the Field of Dreams movie (1989)



# We got the Fantastic Powerful Hardware...

## What could go wrong?



© Keystone

### Baby Is Given Fresh Air in Window Cage

**B**ABIES of flat and tenement dwellers, whose tiny lungs have been forced to breathe stale and overheated air, are offered relief by the recent English invention pictured above. A large metal crate is attached to the outside of the window by two stout iron poles. A baby basket can easily be lowered inside. If insects are likely to trouble the child, the crate can be screened.



# Software Complexity: A Long Story

---

## Hardware → Software

- Reach peak performance
- Several level of parallelism
  - Intra-cores, intra-nodes, inter-nodes
  - Accelerators
- Memory access
  - Various level of memories with different speeds and sizes
  - Flop/s are for free, memory speed can be the ultimate limit
- Mixed-precision computations
- Erratic hardware

## Software → Hardware

- Auto-tuning
  - Rewriting code for each system generation is not an option!
- Parallel programming paradigms
  - E.g. Message Passing, Threading, Tasking
- New algorithms
- Legacy code
  - Yes, the old good Fortran
- New languages and programming models
- Static versus dynamic parallelism
  - Runtime scheduling

**This is just the beginning of the story... very challenging!**

# Parallel Programming Models: Short Overview

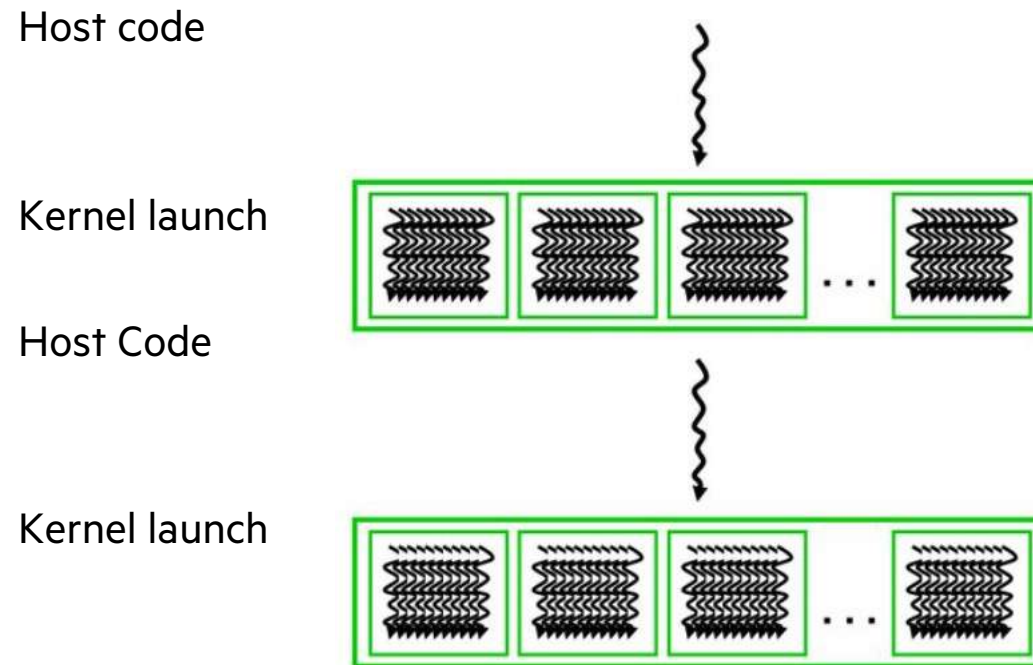
---

- Multi-processes
  - Suitable for distributed-memory systems (e.g. on multi-nodes)
  - Message Passing Model: MPI
  - Partitioned Global Address Space (PGAS): SHMEM, Coarrays, UPC, Chapel, GASPI
- Multi-threaded
  - Suitable for shared-memory systems (e.g. on multi-cores)
  - Direct parallelism: Pthreads, std::threads
  - Directive based: OpenMP
  - Programming languages: C++ Parallel STL
  - Via parallel libraries: HPX, TBB, C++ Parallel STL Executors
- Accelerator execution (via GPUs)
  - Code regions are **offloaded** from a host CPU to be computed on an accelerator
  - Portability across AMD, NVIDIA, Intel GPUs
- Hybrid model: mixing all the above



# GPU Offload execution

- Code regions are **offloaded** from a host CPU to be computed on an accelerator
  - Kernel execution
- Keep latency sensitive work on the CPU (host)
- Memory management via copies (host-to-device, device-to-host, device-to-device)



# Programming considerations

---

- GPU model requires many small tasks executing a kernel
  - Can replace iterations of a CPU loop with a GPU kernel call
    - Would it be better to port 10 iterations with 1M instructions each or 1M iterations with 10 instructions each?
- Need to adapt CPU code to run on the GPU
  - Programming patterns, eg. reduction
  - Decompose your problem in hierarchical thread/grid model
  - **Manage data transfers between CPU and GPU memories**
    - Data movement can easily become the bottleneck for performance
    - Try to reduce data movement, eg. reusing data
    - Hide latency by overlapping data movement with communications
  - **Keep reusing data on the GPU to reach high an efficient use of the hardware (occupancy)**
- Usually, after a first porting of the code to GPU, performance tunings is required to reach good performance



# Accelerate applications



## Accelerated Libraries

- The easiest solution, just link the library to your application without in-depth knowledge of GPU programming
- Many libraries are optimized by GPU vendors, eg. algebra libraries

## Directive based methods

- Add acceleration to your existing code (C, C++, Fortran)
- Can reach good performance with somehow minimal code changes
- OpenACC, OpenMP

## Programming Languages

- Maximum flexibility, require in-depth knowledge of GPU programming and code rewriting (especially for Fortran)
- Kokkos, RAJA, CUDA, HIP, OpenCL, SYCL



# Conclusion

---

*I don't know what the language of the year 2000 will look like,  
but I know it will be called Fortran.*

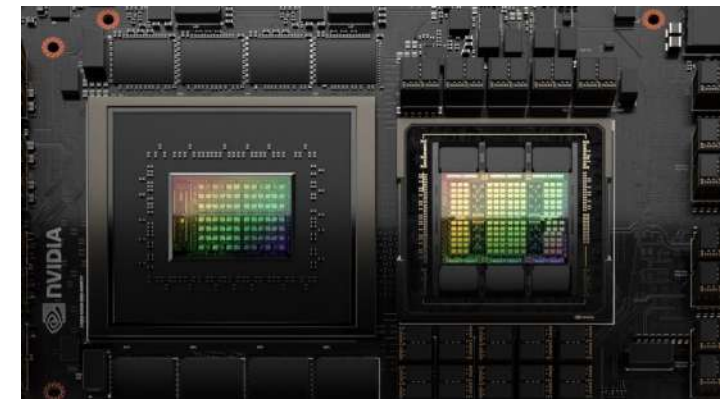
– CA Hoare, 1982



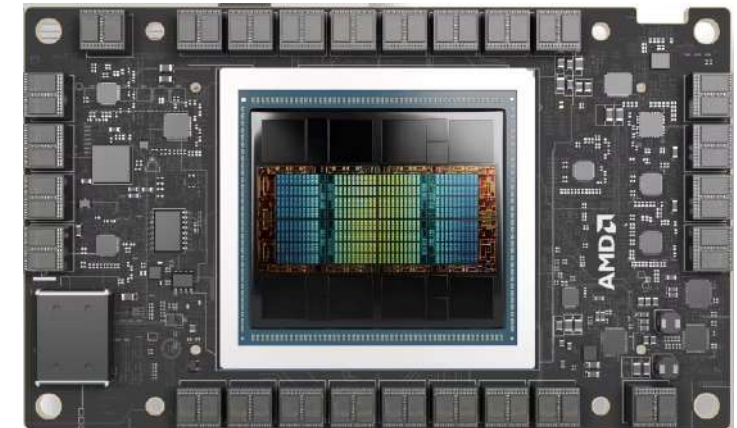


# The Take-home Message

- Accelerators and more accelerators
  - GPUs and more GPUs
  - NVIDIA Grace-Hopper: Jupiter, the 1<sup>st</sup> European Exascale system (end of 2024)
  - AMD Mi300: El-Capitan (USA), target 2 EFlop/s (end of 2024)
- Need to adapt algorithms/code to run on the GPUs
  - Offload model
  - Optimize code for data-movement and occupancy
- Programming models: choose the model that best fits your requirements
  - Do not forget portability in the long run



NVIDIA Grace-Hopper Superchip (source: <https://www.nvidia.com/en-us/data-center/grace-hopper-superchip/>)



AMD Mi300X (source: <https://www.amd.com/en/products/accelerators/instinct/mi300/mi300x.html>)

# One Slide Summary

---

## HPC is there to stay for a long time

- Exascale mission accomplished, Zettascale next...
- Pushing technology to the limit, hardware and software

## Exascale and beyond will enhance computing systems at all scales

- Affordable petascale systems?
- Golden age of HPC software developers, very exciting

## Traditionally HPC can learn a lot from emerging applications (and vice versa)

- *In primis* machine learning and data science

## Major cloud vendors provide HPC infrastructures to experiment with

- HPC on Demand, assuming you don't want to buy a supercomputer or apply for accessing to a supercomputer center...



## Secondments @ HPE

---

- Reconfigurable hardware, namely FPGA, as an accelerator
  - Adapt the hardware to your software
- GPU application porting
- Quantum Computing applications
- AI applications
- Data-driven workflow optimization
- Data movement cost models / benchmarking
- Presenting and accessing memory efficiently
- Memory abstraction frameworks
- Tiling abstractions
- High-level workflow managers
- C/C++/Fortran-related compiler tools
- Performance-portable application containerization

**Contact me if you are interested!**



# Questions?

---

alfio.lazzaro@hpe.com



# (Some) GPU Programming Models Portability





# CUDA / ROCM

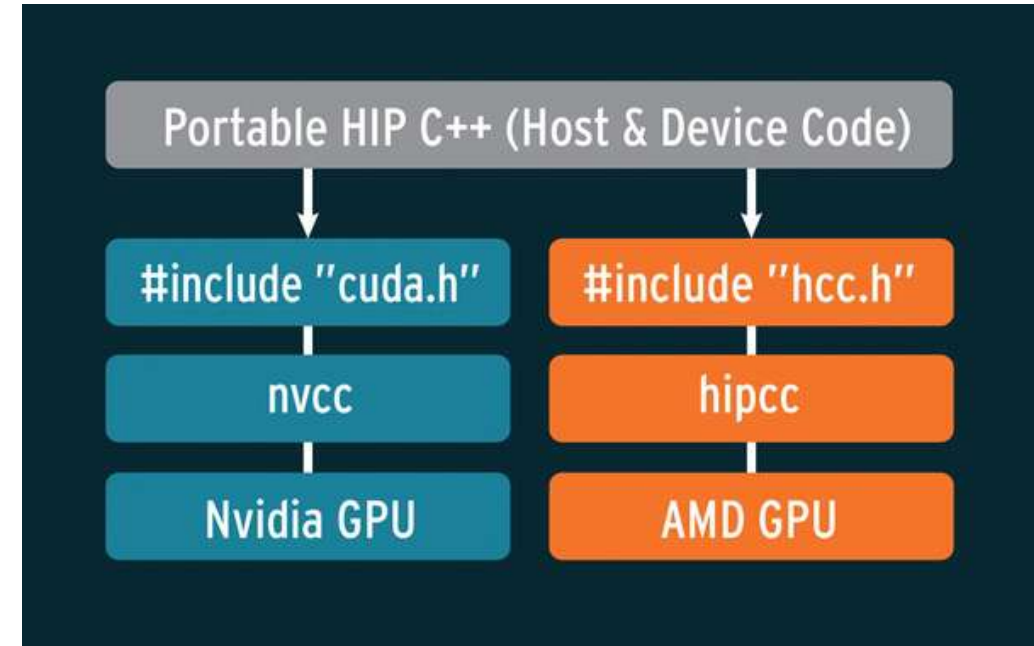
---

- CUDA is targeting NVIDIA GPUs only
  - <https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>
  - Hardware co-design, full control of the hardware features
  - Only a main compiler, proprietary
    - C/C++ extension
  - Quite long history (>10 years)
  - Well supported, rich set of tools and libraries
  - Large community adoption and experience
- ROCM is targeting AMD GPUs
  - Analogous to CUDA-NVIDIA, but
    - Open-source at <https://github.com/RadeonOpenCompute>
    - LLVM/Clang based compiler, full compatibility with C/C++
    - Increasing tools and libraries availability



# HIP (Heterogeneous Interface for Portability)

- C++ runtime API and kernel language
- Supported by AMD
  - <https://github.com/ROCm-Developer-Tools/HIP>
- Provide access to NVIDIA and AMD GPUs



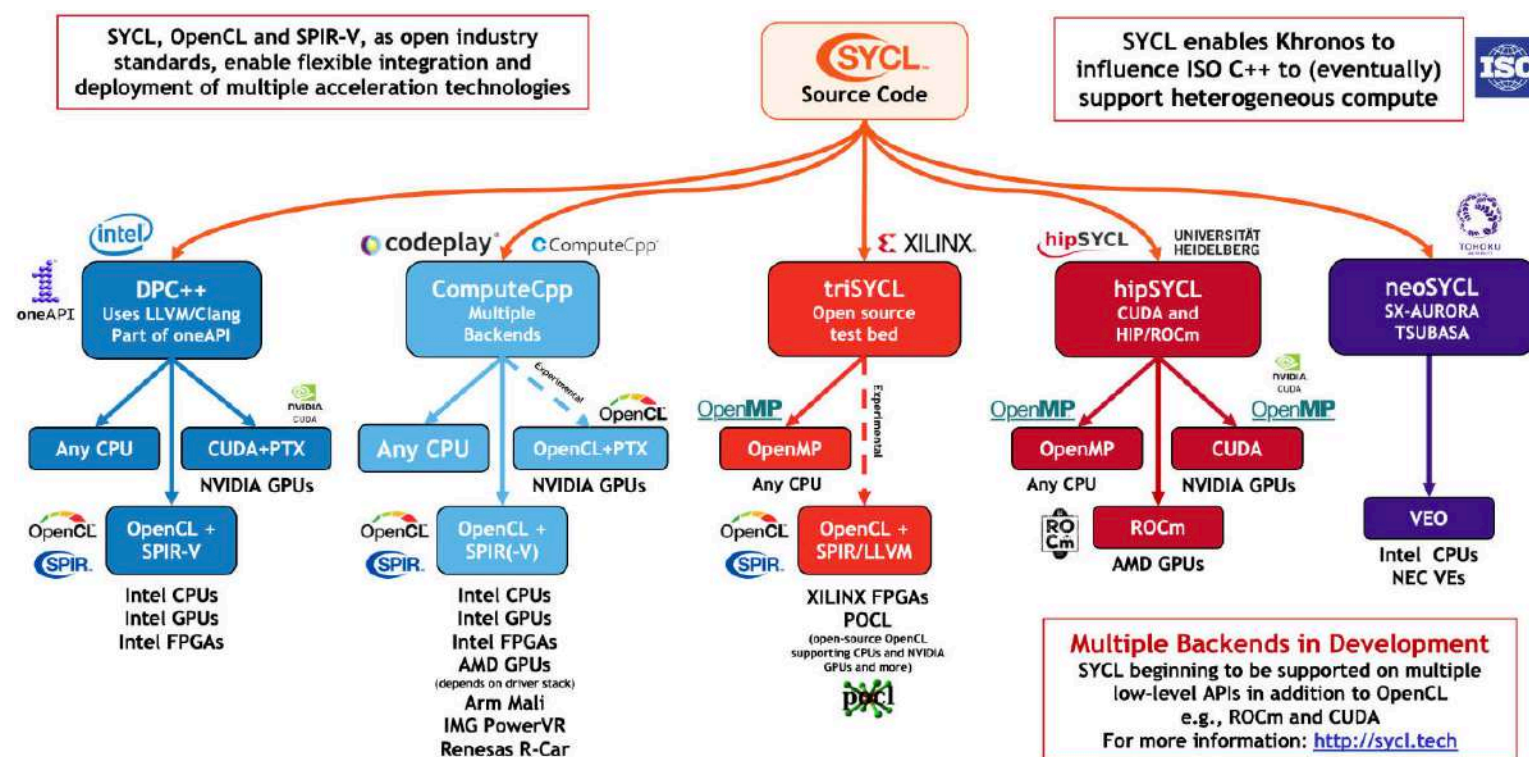
Source: <https://www.admin-magazine.com/HPC/Articles/Porting-CUDA-to-HIP>

- Syntactically similar to CUDA
  - Most CUDA API calls can be converted in place: replace 'cuda' with 'hip'
  - Supports a strong subset of CUDA runtime functionality
  - You can "hipify" an existing CUDA code (<https://github.com/ROCm-Developer-Tools/HIPIFY>)
    - Eg. source-to-source conversion (hipify-perl)

# SYCL



- C++ Single-source Heterogeneous Programming for Acceleration Offload
  - **Standard** from the Khronos Group, <https://www.khronos.org/sycl/>
  - SYCL 2020 is the latest revision, more info at <https://sycl.tech/>
- Several backends available (SYCL implementation ecosystem)



# KOKKOS / RAJA

- Kokkos and RAJA provide portability to C++ programs via backends
  - Based on lambdas and templates
  - Cross-platform abstraction layers targeting all major HPC platforms (CPU, GPUs)
  - Big US labs and projects to support them
    - <https://github.com/kokkos/>
    - <https://github.com/LLNL/RAJA>
- Kokkos provides memory abstraction via “view” datatype
  - Separation of concerns between “where data is stored” and “where code is run”
  - Default memory and execution space selected at compile time

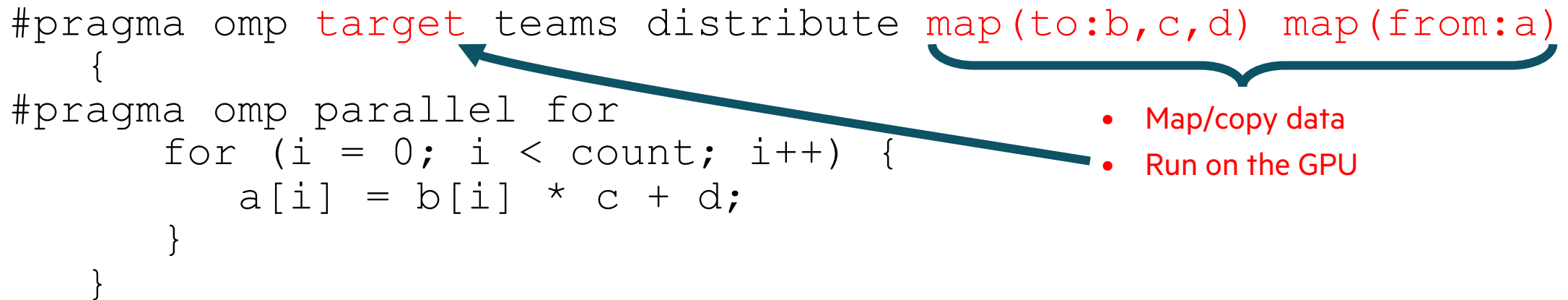
```
int n = 100000;  
Kokkos::View<double*> a("a", n), b("b", n);  
// Fill a and b - not shown  
Kokkos::parallel_for(n, KOKKOS_LAMDA(size_t const i) { a(i) += b(i); });
```

- Similarly, RAJA insulates application loop kernels from underlying architecture and programming model-specific implementation details



- De-facto standard for Shared-Memory Parallelization (OpenMP 1.0, 1997)
  - Directives-based (pragma's) approach
  - Easily add to an existing code, minimal disruption
- OpenMP 4.0 (2013) includes initial support for accelerators
  - Significant revisions/extensions in 4.5 (2015), 5.0 (2018), 5.1 (2020), 5.2 (2021)

```
#pragma omp target teams distribute map(to:b,c,d) map(from:a)
{
#pragma omp parallel for
    for (i = 0; i < count; i++) {
        a[i] = b[i] * c + d;
    }
}
```



- All vendors committed to supporting OpenMP offload





# C++



- C++17 introduced several parallel algorithms (Parallel STL)
  - Extended in C++20, C++23 and beyond (*executors*)
  - See `<numeric>`, `<algorithm>`, `<execution>`
- Compiler support for GPU accelerated is emerging
  - Evolving support as part of the C++ standard
- Portability will be guarantee as part of the C++ standard
  - Can run on CPUs and GPUs
  - Interesting option in the long run

