Introduction to accelerated computing ecosystem

PRACE Parallel programming course with Python

Sagar Dolas
Program Lea, Innovation Lab
SURF

Amsterdam Science Park

Agenda

- Overview of accelerated computing
- GPU Architecture
- Software ecosystem & applications
- Alternative computing paradigms

Go to menti.com and enter the code : **9671 7210**

Main message

- Difference between CPU vs GPU
- State of GPUs internationally
- Different GPU architecture
- Programming GPUs
- New computing paradigms

Snellius : Dutch National Supercomputer

AMD

Lenovo

NVIDIA

intel

Snellius

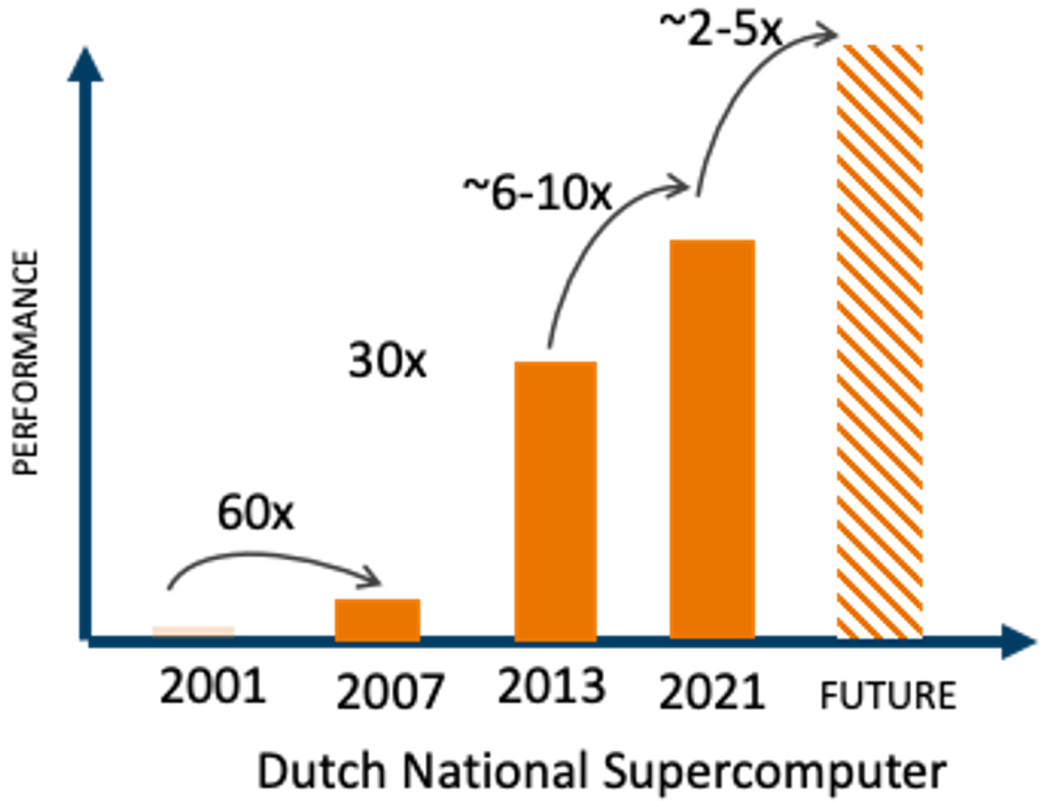
Phase 1:

- 76,832 cores (1.6 ×)
- 144 GPUs (3 Pflop/s, 14 ×)
- Total peak: 6.1 Pflop/s (3.4 ×)

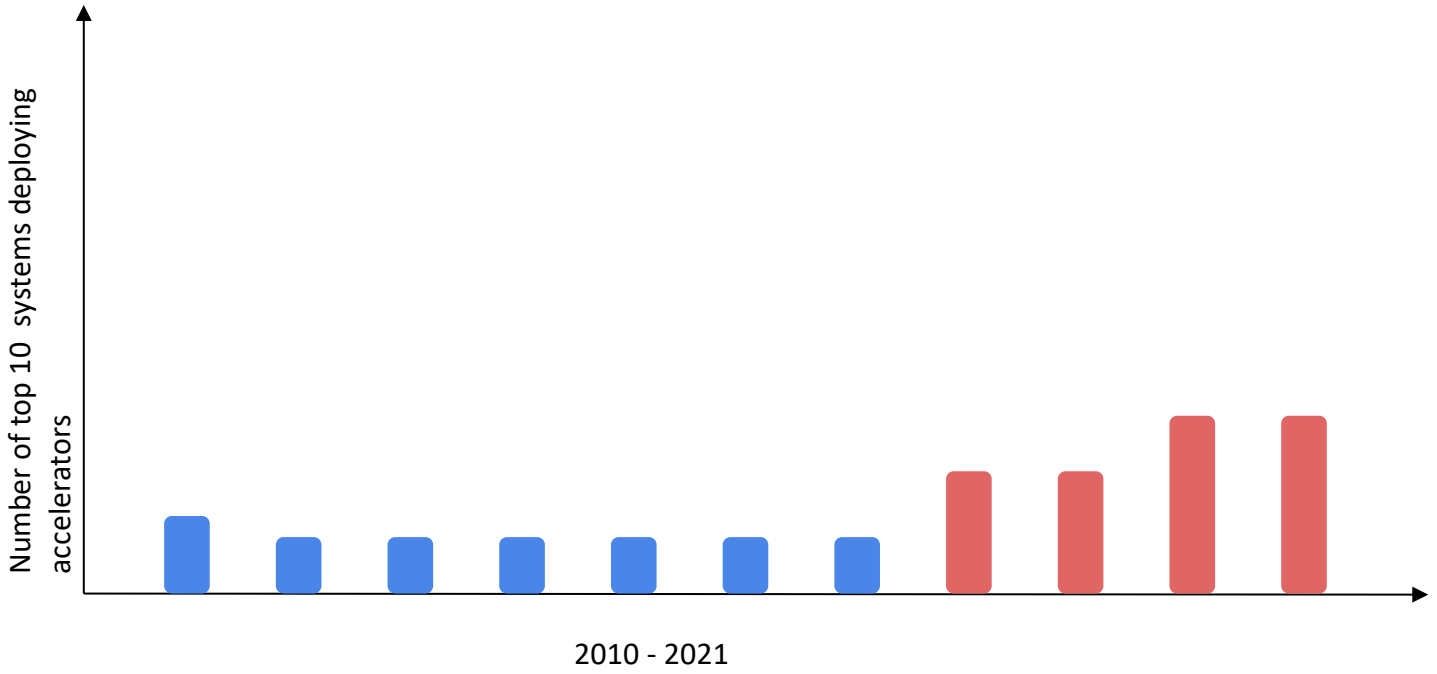
Phase 1 and 2

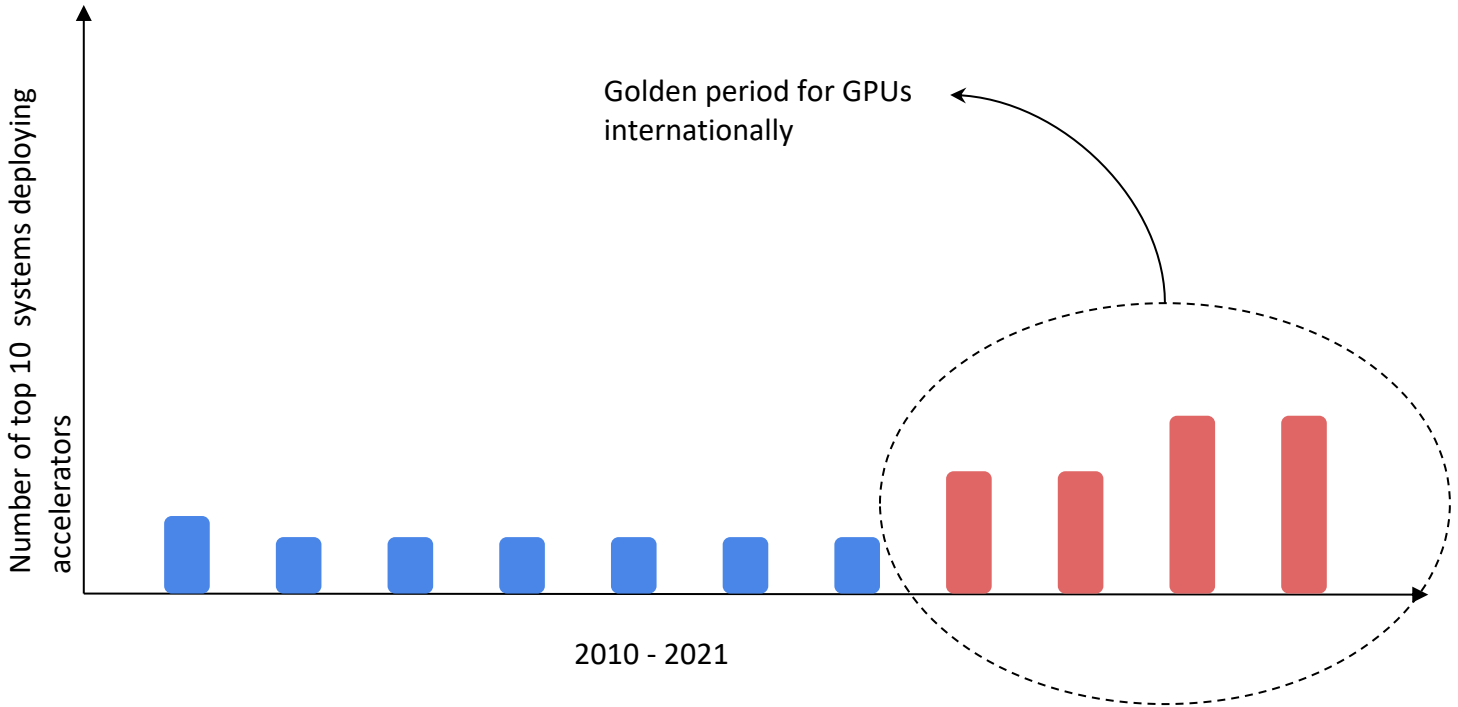
- Total peak: 11.2 Pflop/s (6.2 ×)

- Full system, based on choice for Phase 3:
  - > 200,000 cores (> 4 ×)
  - Total peak: 13.6 – 21.5 Pflop/s (7.6 – 11.9 ×)


~2-5x
~6-10x
30x
60x
PERFORMANCE
2001
2007
2013
2021
FUTURE
Dutch National Supercomputer

Let’s look at the supercomputing trends with respect to GPUs


Number of top 10 systems deploying accelerators
2010 - 2021


Number of top 10 systems deploying accelerators
Golden period for GPUs internationally
2010 - 2021

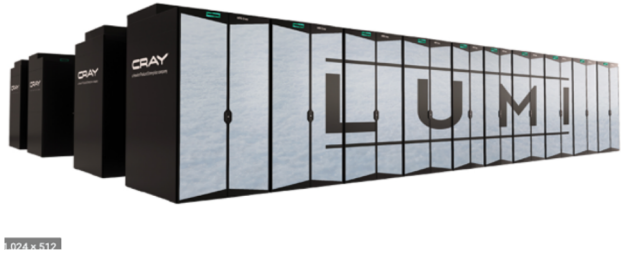
Let’s look at the pre-exascale systems & exascale system architecture trends

1.5 ExFLOPs/s : AMD CPUs and GPUs

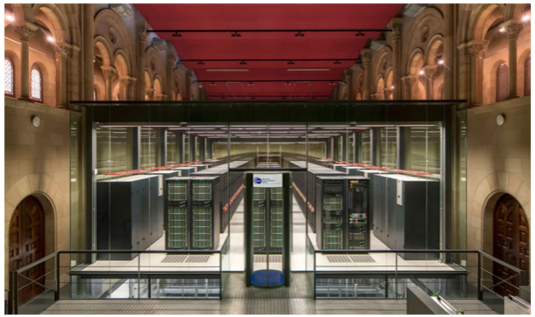
2 ExFLOPs/s : AMD CPUs and GPUs

~2 ExFLOPs/s : Intel CPUs and GPUs

SURF



~350 PetaFLOPs: AMD CPUs and GPUs



N/A

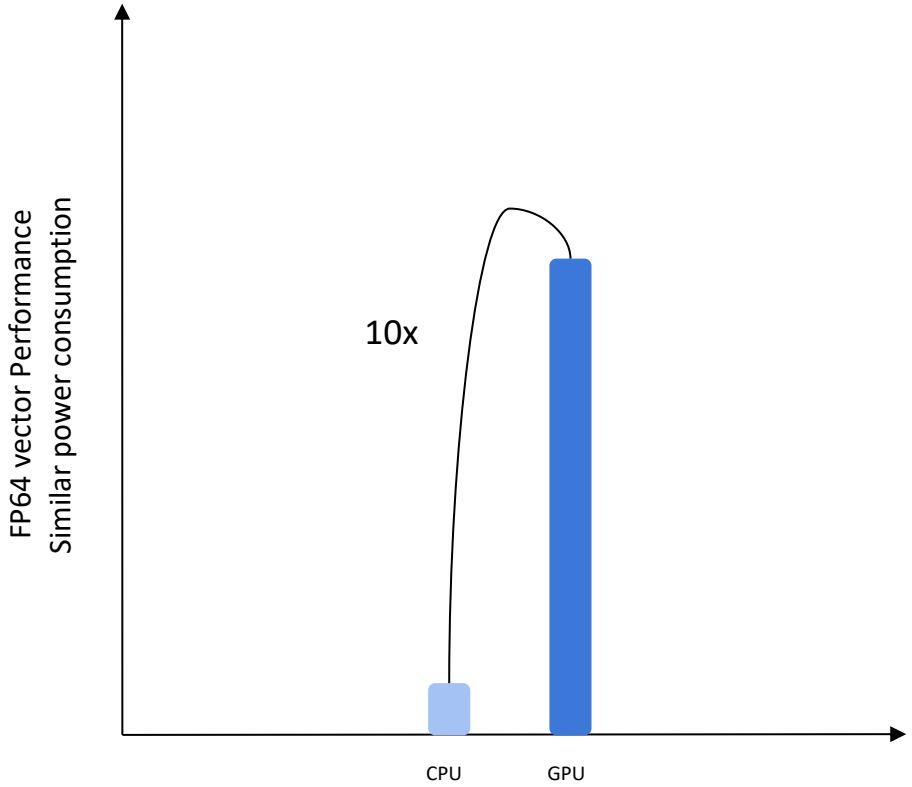
~250 PetaFLOPs : Intel CPUs and NVIDIA GPUs

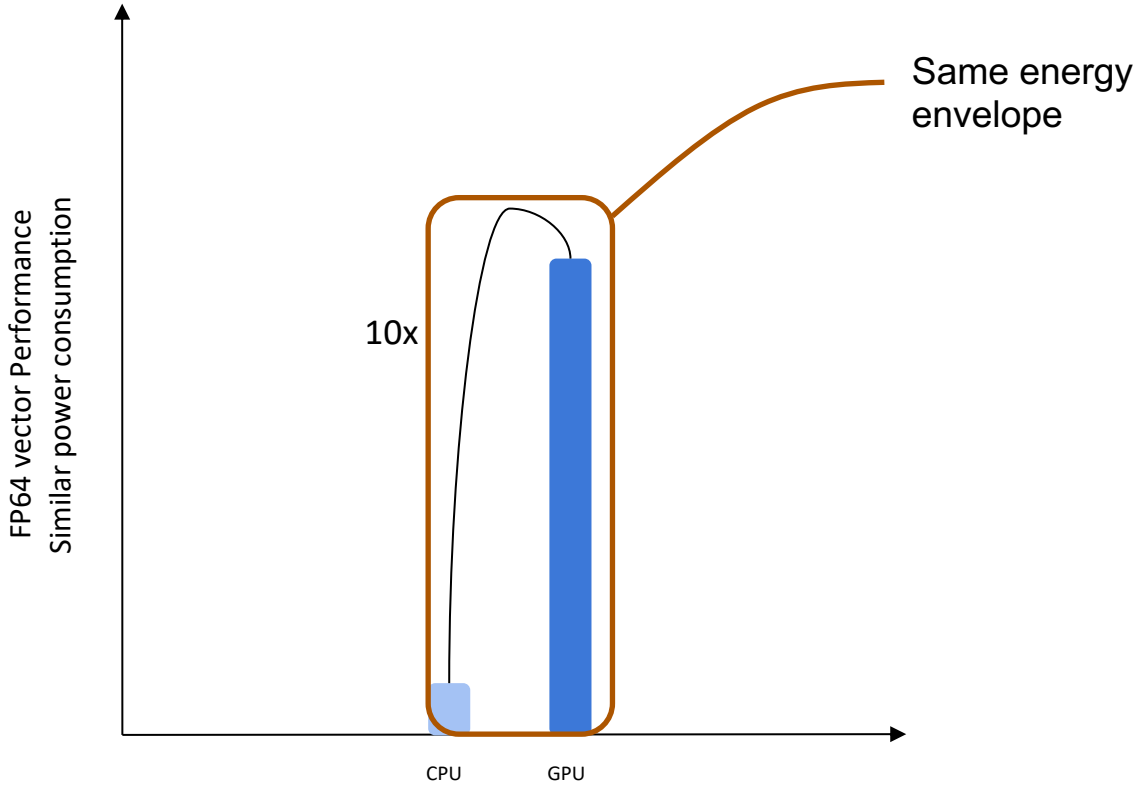
What’s the difference between CPU & GPU

"If you were plowing a field, which would you rather use? Two strong oxen or 1024 chickens?"

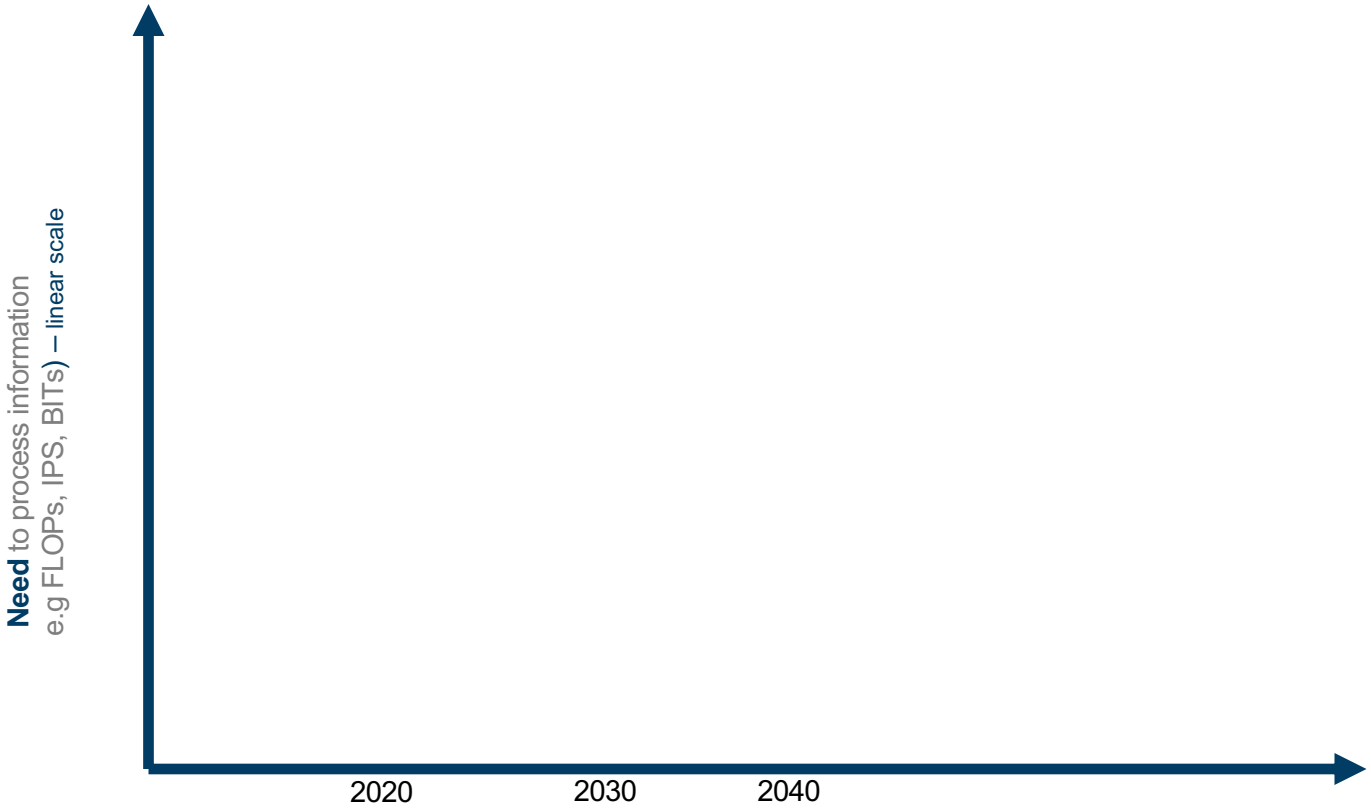
—Seymour Cray

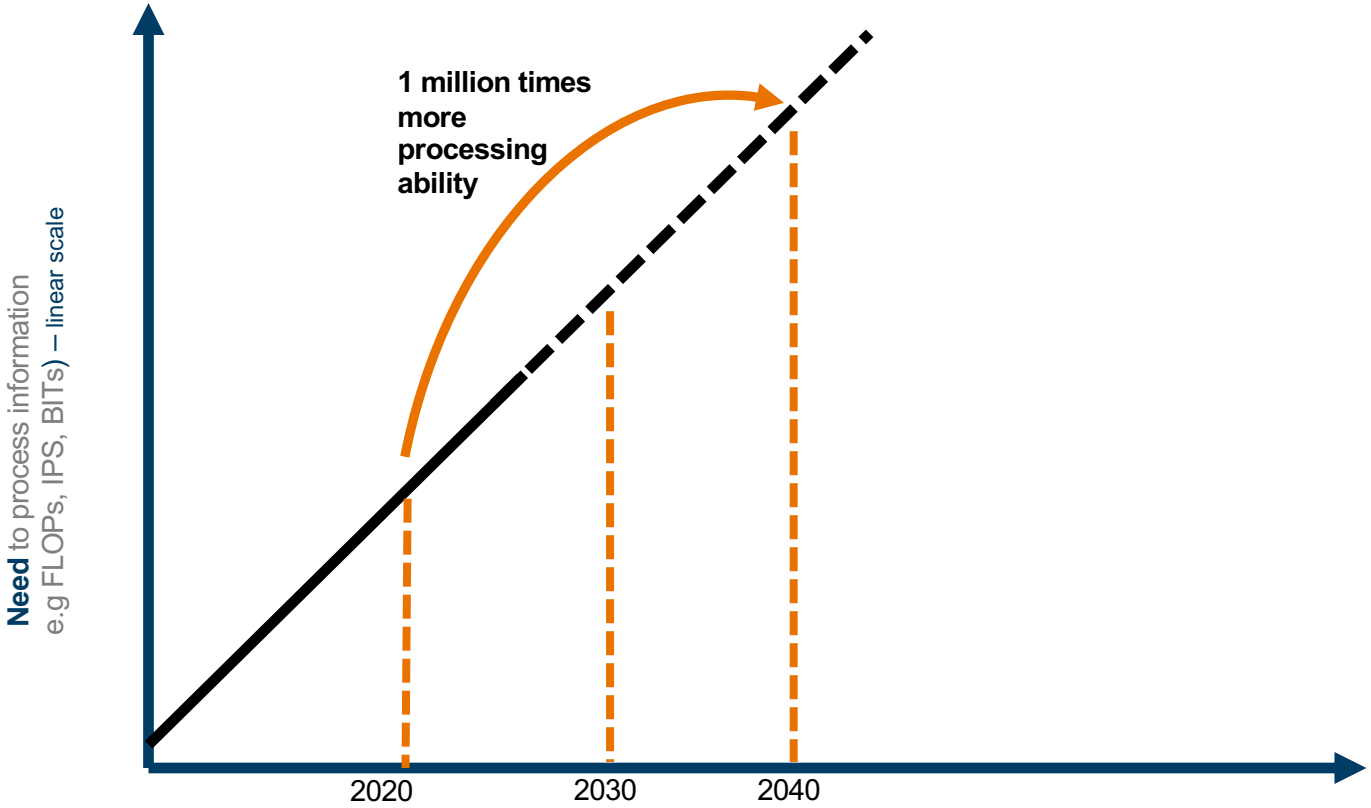
CPU are much better at managing lot of tasks
GPUs are specialised for particular workloads involving lot of threads & parallelism


FP64 vector Performance
Similar power consumption
10x
CPU
GPU

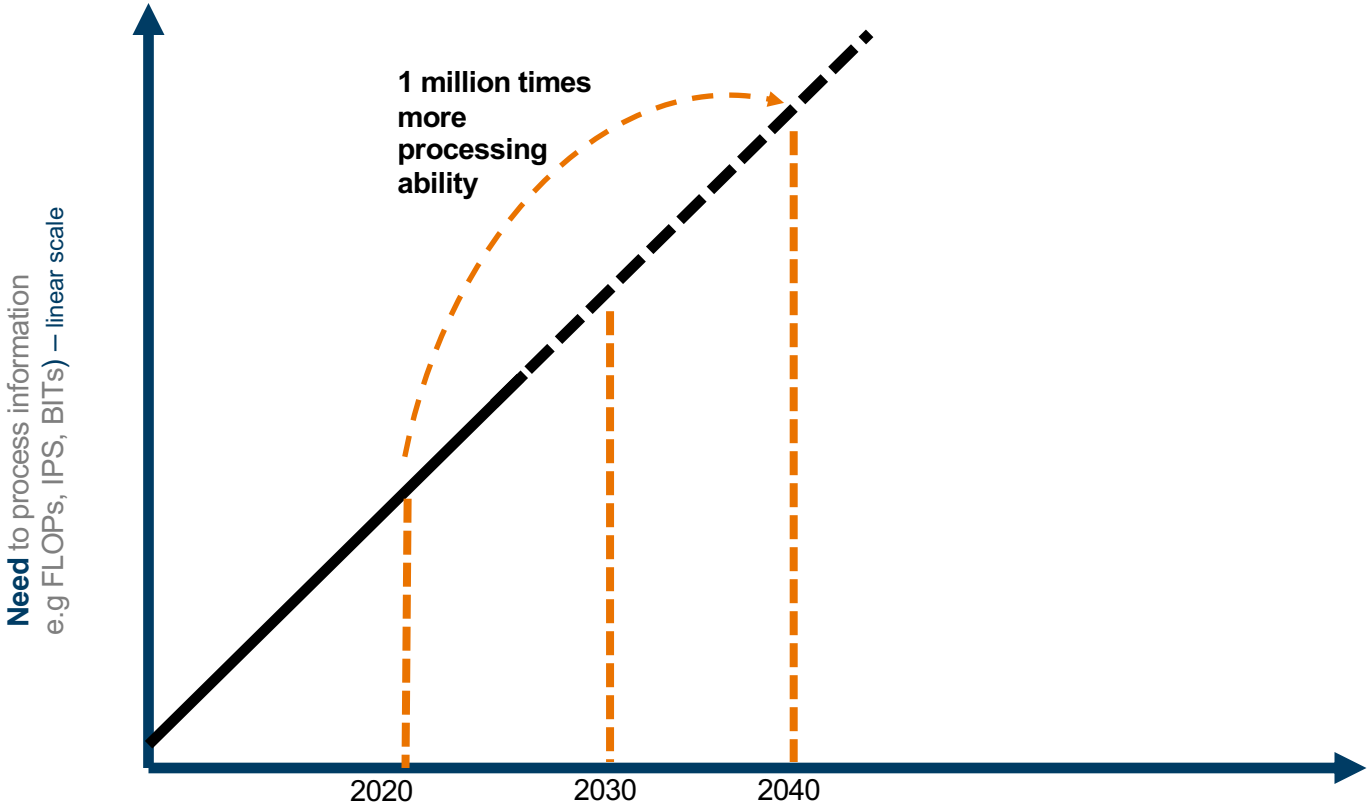

FP64 vector Performance
Similar power consumption
Same energy
envelope
10x
CPU
GPU

Let’s look at the need for computing


Need to process information
e.g FLOPs, IPS, BITs) – linear scale
2020
2030
2040



HTTPS://WWW.SRC.ORG/ABOUT/DECADAL-PLAN/



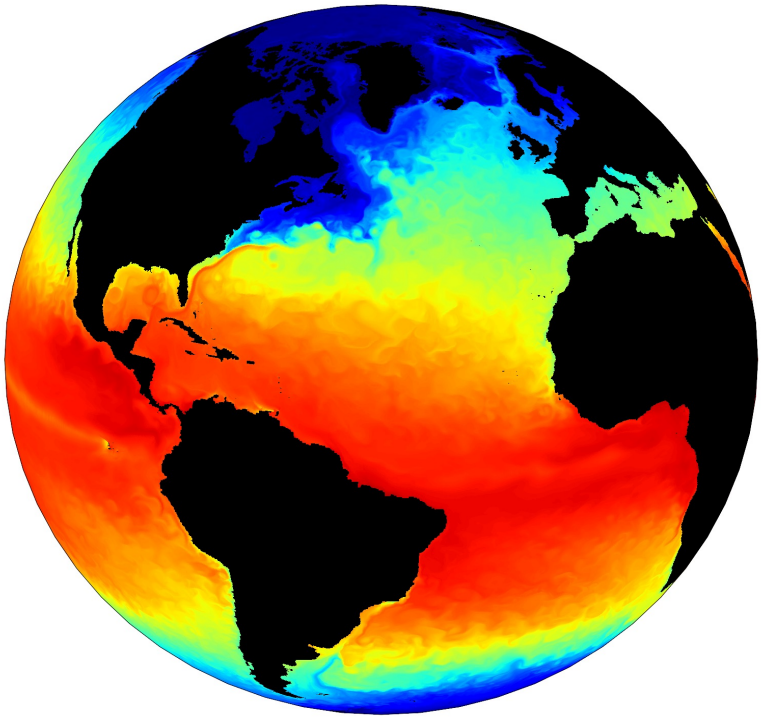
HTTPS://WWW.SRC.ORG/ABOUT/DECADAL-PLAN/

Ocean component output of an Earth System Model simulation - Rene va
Westen, Henk Dijkstra, Ocean Eddies Strongly Affect Global Mean
Sea-level Projections - Science Advances 7, 2021

SURF

Think about **global** climate simulations

1km x 1km x 1km resolution



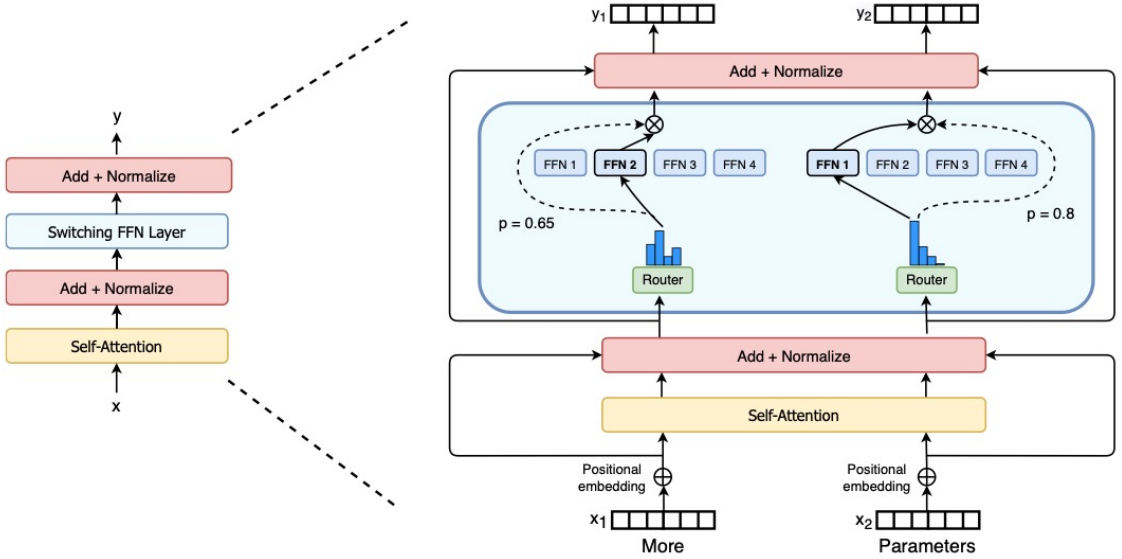
Ocean component output of an Earth System Model simulation - Rene van Westen, Henk Dijkstra, Ocean Eddies Strongly Affect Global Mean Sea-level Projections - Science Advances 7, 2021

Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

William Fedus*
LIAMFEDUS@GOOGLE.COM

Barret Zoph*
BARRETZOPH@GOOGLE.COM

Noam Shazeer
NOAM@GOOGLE.COM
Google, Mountain View, CA 94043, USA



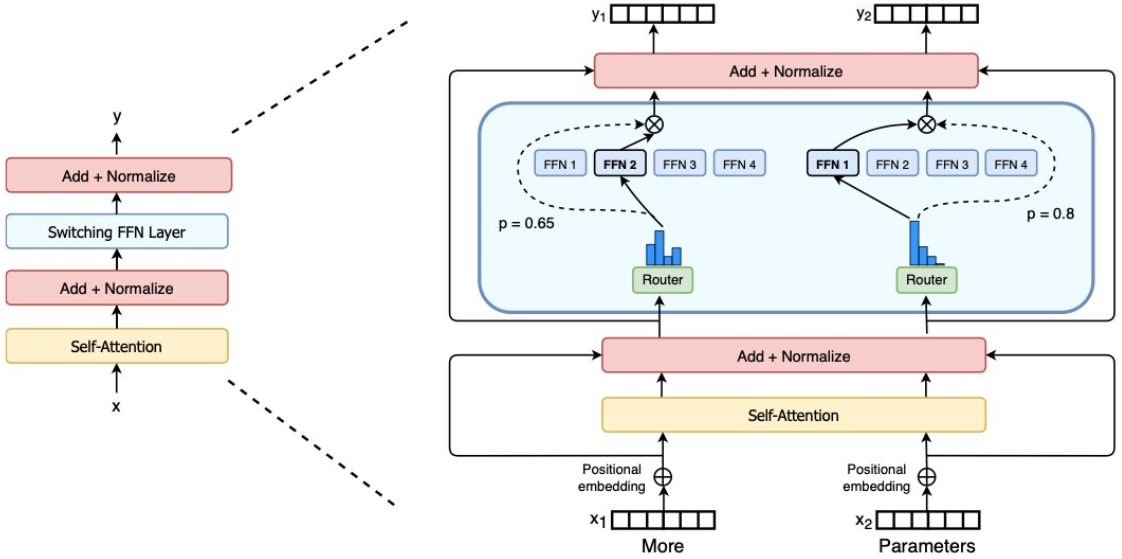
Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

William Fedus*
LIAMFEDUS@GOOGLE.COM

Barret Zoph*
BARRETZOPH@GOOGLE.COM

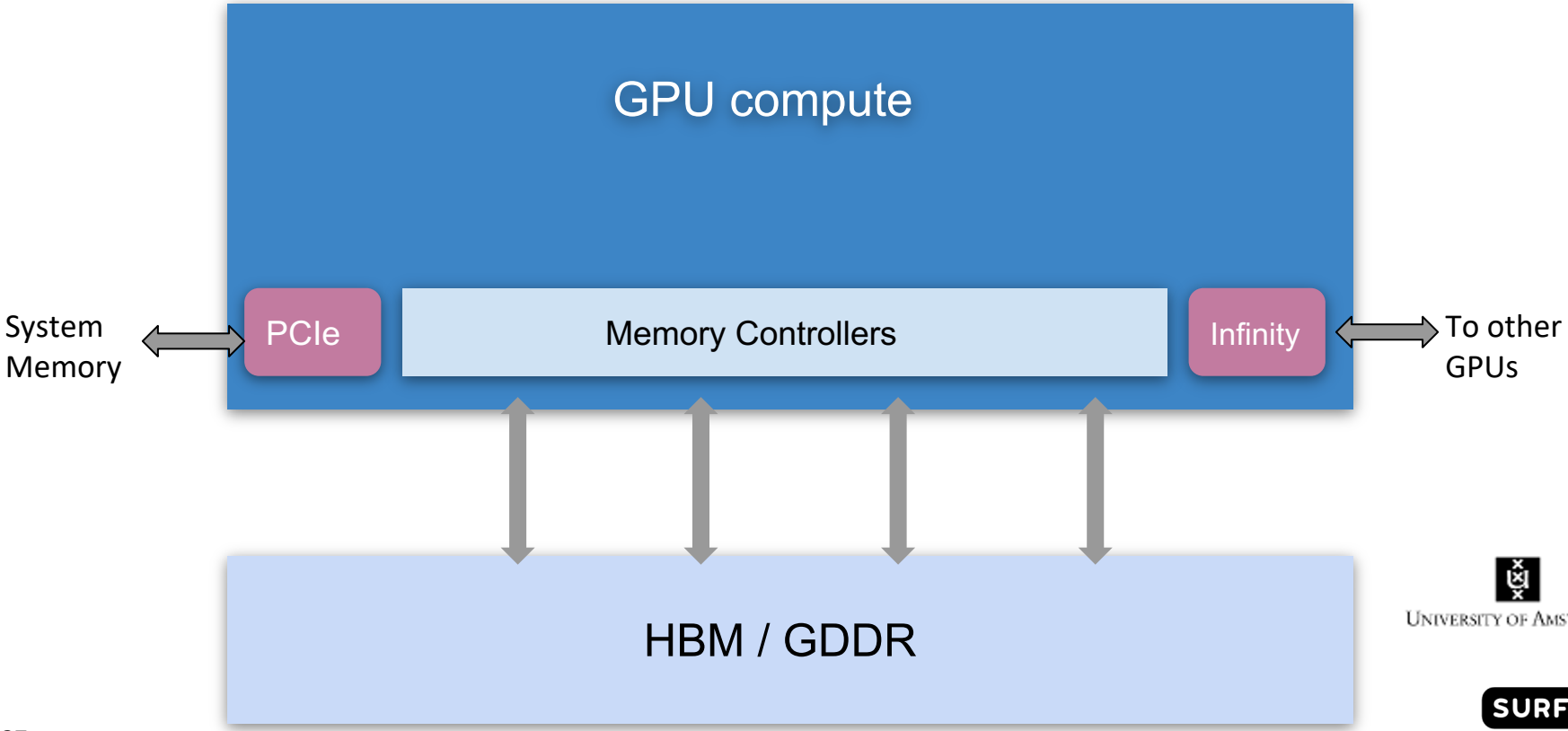
Noam Shazeer
NOAM@GOOGLE.COM
Google, Mountain View, CA 94043, USA

Model parameters doubling every 4 months or so !!!!

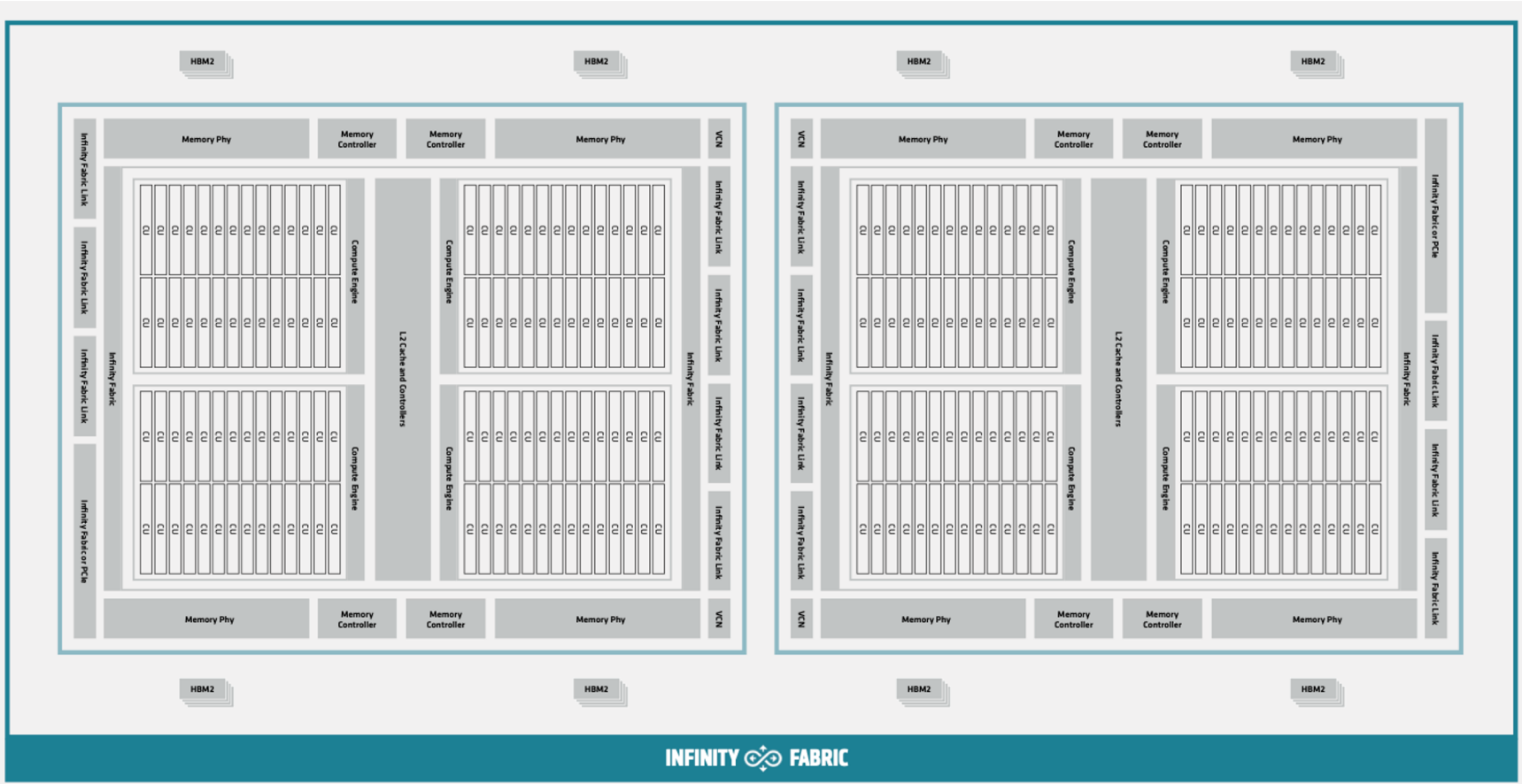


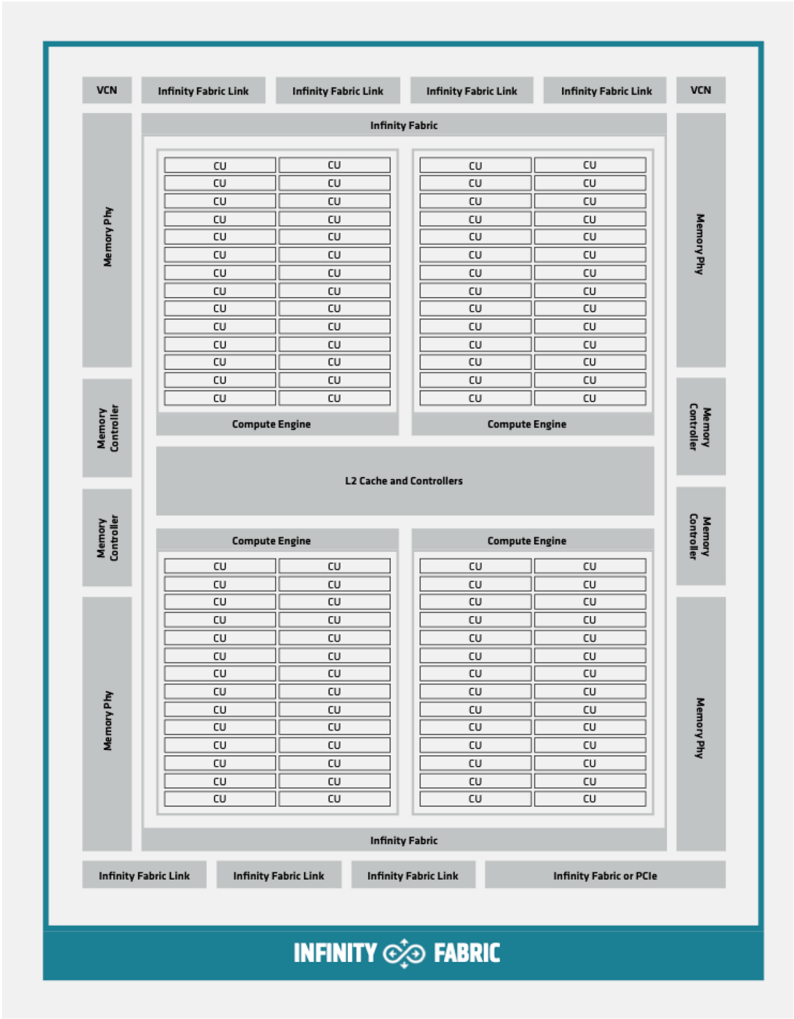
Let’s look at the GPU architecture

GPU Memory subsystem



AMD GPU architecture

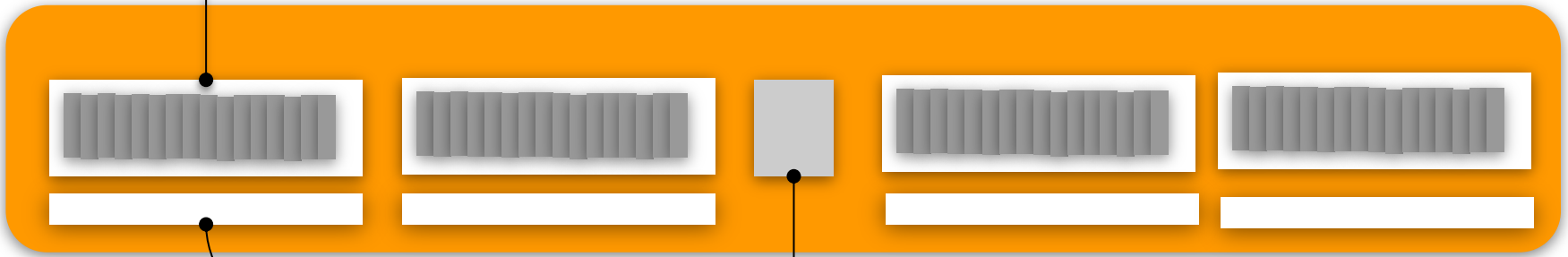

HBM2
Memory Phy
Memory Controller
VCN
Infinity Fabric Link
Infinity Fabric or PCIe
Infinity Fabric
L2 Cache and Controllers
Compute Engine
CU
INFINITY FABRIC


VCN
Infinity Fabric Link
Infinity Fabric Link
Infinity Fabric Link
Infinity Fabric Link
VCN
Infinity Fabric
Memory Phy
Memory Phy
CU
Compute Engine
Compute Engine
Memory Controller
Memory Controller
L2 Cache and Controllers
Memory Controller
Memory Controller
Compute Engine
Compute Engine
Memory Phy
Memory Phy
Infinity Fabric
Infinity Fabric Link
Infinity Fabric Link
Infinity Fabric Link
Infinity Fabric or PCIe
INFINITY FABRIC

Compute Unit

4x SIMD 16 lane

- Total throughput : 64 SP ops / clock
- Support upto 10 wavefronts per SIMD-16 active at same time



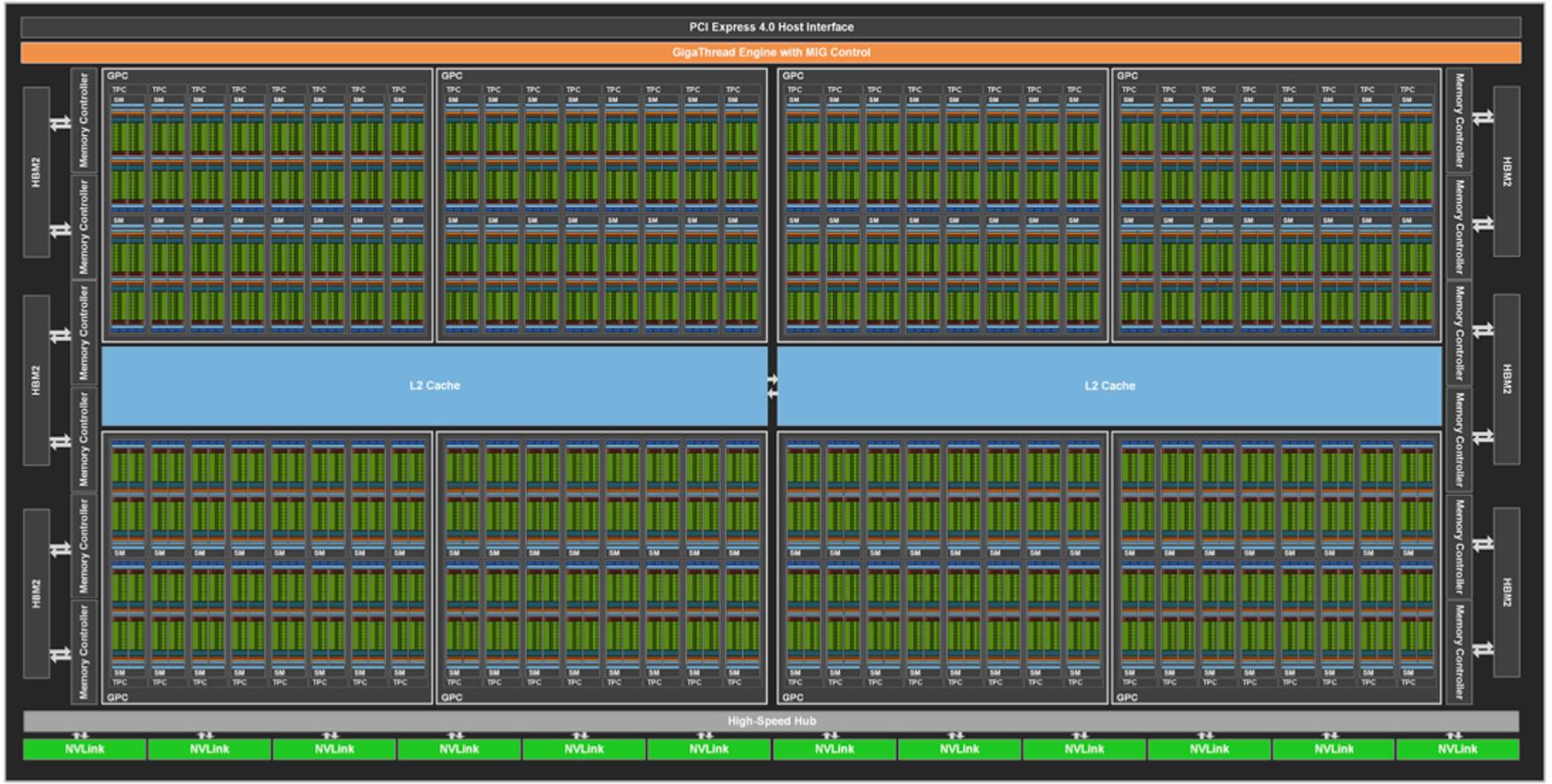
Scalar Unit

- Shared by all threads in wavefronts
- Used for flow control
- Pointer arithmetic
- 8 KB general purpose register

64 KB Vector registers
~ each register is 64 4-byte wide entries
256 total registers

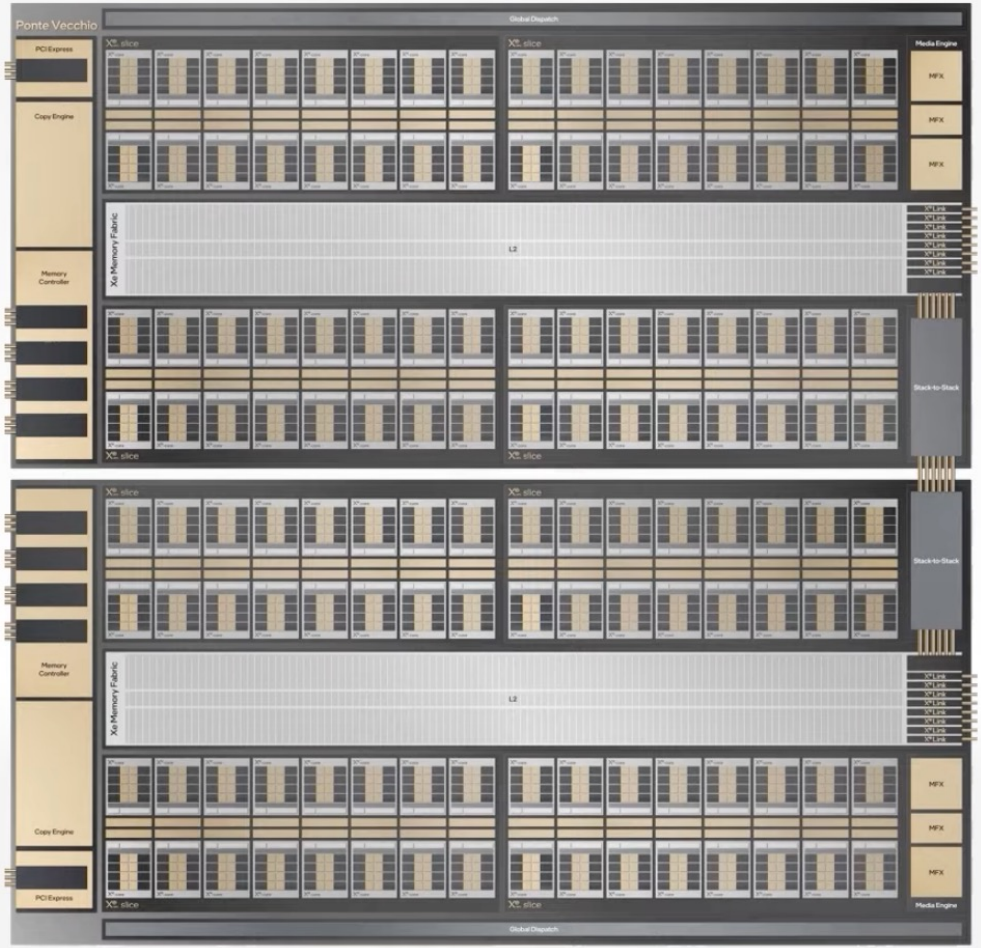
SURF

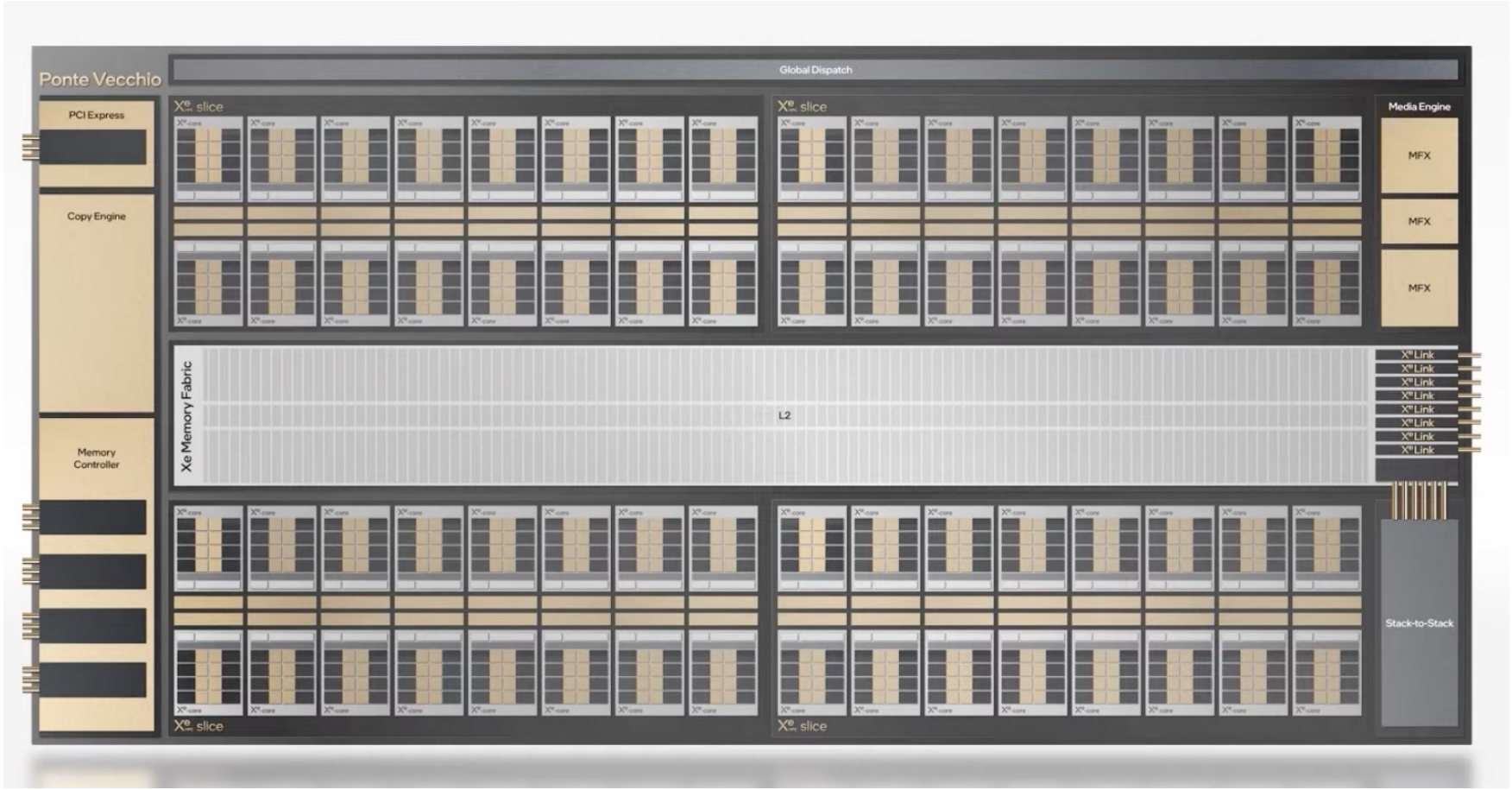
NIVIDIA GPU architecture

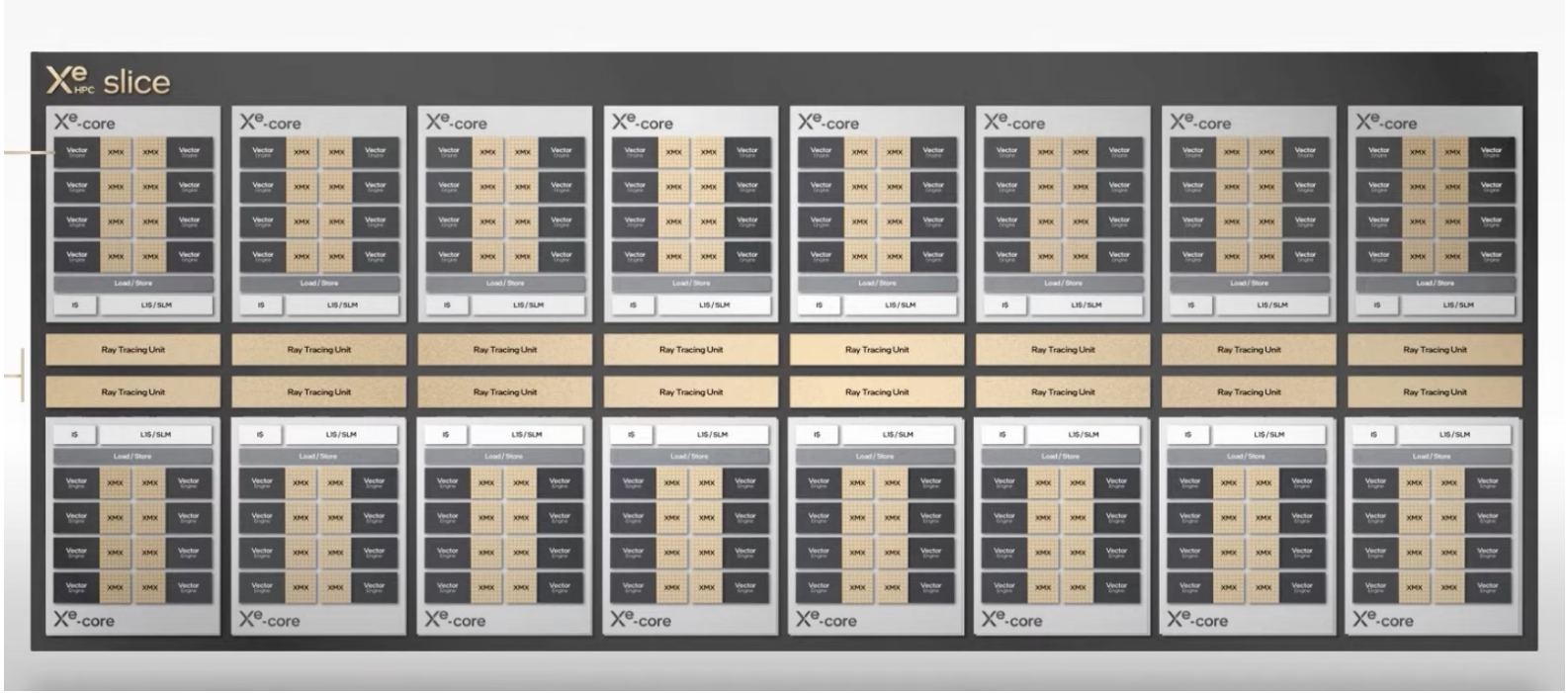

PCI Express 4.0 Host Interface
GigaThread Engine with MIG Control
GPC
TPC
SM
L2 Cache
L2 Cache
Memory Controller
HBM2
High-Speed Hub
NVLink


SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
TENSOR CORE
LD/ ST
SFU
192KB L1 Data Cache / Shared Memory
Tex

Intel GPU architecture

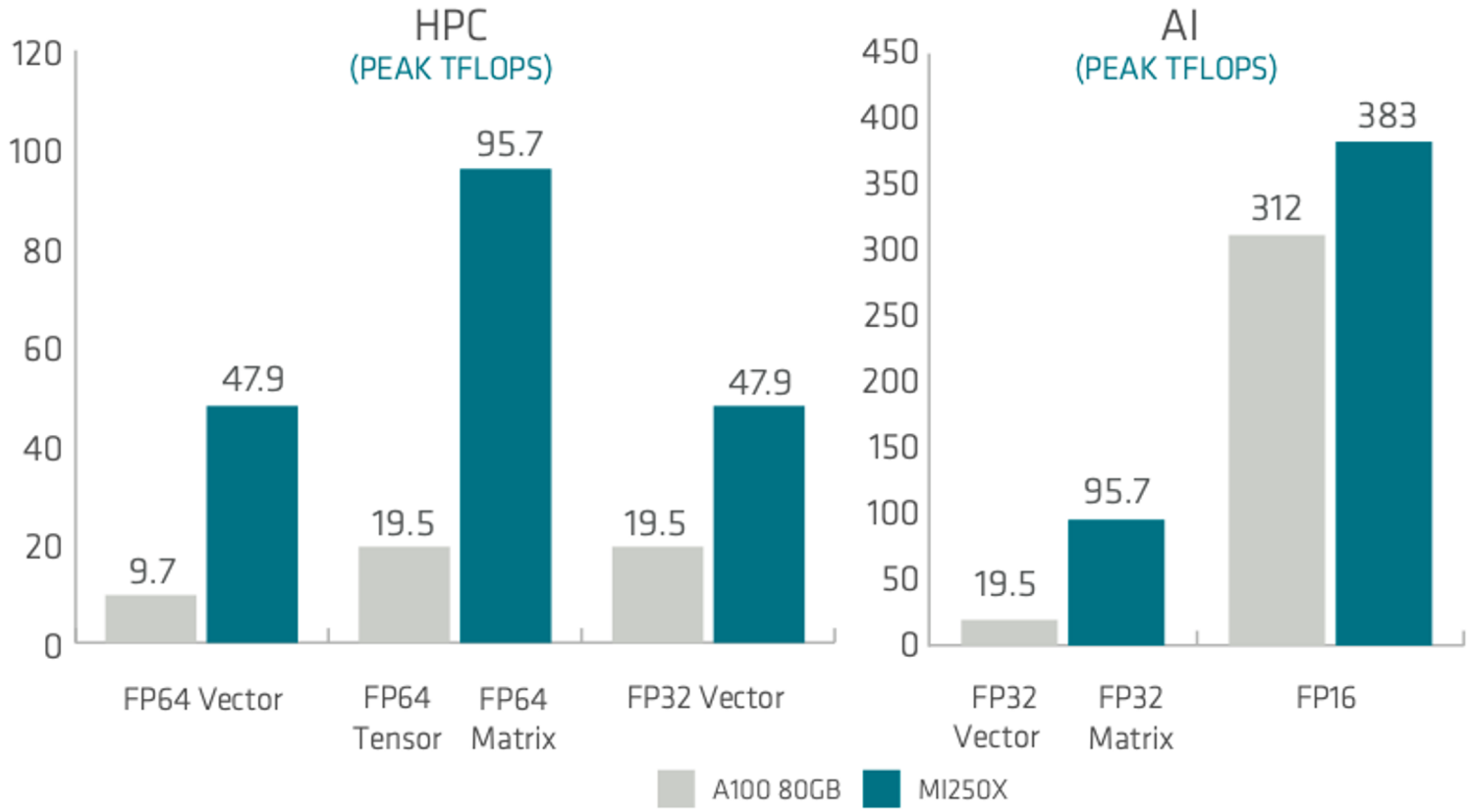

Ponte Vecchio
Global Dispatch
PCI Express
Copy Engine
Memory Controller
Xe slice
Media Engine
MFX
Xe Memory Fabric
L2
Xe Link
Stack-to-Stack
Global Dispatch


Ponte Vecchio
Global Dispatch
PCI Express
Copy Engine
Memory Controller
Xe slice
Xe slice
Media Engine
MFX
MFX
MFX
Xe Memory Fabric
L2
Xe Link
Xe Link
Xe Link
Xe Link
Xe Link
Xe Link
Xe Link
Xe Link
Stack-to-Stack
Xe slice
Xe slice


Xe HPC slice
Xe-core
Vector Engine
XMX
Load / Store
IS
L1S / SLM
Ray Tracing Unit

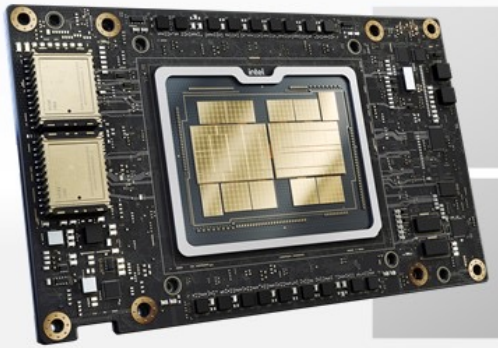

Xe-core
Vector Engine
XMX
Load / Store
I$
L1$ / SLM
Xe-core
Compute Building Block of Xe HPC GPUs
8
Vector Engines
512 bit per engine
8
Matrix Engines
4096 bit per engine
Load / Store
512 B/CLK
Cache
L1$/ SLM (512KB), I$

Some Performance Numbers


HPC
(PEAK TFLOPS)
120
100
80
60
40
20
0
9.7
47.9
19.5
95.7
19.5
47.9
FP64 Vector
FP64 Tensor
FP64 Matrix
FP32 Vector
AI
(PEAK TFLOPS)
450
400
350
300
250
200
150
100
50
0
19.5
95.7
312
383
FP32 Vector
FP32 Matrix
FP16
A100 80GB
MI250X

Ponte Vecchio

Execution Progress



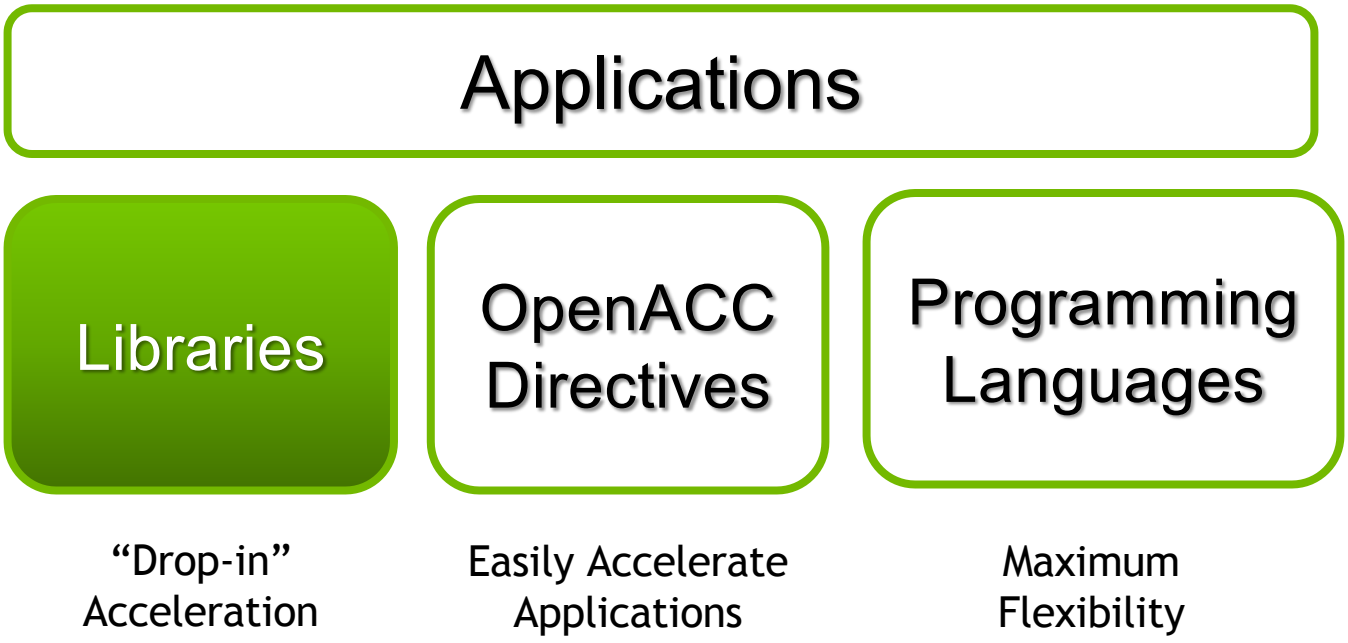
A0 Silicon Current Status

> 45 TFLOPS	FP32 Throughput
> 5 TBps	Memory Fabric Bandwidth
> 2 TBps	Connectivity Bandwidth

For workloads and configurations visit www.intel.com/ArchDay21claims . Results may vary.

Programming GPUs

3 Ways to Accelerate Applications



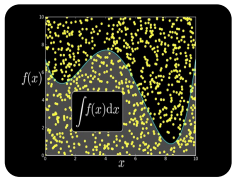
SURF

Libraries: Easy, High-Quality Acceleration

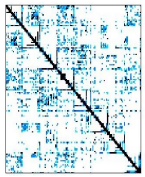
- **Ease of use:** Using libraries enables GPU acceleration without in-depth knowledge of GPU programming
- **"Drop-in":** Many GPU-accelerated libraries follow standard APIs, thus enabling acceleration with minimal code changes (replacing MKL/IPP/FFTW/…)
- **Quality:** Libraries offer high-quality implementations of functions encountered in a broad range of applications
- **Performance:** NVIDIA libraries are tuned by experts

Some GPU-accelerated Libraries

NVIDIA cuBLAS



NVIDIA cuRAND



NVIDIA cuSPARSE

NVIDIA NPP

GPU VSIPL

Vector Signal Image Processing

CULA tools

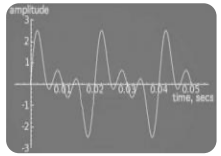
GPU Accelerated Linear Algebra

MAGMA

ICL UT

Matrix Algebra on GPU and Multicore

open source initiative



NVIDIA cuFFT

ROGUE WAVE SOFTWARE

IMSL Library

ArrayFire

Matrix Computations

CUSP

Sparse Linear Algebra

open source initiative

Thrust

C++ STL Features for CUDA

open source initiative

SURF

3 Steps to CUDA-accelerated application

- **Step 1:** Substitute library calls with equivalent CUDA library calls

```
saxpy ( … )  ⊳  cublasSaxpy ( … )
```

- **Step 2:** Manage data locality

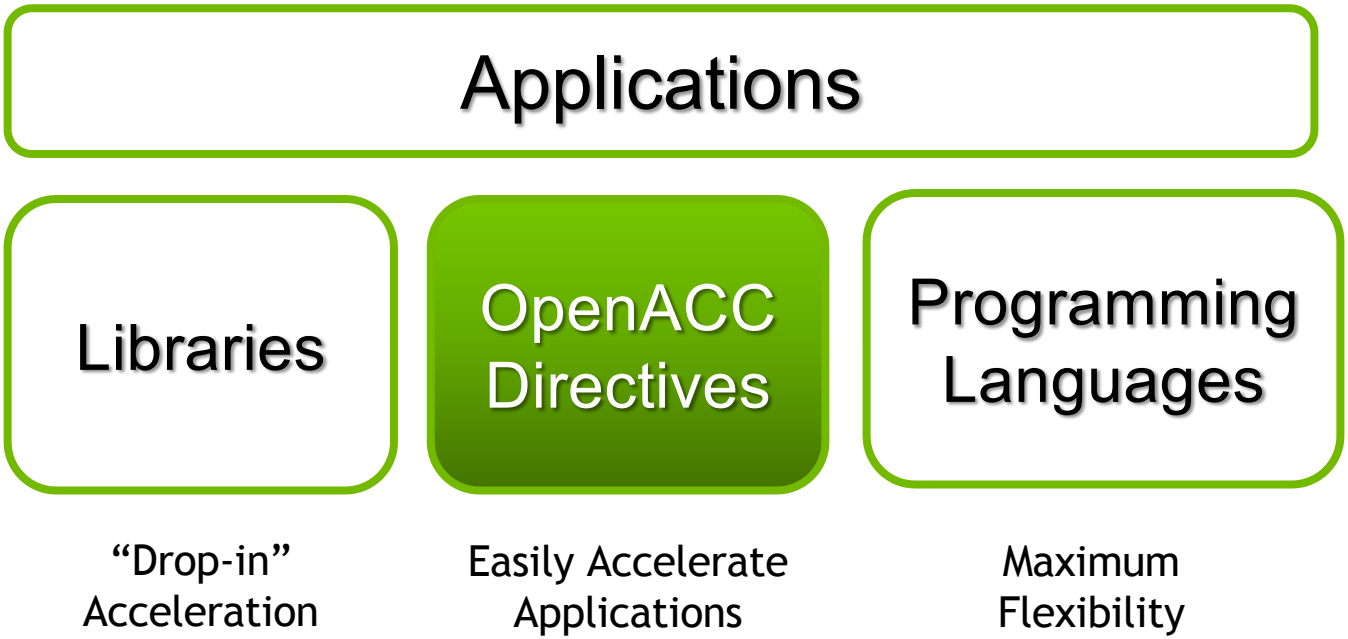
  - with CUDA: `cudaMalloc(), cudaMemcpy(), etc.`
  - with CUBLAS: `cublasAlloc(), cublasSetVector(), etc.`

- **Step 3:** Rebuild and link your CUDA Library-accelerated application

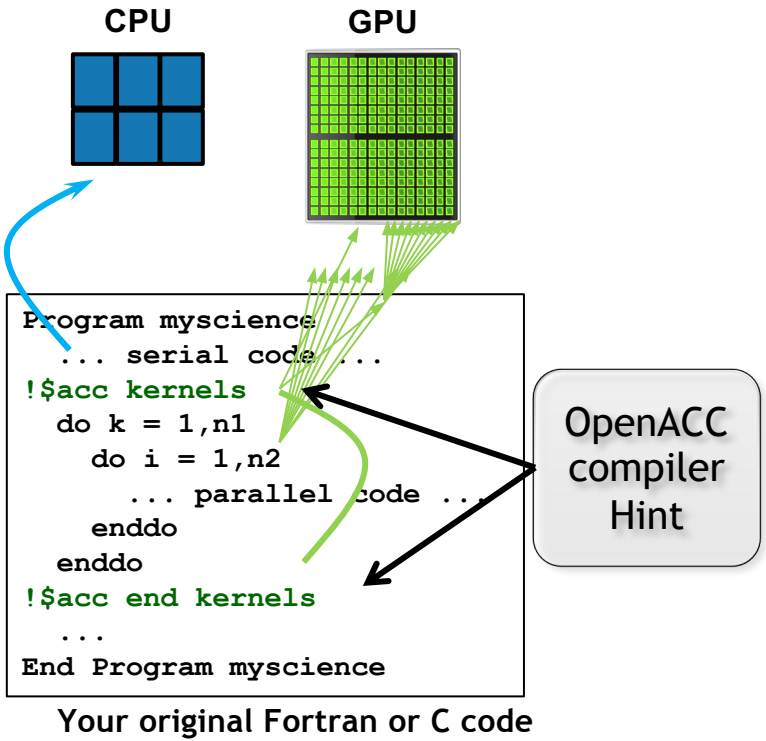
```
nvcc myobj.o –l cublas
```

SURF

3 Ways to Accelerate Applications



OpenACC Directives



Simple Compiler hints

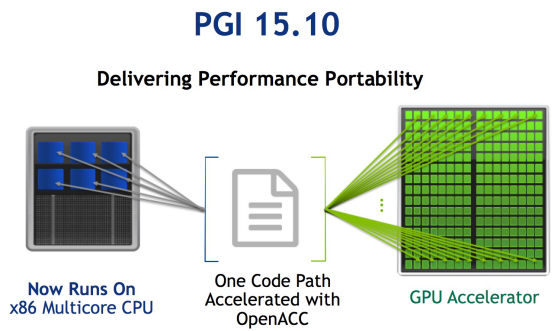
Compiler Parallelizes code

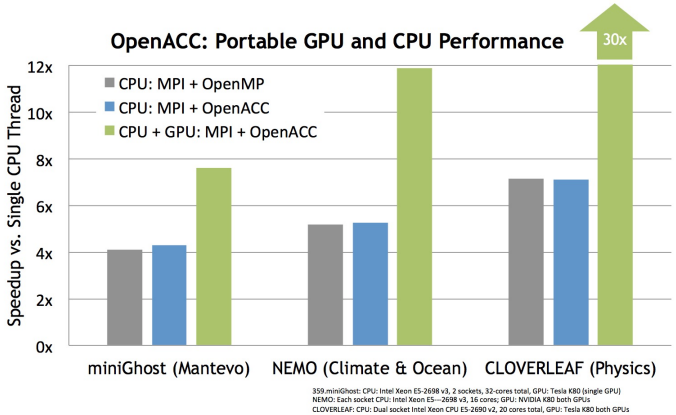
Works on many-core GPUs & multicore CPUs

OpenACC
DIRECTIVES FOR ACCELERATORS

OpenACC: The Standard for GPU Directives

- **Easy:** Directives are the easy path to accelerate compute intensive applications
- **Open:** OpenACC is an open GPU directives standard, making GPU programming straightforward and portable across parallel and multi-core processors
- **Powerful:** GPU Directives allow complete access to the massive parallel power of a GPU



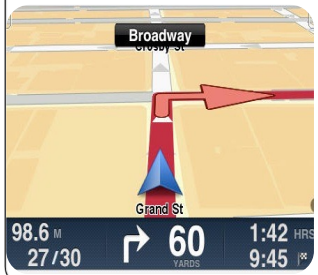


SURF

Directives: Easy & Powerful

Real-Time Object Detection

Global Manufacturer of Navigation Systems



5x in 40 Hours

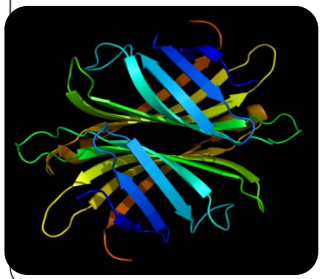
Valuation of Stock Portfolios using Monte Carlo

Global Technology Consulting Company

2x in 4 Hours

Interaction of Solvents and Biomolecules

University of Texas at San Antonio



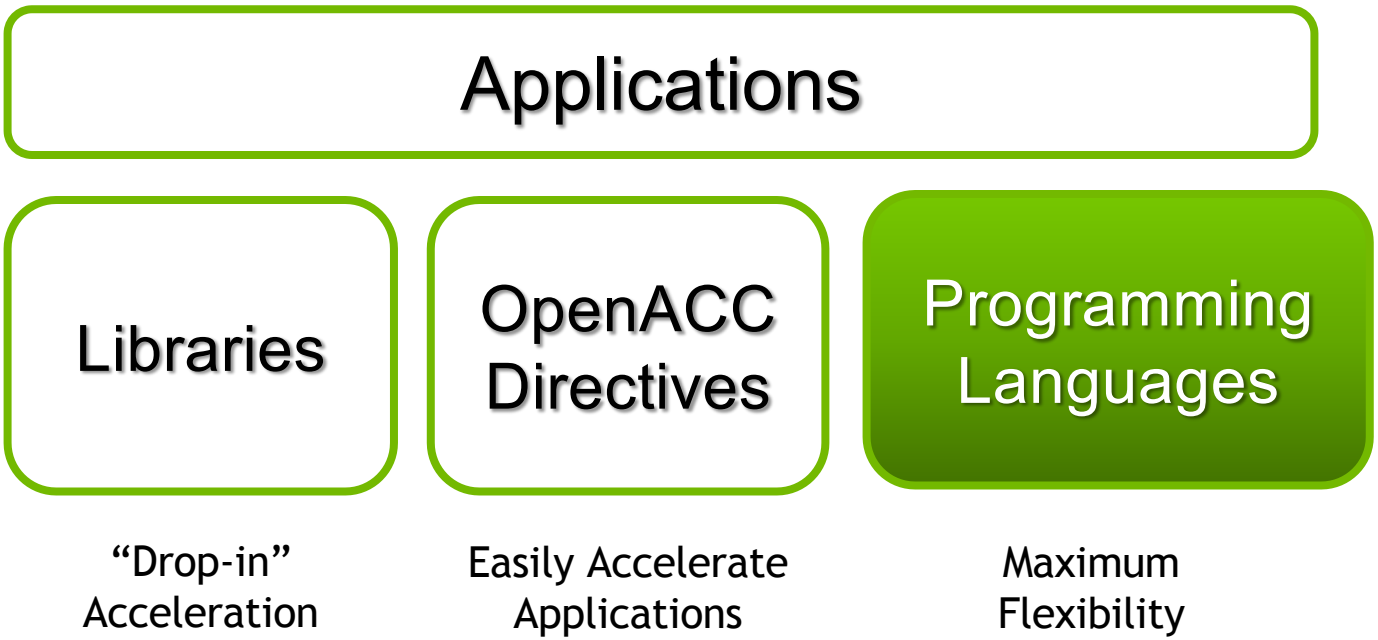
5x in 8 Hours

"Optimizing code with directives is quite easy, especially compared to CPU threads or writing CUDA kernels. The most important thing is avoiding restructuring of existing code for production applications."

-- Developer at the Global Manufacturer of Navigation Systems

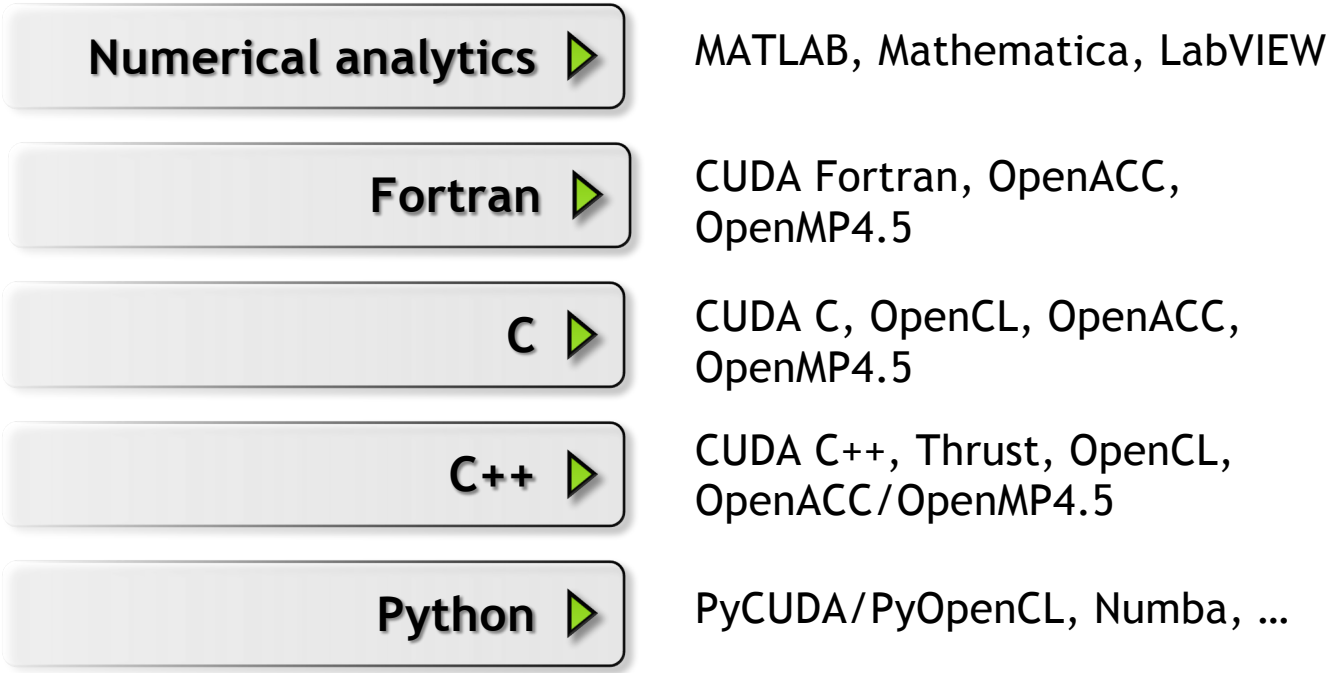
SURF

3 Ways to Accelerate Applications



SURF

GPU Programming Languages

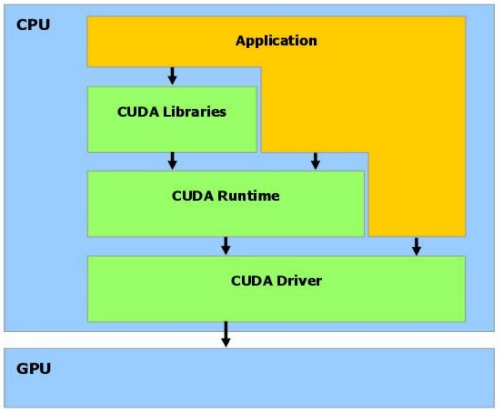


SURF

CUDA

- “Compute Unified Device Architecture”
- General purpose programming model
  - User kicks off batches of threads on the GPU
  - GPU = dedicated super-threaded, massively data parallel co-processor

- Targeted software stack
- Driver for loading computation programs into GPU
  - - Standalone Driver - Optimized for computation
  - - Explicit GPU memory management



CUDA Programming Model

- The GPU is viewed as a compute device that:

- Is a coprocessor to the CPU or host
- Has its own DRAM (device memory)
- Runs many threads in parallel
  - - Hardware switching between threads (in 1 cycle) on long-latency memory reference
  - - Overprovision (1000s of threads) → hide latencies

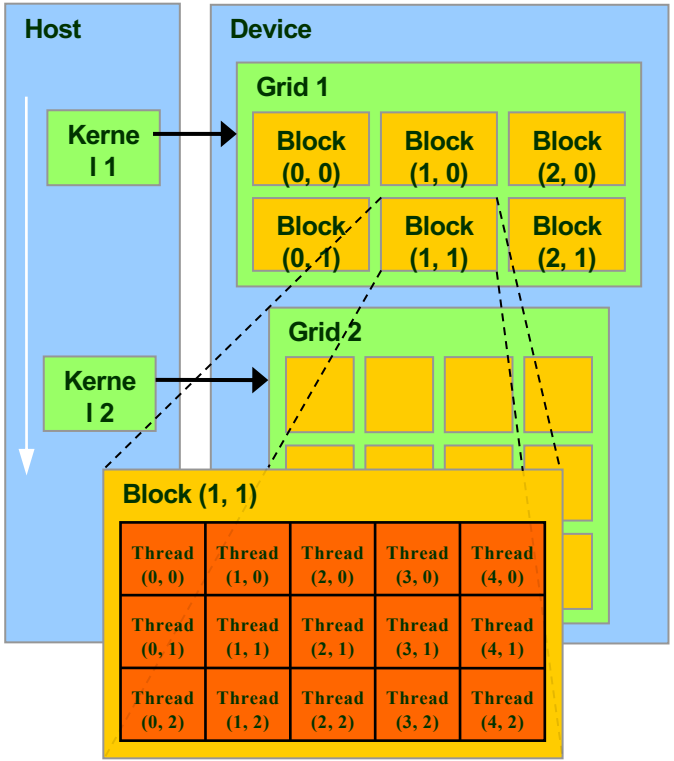
- Data-parallel portions of an application are executed on the device as kernels which run in parallel on many threads

- Differences between GPU and CPU threads
  - - GPU threads are extremely lightweight
    - - Very little creation overhead
  - - GPU needs 1000s of threads for full efficiency
    - - Multi-core CPU needs only a few

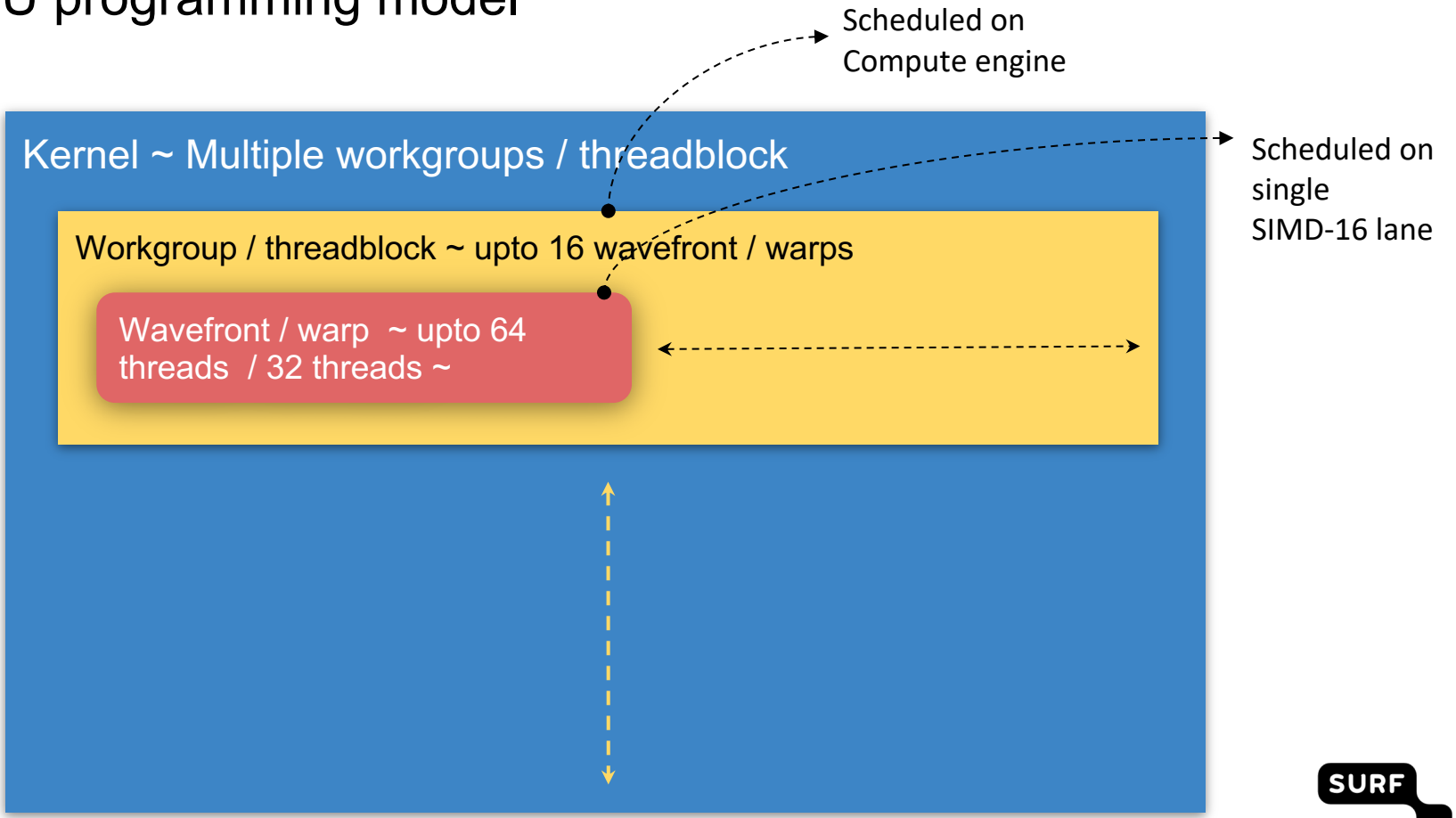
SURF

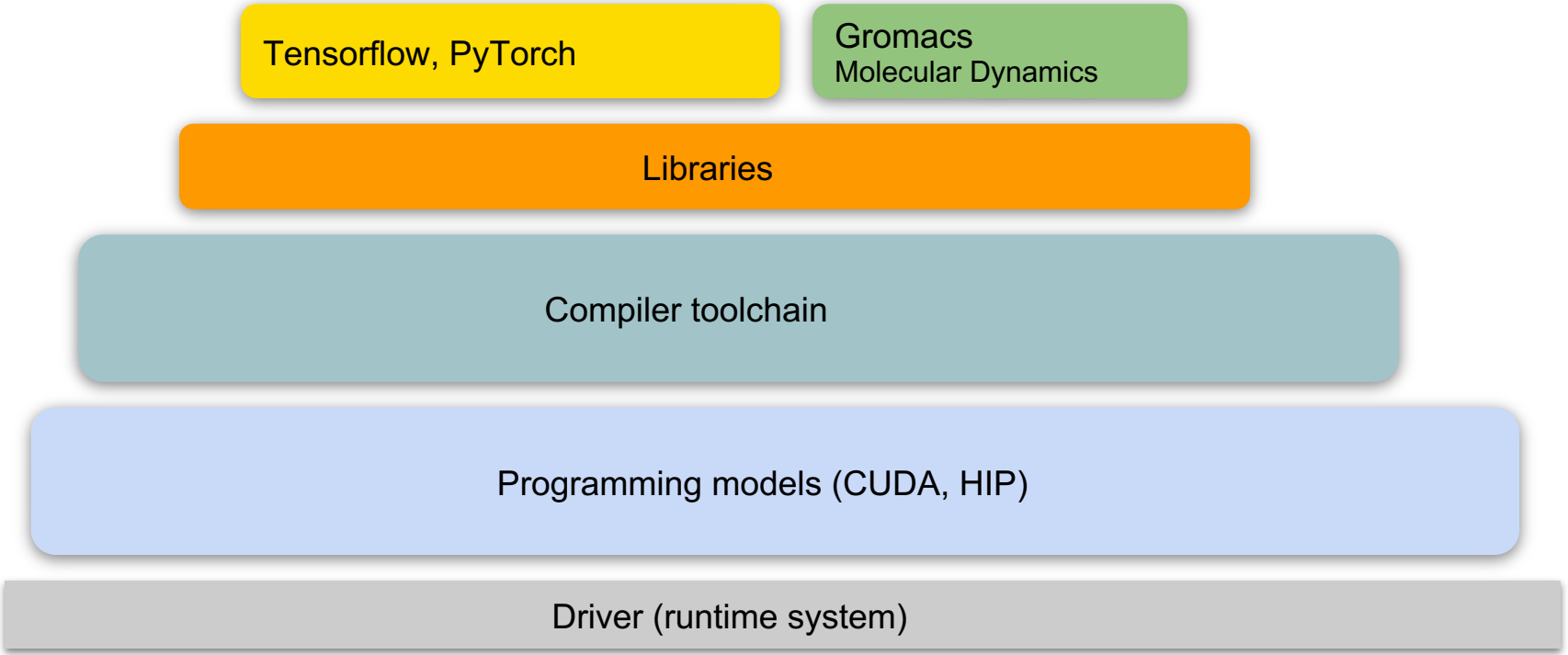
Thread Batching: Grids and Blocks

- Kernel executed as a grid of thread blocks
  - - All threads share data memory space
- Thread block is a batch of threads, can cooperate with each other by:
  - - Synchronizing their execution:
    - For hazard-free shared memory accesses
  - - Efficiently sharing data through the low latency shared memory
- Two threads from two different blocks cannot cooperate **(until CUDA8/9 and Volta)**
  - - Unless thru slow global memory
- Threads and blocks have IDs



GPU programming model




Tensorflow, PyTorch
Gromacs
Molecular Dynamics
Libraries
Compiler toolchain
Programming models (CUDA, HIP)
Driver (runtime system)

Novel / Unconventional accelerated computing paradigm

- Quantum Technology (Qubits)
- Brain-inspired computing (Neuromorphic paradigm)
- Mixed precision algorithm (floating point + Integer precision + bfloat + etc)
- Analog computing
- Specialized computing (Google TPUs, SambaNova, Cerebras, Groq, Graphcore, Habana AI, Intel Loihi, IBM Analog systems,)
- Application design (AI + HPC)
- Edge + cloud + HPC : Computing continuum

- **All images available in the presentation are present in the public domain**

Thank you