COMPUTER ARCHITECTURES AND TECHNOLOGIES TOWARDS EXASCALE

Dirk Pleiter | Indo-German Workshop on Computational Mathematics, Bangalore | 04.12.2019

JÜLICH Forschungszentrum

Disclaimer

[wikimedia]

- Long-term technology trends provide guidance
- Budget constraints can change
- (Non-HPC) Market trends are of critical importance and can change

Outline

INTRODUCTION

What Is Exascale Computing?

Answer 1: 1 EFlop/s during execution of the High-Performance Linpack benchmark

- Top500 continues to be important metric for governments and funding agencies

Answer 2: HPC infrastructure allow to address new science and engineering challenges providing 10-100x more performance compared to today's systems

- Focus on purpose of supercomputers: enable science and engineering
- No performance metric fits all
- Monolithic supercomputers may in the future become less relevant

Canonical Exascale Computing Challenges

Reducing power requirements

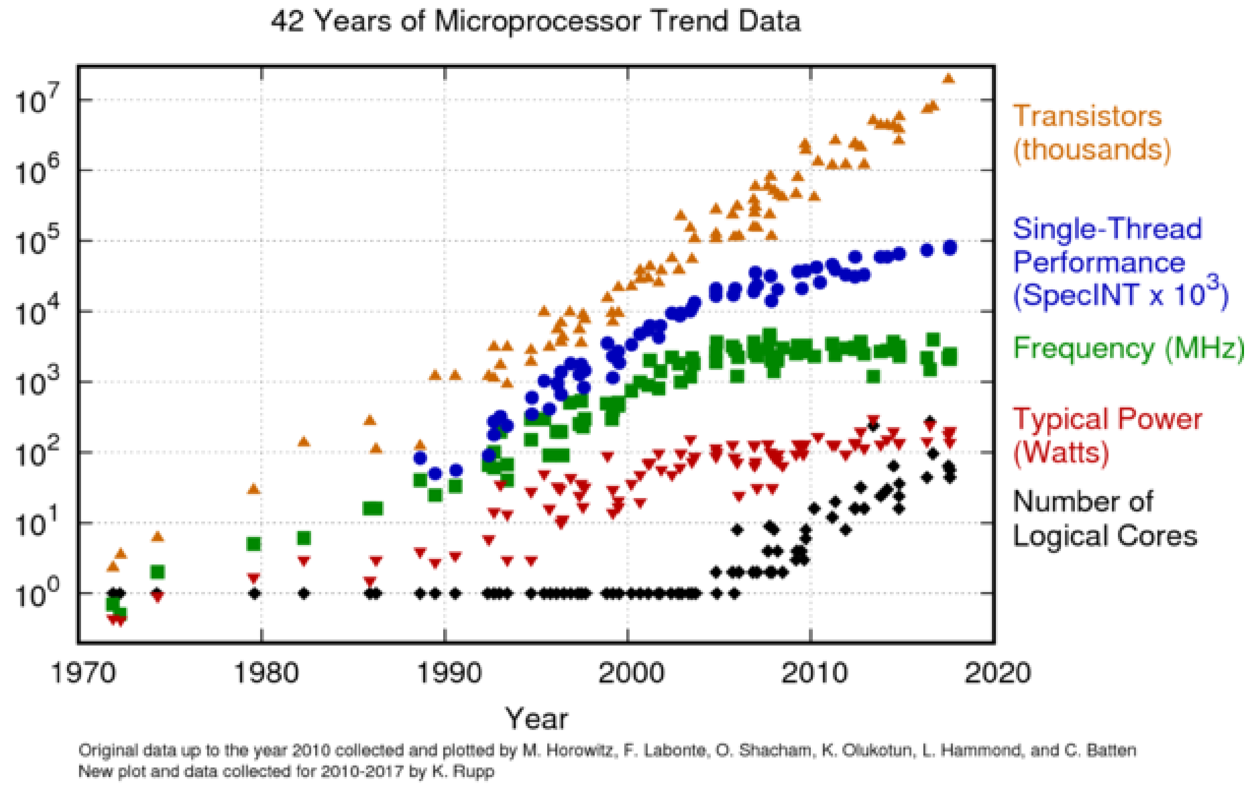
- Few sites can afford >10-15 MW
- Limits mainly dictated by budget constraints

Exploiting massive parallelism

- More computational performance
  → more parallelism, scalability challenge
  → **need for mathematical algorithms**

Maintaining a "balanced system"

- Improved data transport capabilities
- Flops are cheap, but data transport is expensive



Coping with run-time errors

- More components → higher risk of system failures → **need for robust mathematical methods**

JÜLICH Forschungszentrum

Exascale Science Cases

E. Lindahl, S. Ryan et al., “The Scientific Case for Computing in Europe 2018-2026”, October 2018

Fundamental sciences

- Astrophysics, cosmology
- Particle physics

Climate, weather and earth sciences

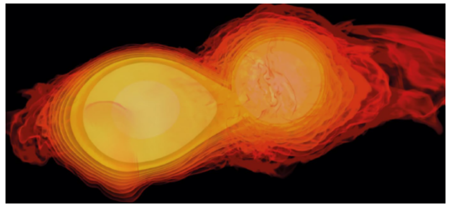
- Understanding and Predicting a Changing Climate
- Accurate Weather Forecasting and Meteorology

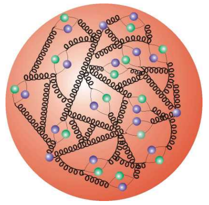
Materials science and energy research

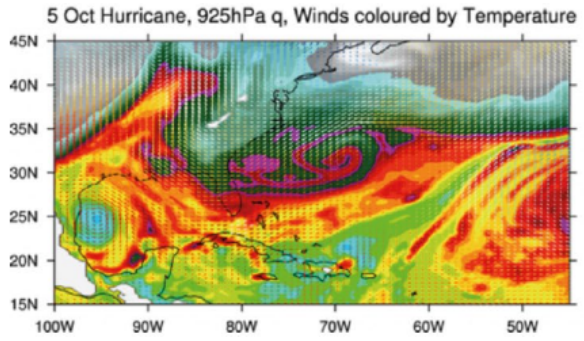
- Material properties from atomic and electronic structure simulations
- Improving lower-CO2-impact energy source

Life sciences and health

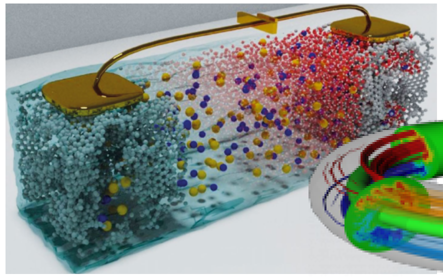
- Molecular life science and structural biology
- Neuroscience

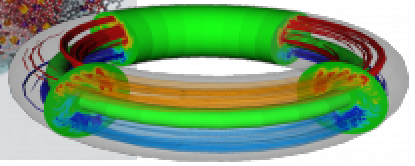

[Luciano Rezzolla]

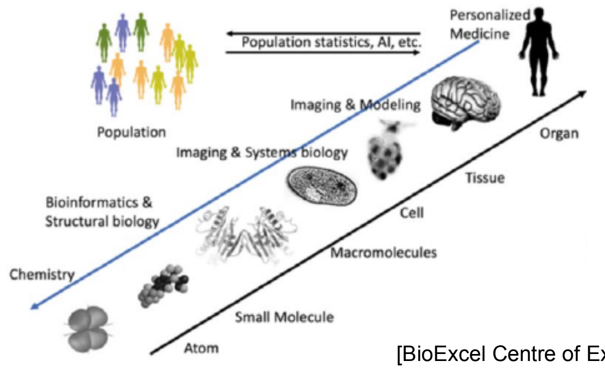




[Luigi Vidale/University of Reading]


[Mathieu Salanne, EPFL]


[J. Bigot et al., CEA]



[BioExcel Centre of Excellence]

Exascale Programs

US

- 3 exascale systems through CORAL2 procurement
- ECP project: application development, software technology, integration

Japan

- 1 exascale system at RIKEN
- Application-driven co-design of hardware

Europe

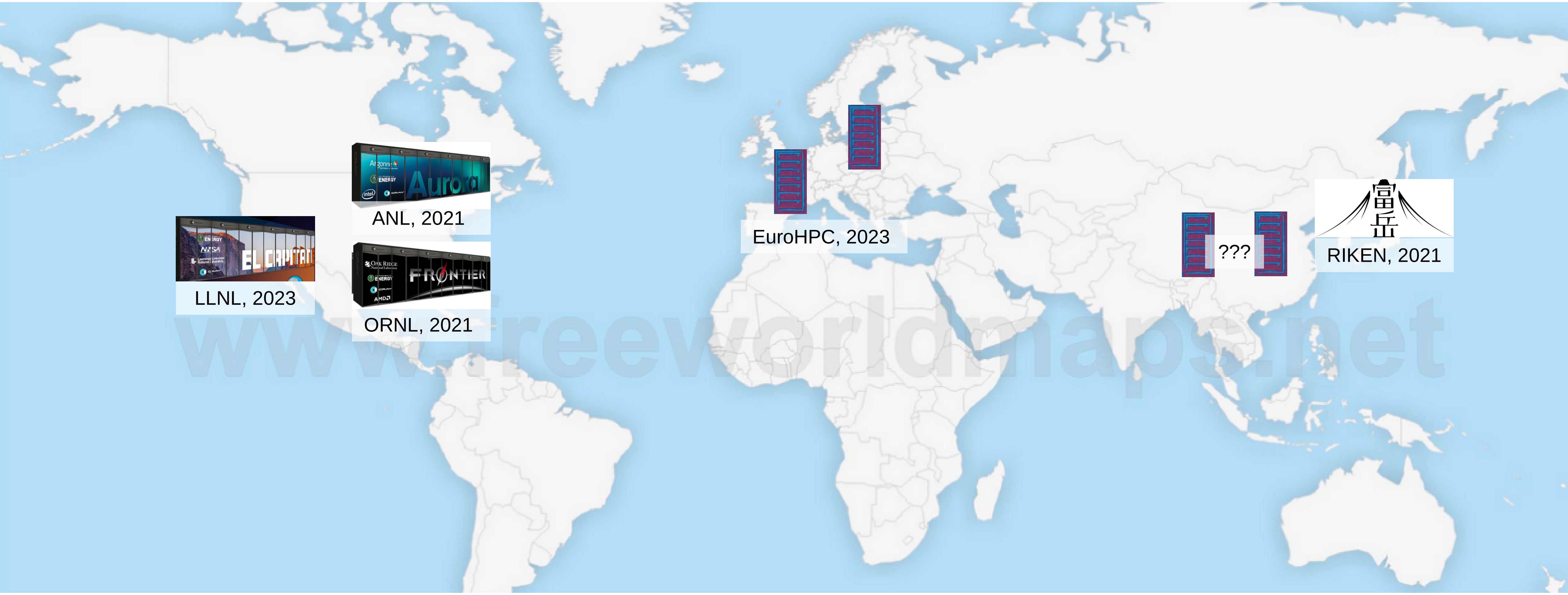
- 2 exascale systems through Joint Undertaking EuroHPC
- Research and innovation program, Centres of Excellence

China

- Details on exascale system deployments less clear
- Research program addressing development of applications and HPC environment

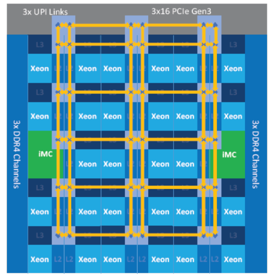
JÜLICH Forschungszentrum

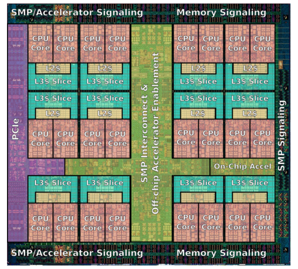
Exascale Systems Planning



EXASCALE HARDWARE TECHNOLOGIES

Diversifying Landscape of CPUs for HPC




SMP/Accelerator Signaling
Memory Signaling
PCIe
SMP Interconnect & Off-chip Accelerator Enablement
SMP Signaling
CPU Core
L3 Slice
On Chip Accel

Intel Xeon

- High compute capabilities, relatively low memory bandwidth
- Dominating market position

AMD EPYC

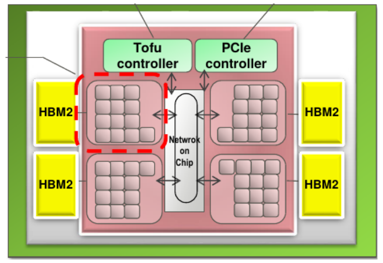
- High compute capabilities, slightly better memory bandwidth
- Emerging market position

IBM POWER9

- Low compute capabilities, relatively good data transport capabilities
- Used for current Top500 #1 and #2, lacking HPC market uptake

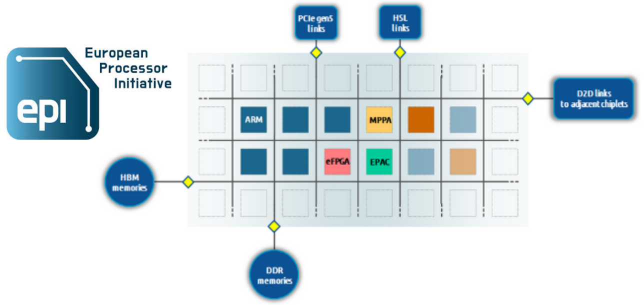
Marvell ThunderX2

- Moderate compute capabilities, relatively good memory bandwith
- Still low overall market uptake



Fujitsu A64FX

- High compute capability and memory bandwidth
- Used for Japan’s exascale system



Upcoming: European Processor Initiative

Fujitsu A64FX Features

[T. Yoshida, 2018]

Arm-based processor architecture

- Previous processor based on SPARC

High floating-point operations throughput

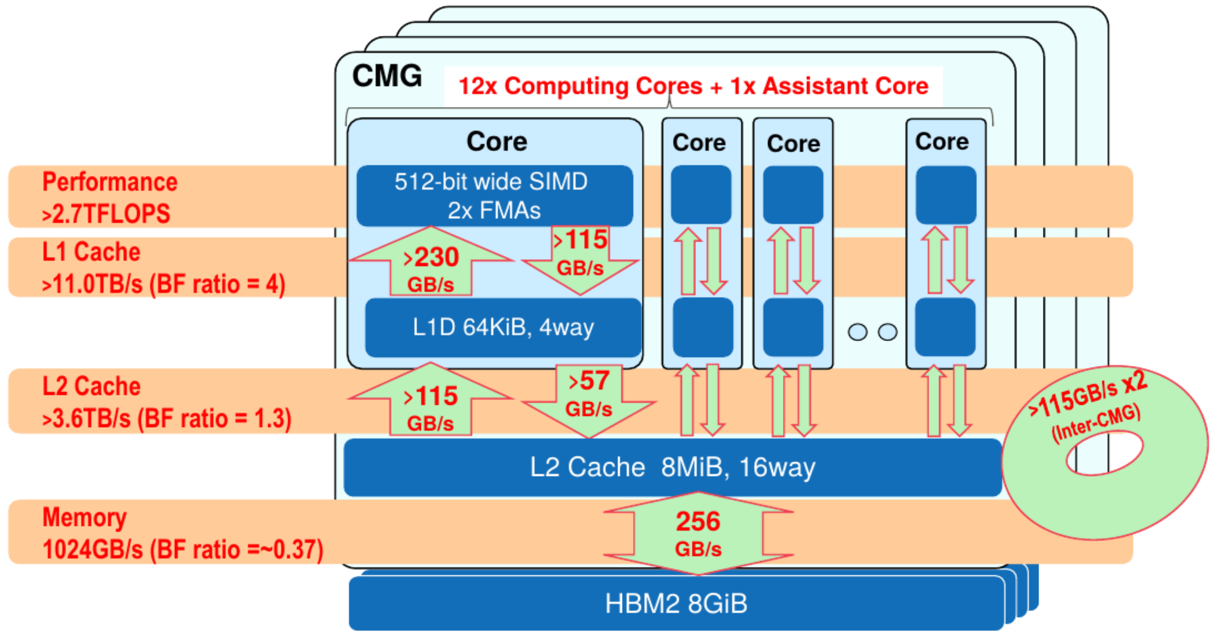
- 2x 512-bit vector pipelines per core
- First processor to implement Arm's new SVE Instruction Set Architecture

High-memory bandwidth

- First processor to use HBM2
- But: relatively low memory capacity

Outstanding power efficiency

- 16.9 GFlop/s/W



Number of cores	48 + x
Clock frequency	2 GHz
Throughput of FP64	3 TFlop/s
Memory bandwidth	1 TByte/s
Memory capacity	32 GiByte

JÜLICH Forschungszentrum

CPU: Core vs. Vector Parallelism

Strategies for increasing CPU-level parallelism of floating-point operations

- Increase number of cores
- Increase width of SIMD/vector instruction

	Increase core count	Increased SIMD/vector width
Pro	More flexible thread-parallelism	Simplified hardware architecture
Con	Replication of instruction front-end	Increased power-consumption per node; Vector/SIMD parallelism more difficult to exploit

→ Need for trade-off decisions

New vector ISA: Arm Scalable Vector Extension (SVE)

Key feature: Vector length agnostic

- Vector length not defined at compile time
- Multiple lengths supported by ISA:
  128, 256, …, 2048 bit

```
void daxpy(double a, double *restrict x,
           double *restrict y, int n)
{
  int i;
  for (i = 0; i < n; i++)
    y[i] += a * x[i];
}
```

Arm C/C++/Fortran Compiler version 19.3

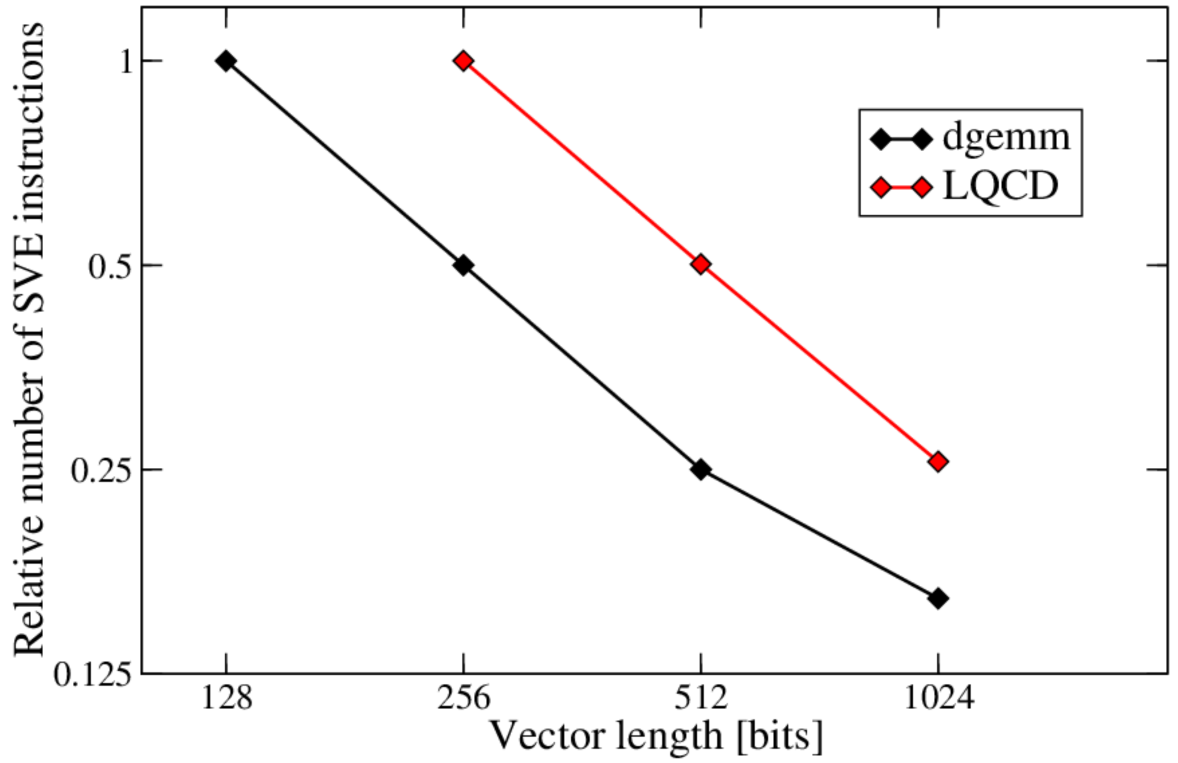
```
        mov      x8, xzr
        mov      z0.d, d0
        whilelo p1.d, xzr, x9
        ptrue   p0.d
.LBB0_2:
        Ld1d {z1.d}, p1/z, [x0, x8, lsl #3]
        ld1d {z2.d}, p1/z, [x1, x8, lsl #3]
        fmad z1.d, p0/m, z0.d, z2.d
        st1d {z1.d}, p1, [x1, x8, lsl #3]
        incd x8
        whilelo p1.d, x8, x9
        b.mi .LBB0_2
.LBB0_3:
        ret
```

Required hardware support for VLA

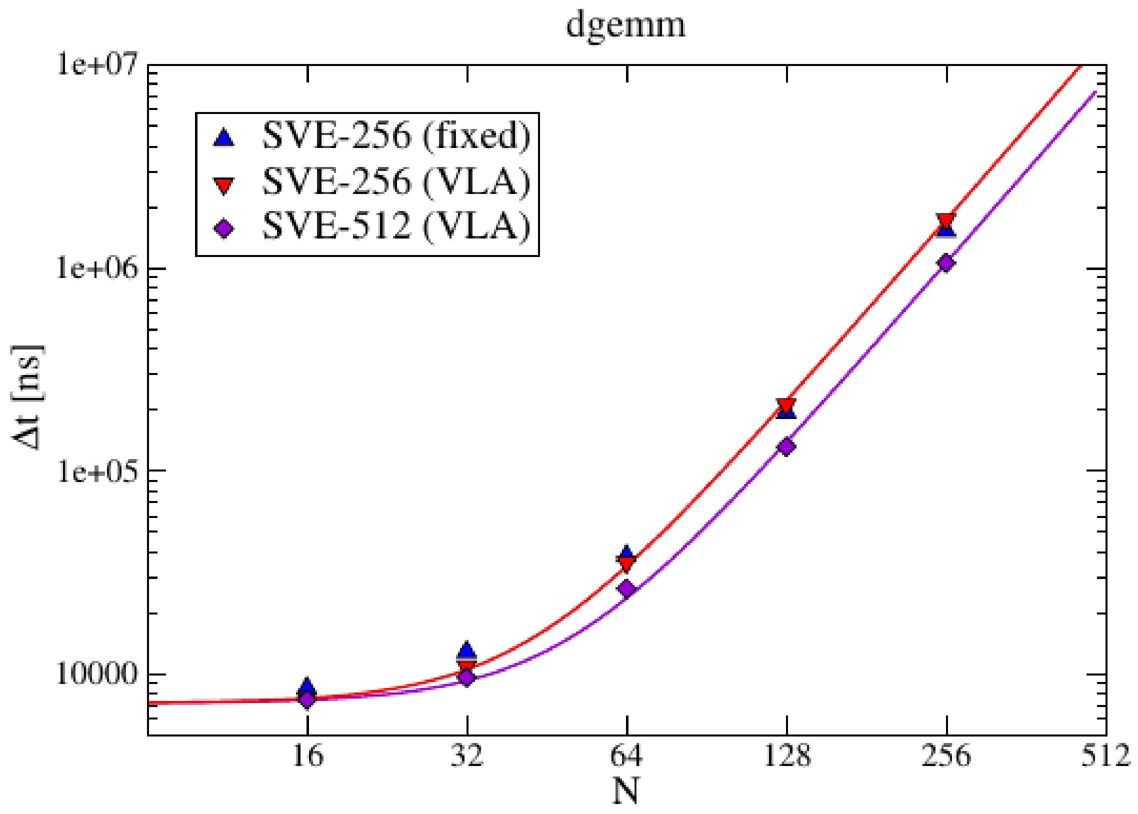
- Update of predication registers
- Update of loop counters

Exploring Arm’s Scalable Vector Extension

Change of number of SVE instructions as a function of the vector length



Gem5 simulation results for 256 bits and 512 bits



[B. Brank, N. Ho, S. Nasyr, DP, A. Portero, 2019]

Selected Processing Device Trends

Reduced precision and new floating-point formats

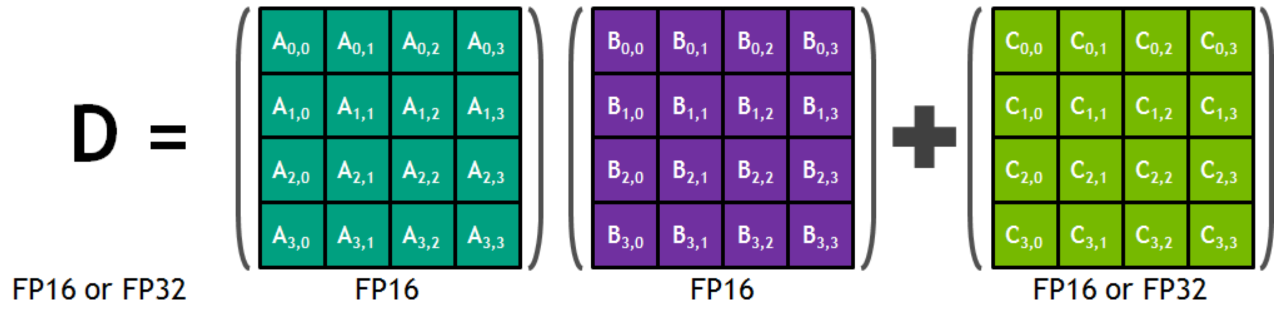
- Instructions for FP16 arithmetics became the standard
- Increasing support for bfloat16 floating-point format
- Other formats under investigation (e.g. posit)

  → **Need for reduced/mixed-precision algorithms**

Tensor instructions

- E.g. instructions introduced NVIDIA's Volta GPU Architecture

  → **Need for mathematical algorithms benefiting from such instructions**


$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

[NVIDIA, 2017]

Power Efficiency: State of Affair

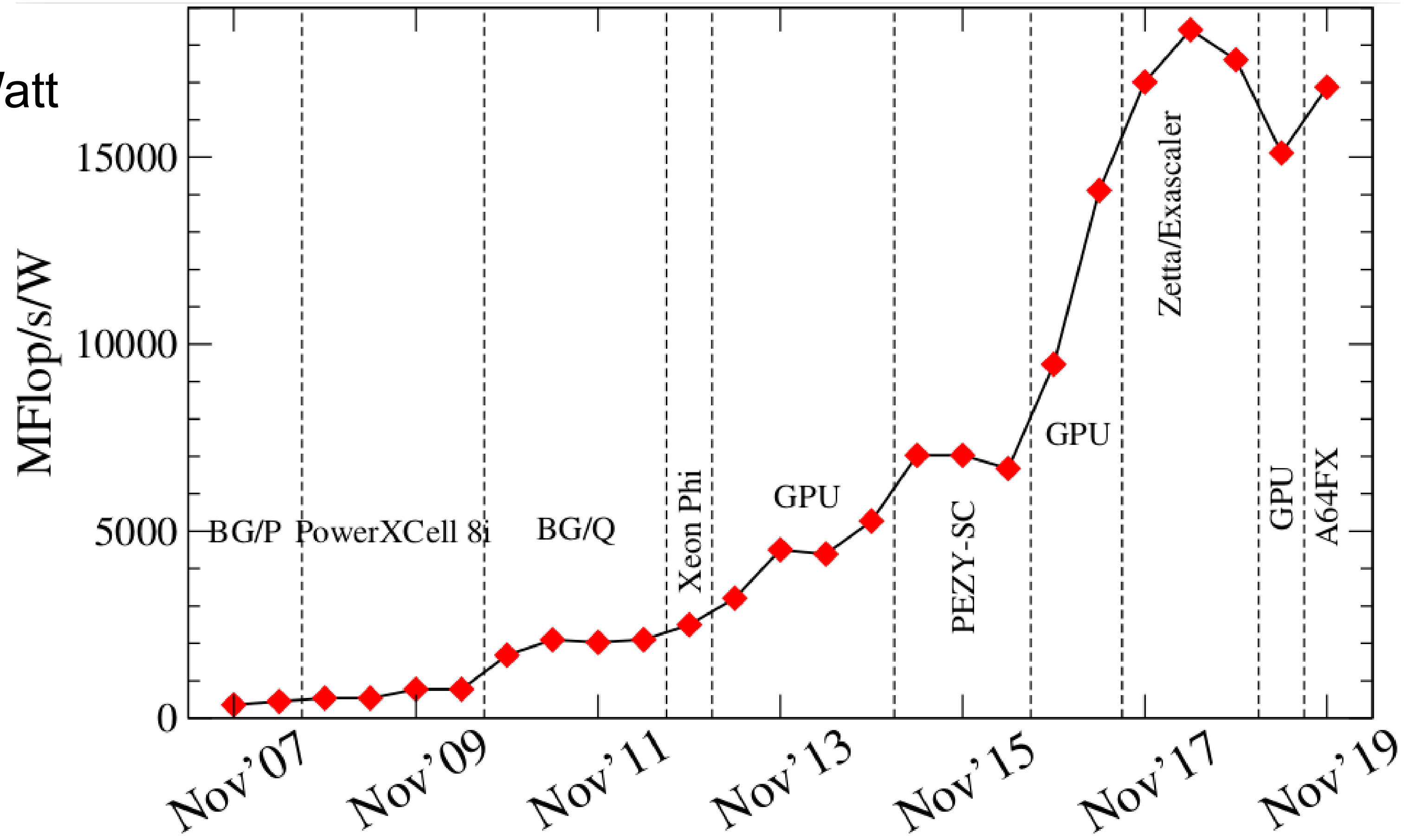
Exascale targets

- US labs: 30 MWatt → 33 GFlop/s/Watt

Green500 list (Nov'19)

- #1: A64FX processor with 16.9 GFlop/s/Watt
- Top positions dominated by compute accelerators
- Best Xeon-based system at #37: 5.8 GFlop/s/Watt

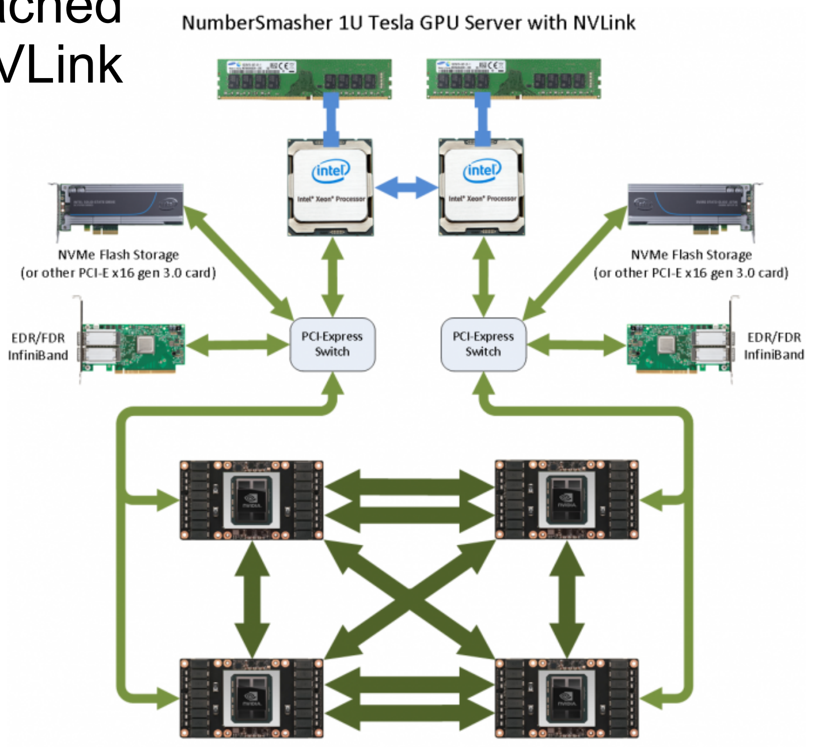
→ Use of compute accelerators crucial



Compute Accelerators for HPC: Today and Near Future

Today: NVIDIA GPUs

- Group of 3-4 Tesla GPUs interconnected via NVLink
  - Up to ~20-30 TFlop/s per group
- 1-2 CPUs attached via PCIe or NVLink


NumberSmasher 1U Tesla GPU Server with NVLink
NVMe Flash Storage
(or other PCI-E x16 gen 3.0 card)
NVMe Flash Storage
(or other PCI-E x16 gen 3.0 card)
EDR/FDR
InfiniBand
PCI-Express
Switch
PCI-Express
Switch
EDR/FDR
InfiniBand

Future competitors: AMD and Intel GPUs

- Intel Xe GPU set for Aurora at ANL
- AMD GPUs set for Frontier at ORNL and El Capitan at LLNL
  - Similar integration approach as for NVIDIA today

Common future trend: Improve integration of discrete accelerator and CPU through coherent I/O interfaces

JÜLICH Forschungszentrum

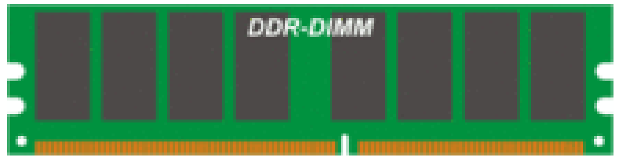
Memory Technologies (1/2)

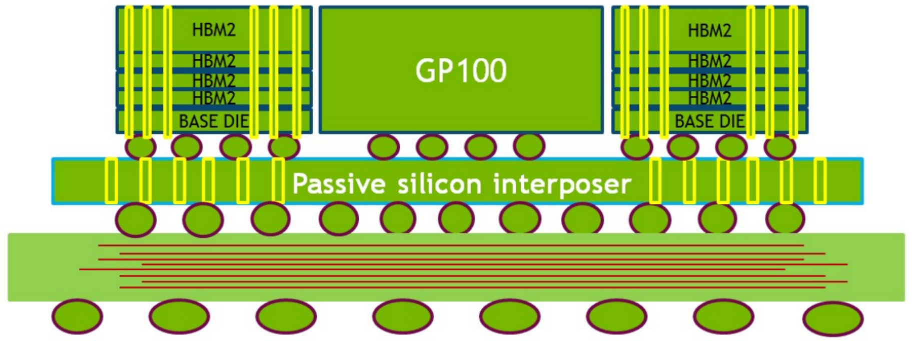
Desirable memory performance features

- Large memory capacity C_{mem}
- High memory bandwidth B_{mem}

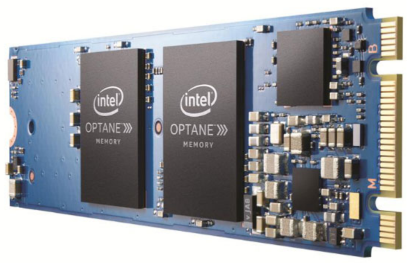
Main types of memory

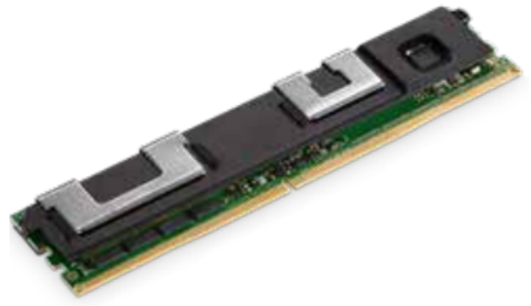
- DDRx
  - DDR4: Up to ~200 GByte/s per CPU using 8 channels
  - DDR5: Aims on doubling performance
  - Capacity of O(100) GiByte
- High-Bandwidth Memory (HBM)
  - Today: ~250 GByte/s per stack
  - Near future: ~400 GByte/s per stack
  - Capacity of O(10) GiByte
    - Capacity per stack announced to double
  - More power efficient (lower data rate per line, shorter traces)





Memory Technologies (2/2)





Main types of memory (cont.)

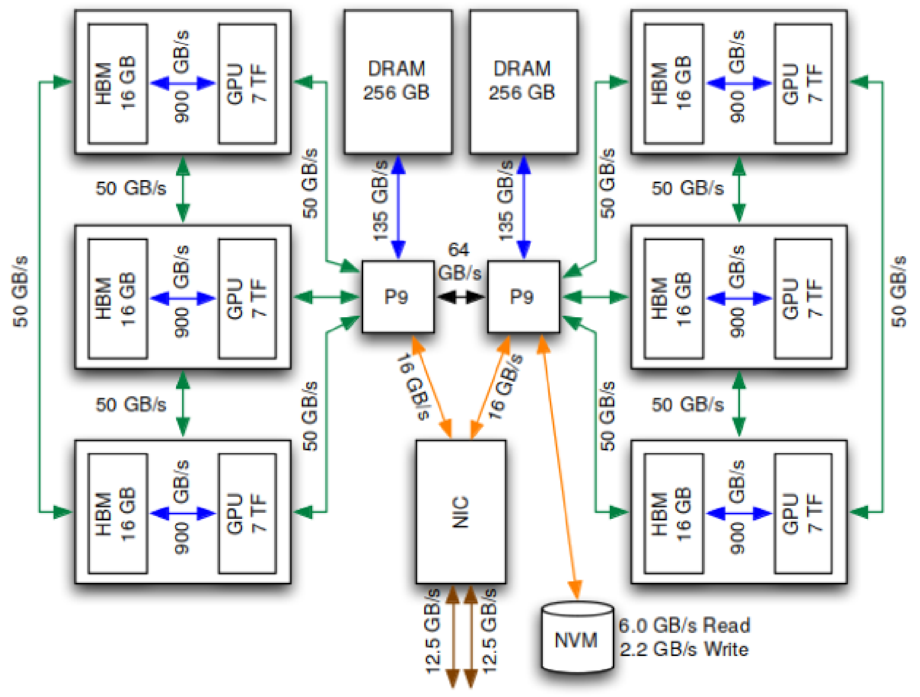
- Non-volatile memory
  - Technologies: NAND Flash, 3D-Xpoint
  - Attachment via I/O interface or memory bus
  - Large variety of software interfaces

Significant differences in technology for $\Delta\tau = C_{mem} / B_{mem}$

NVIDIA V100 GPU	HBM2	$\Delta\tau \approx$ 20-40 ms
JUWELS compute node	DDR4	$\Delta\tau \approx$ 0.4 s
Intel Optane DC	Memory attached 3D XPoint	$\Delta\tau$ = O(35-140s)
Intel DC P4511 2 TByte	PCIe attached NAND Flash	$\Delta\tau \approx$ 1300 s

JÜLICH Forschungszentrum

Deeper Memory Hierarchies

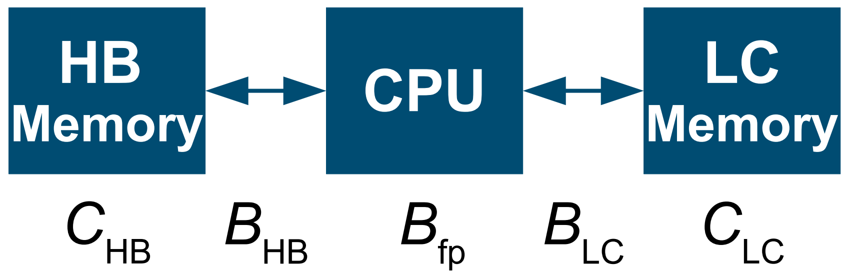


Rationale

- High-bandwidth memory tier based on technologies like HBM (maximise performance/€)
- Large-capacity memory tier based on technologies like DDR4 or non-volatile memory (maximise capacity/€)

Example: Summit node architecture

- HBM2 memory attached to NVIDIA V100
- DDR4 + Flash memory attached to IBM POWER9



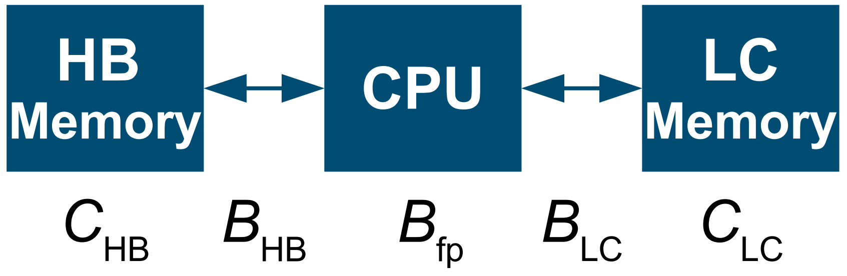
Need for trade-off decisions

- Bandwidth and capacity ratio high-bandwidth vs. large-capacity memory tier
- Hardware support for data transport between both memory tiers

JÜLICH Forschungszentrum

Deeper Memory Hierarchies: Exploitation Scenarios

Separation of hot and cold data objects



- Hot data used in performance critical regions (kernels) is accessible from High-Bandwidth Memory tier
- C_{HB} must be large enough to hold hot data objects
- C_{LC} must be large enough to hold hot data objects
- B_{HB} should be balanced for given B_{fp}

Data staging during kernel execution scenario (double buffering)

- B_{LC} must be large enough to allow data staging and results migration during kernel execution, i.e. $\Delta t_{\mathrm{transport}} \leq \Delta t_{\mathrm{compute}}$

Data staging outside of kernel execution scenario

- B_{LC} must be large enough to allow data staging and results migration during kernel execution, i.e. $\Delta t_{\mathrm{transport}} < \varepsilon\, \Delta t_{\mathrm{compute}}$

→ Need for mathematical algorithms exploiting data locality

Brief Outlook on Network Technologies

Main high-end technologies

- Infiniband
- Slingshot

Linkspeed

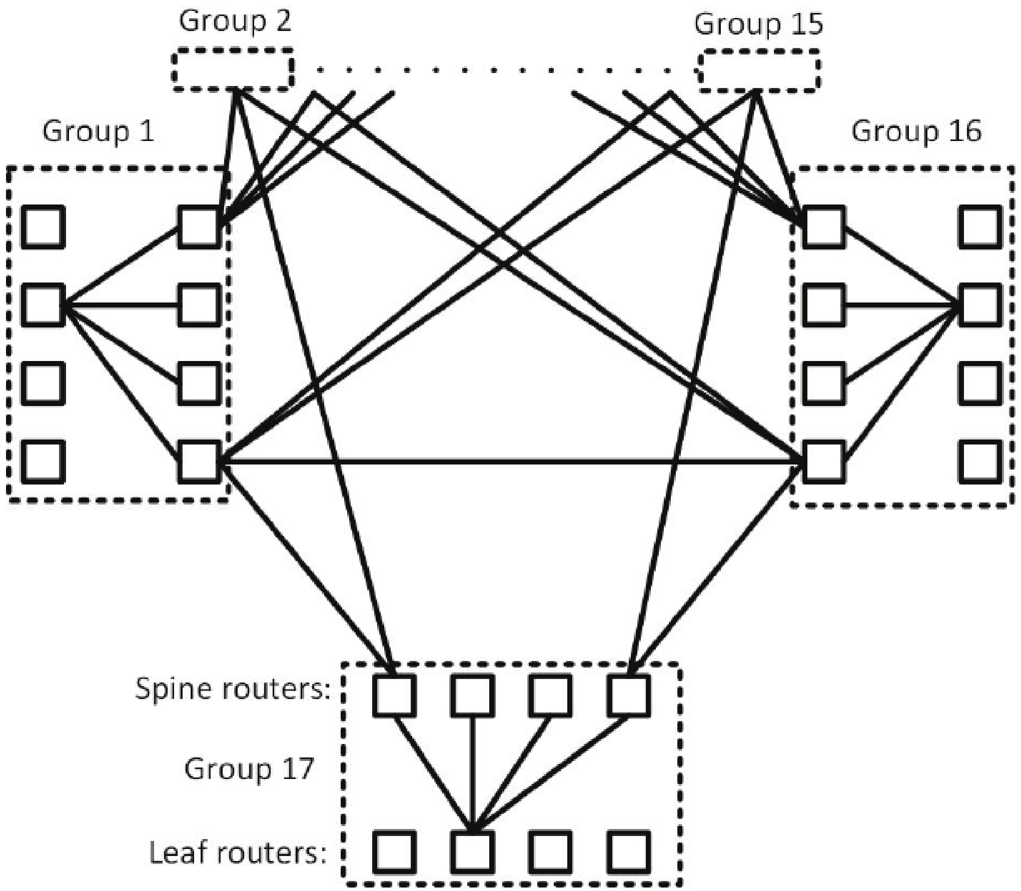
- 200 Gbps during the next few years

Topologies

- Dragonfly topology likely prime choice for largest systems for cost reasons
  - Cray Slingshot
  - Mellanox Dragonfly+
- Reduced bi-section bandwidth

→ **Need for communication-avoiding algorithms**

[Alexander Shpiner et al., 2017]



Known Exascale Architecture Swim Lanes

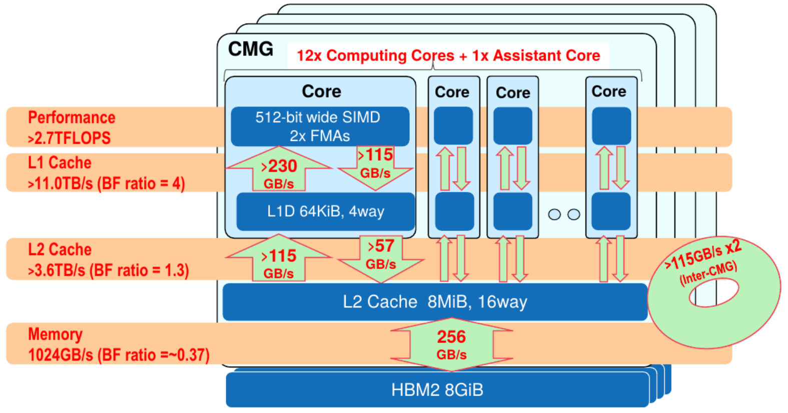
GPU-based: Frontier, Aurora, El Capitan

- Node architecture
  - 1-2 CPUs
  - Multiple GPUs
  - $\gg$40 TFlop/s per node
    $\rightarrow$ O(10,000) nodes
  - $B_{fp}/B_{mem} \gg 5$?, B_{mem} = ~512-1024 GiByte?
- Dragonfly-type network

Commonality: Use of high-bandwidth memory technologies

CPU-based: Fugaku

- Node architecture
  - 1 CPU
  - >2.7 TFlop/s per node
  - $B_{fp}/B_{mem} \approx 3$, B_{mem} = 32 GiByte
- 6-dimensional torus network


CMG
12x Computing Cores + 1x Assistant Core
Core
Core
Core
Core
Performance >2.7TFLOPS
512-bit wide SIMD 2x FMAs
L1 Cache >11.0TB/s (BF ratio = 4)
>230 GB/s
>115 GB/s
L1D 64KiB, 4way
L2 Cache >3.6TB/s (BF ratio = 1.3)
>115 GB/s
>57 GB/s
>115GB/s x2 (Inter-CMG)
L2 Cache 8MiB, 16way
Memory 1024GB/s (BF ratio =~0.37)
256 GB/s
HBM2 8GiB

[T. Yoshida, 2018]

Hierarchical Storage Architectures

Increasing diversity of storage device technologies used for HPC

Devices currently (or soon) in use at JSC:

Technology	Capacity [TiByte]	Bandwidth [GByte/s]	C [s]
Intel Optane DC	0.125-0.5	~4 (r/w mix)	35-140
Intel Optane SSD	0.4-1.5	~2	210-830
Toshiba CM5 SSD	0.8-6.4	~3	290-2300
Lenovo 01DC407	1.2	~0.30	4,500
Seagate Tatsu series	10	~0.17	65,000

Upcoming HPC challenges and trends

- Scale-up to $\gg$O(10) TByte/s bandwidth
  - Mandates storage architectures becoming (more) hierarchical
- Enable near-node storage
  - Drop separation of compute and storage cluster
- Mitigate metadata performance limitations
  - Promote APIs beyond POSIX

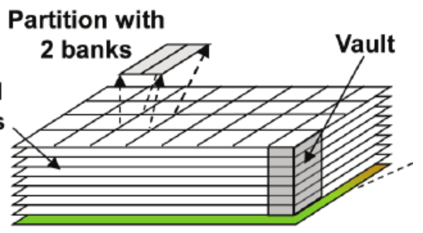
Emerging Hardware Architectures and Technologies

Data-flow architectures

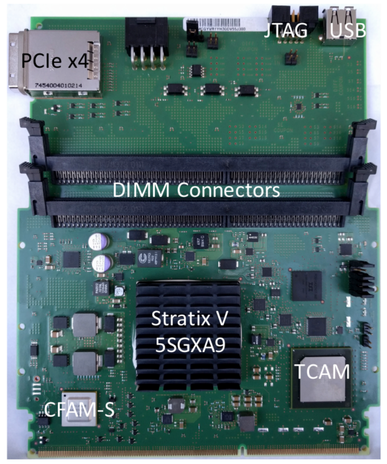
- Promising approach for improving energy efficiency
- Today based on FPGA, in future other options might emerge

In-memory compute / Near-Memory Acceleration

- Rationale: Reduce energy by avoiding data movement by moving computation to data
- Different implementations
  - Compute in stacked memory, e.g. AMC
  - Compute in memory buffer
  - Compute in PCM
- Challenge: Programming model



[R. Nair et al., 2015]



[J.v. Lunteren, 2016]

PROGRAMMING

Programming Models

Programming models and run-time systems critical for coping with increasing hardware complexity, but choice increases

- MPI, OpenMP, OpenACC, TBB, PGAS, GPI, StarPU, OmpSs, CUDA, ROCm, Sycl, HPX, Kokkos, Legion, RAJA, ...

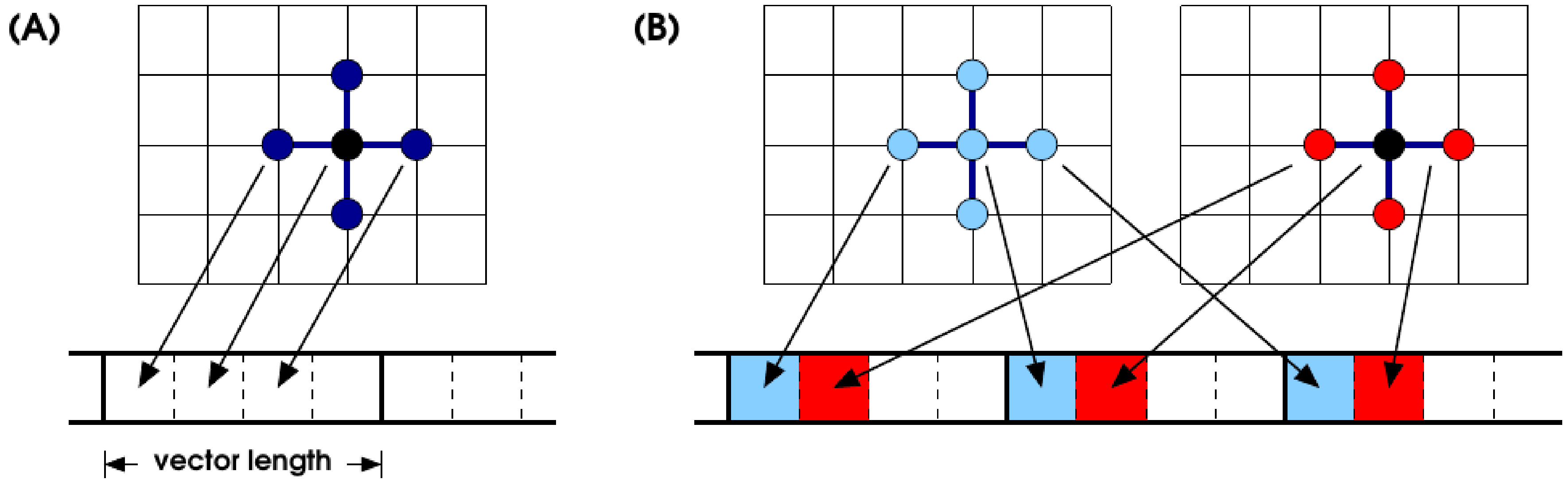
"MPI + X" likely dominates

- No one-fits-all parallel programming model
  - Example: Upcoming diversity of GPUs

Critical short-coming: Most programming models lack data transformation and data orchestration capabilities

Data Transformation: Vectorization of Stencil Apps

Data layout options:



Option (B) removes need for vector shuffling

- But: Data layout in case of (B) depends on vector length
- Need to flexible when supporting different processor architectures

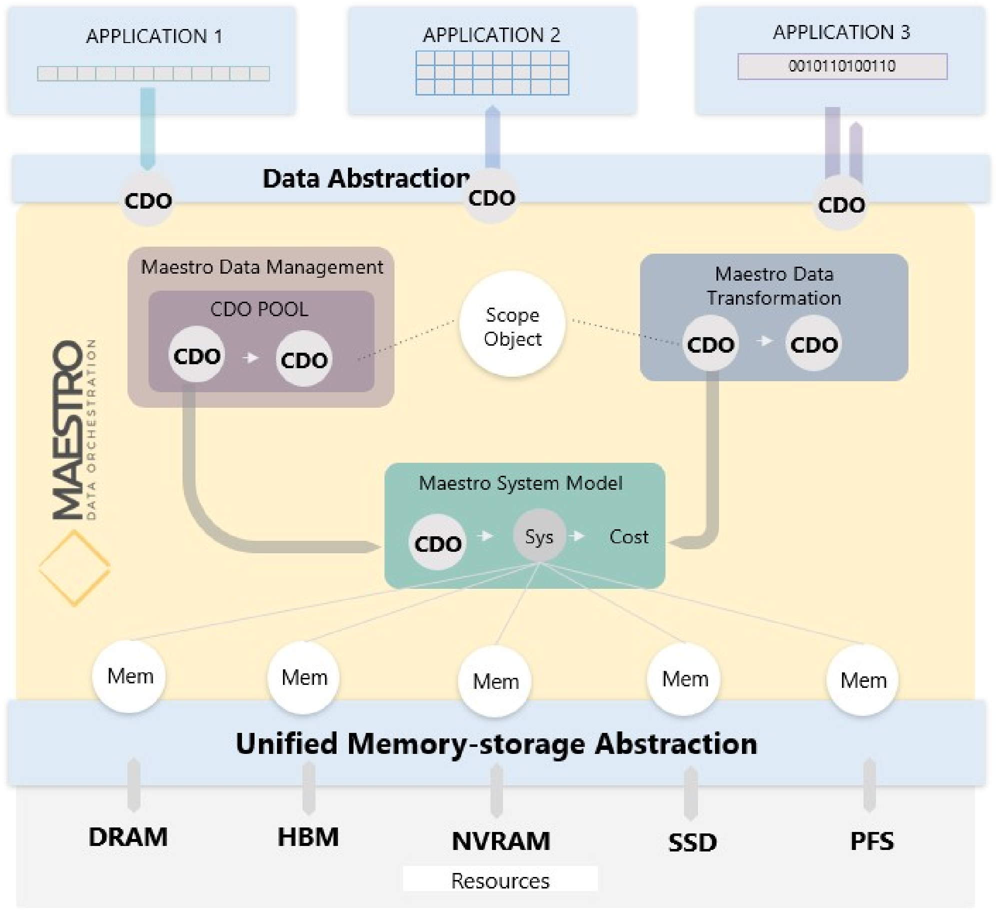
Data Orchestration: Maestro's Approach

Solution concept

- Systems software middleware that intelligently manages data placement and movement
- Object-based approach to encapsulate data with application and Maestro related metadata
- Data movement decision based on workflow annotations and real-time I/O monitoring
  → **Need for mathematical solutions to optimisation problems**

Implementation

- Data pool managed by middleware
- Give/take object semantics



Modernising Application Design

[Schulthess, 2015]

Programming means

- Specifying computation
- Managing computer resources

Opportunity for separation of concern

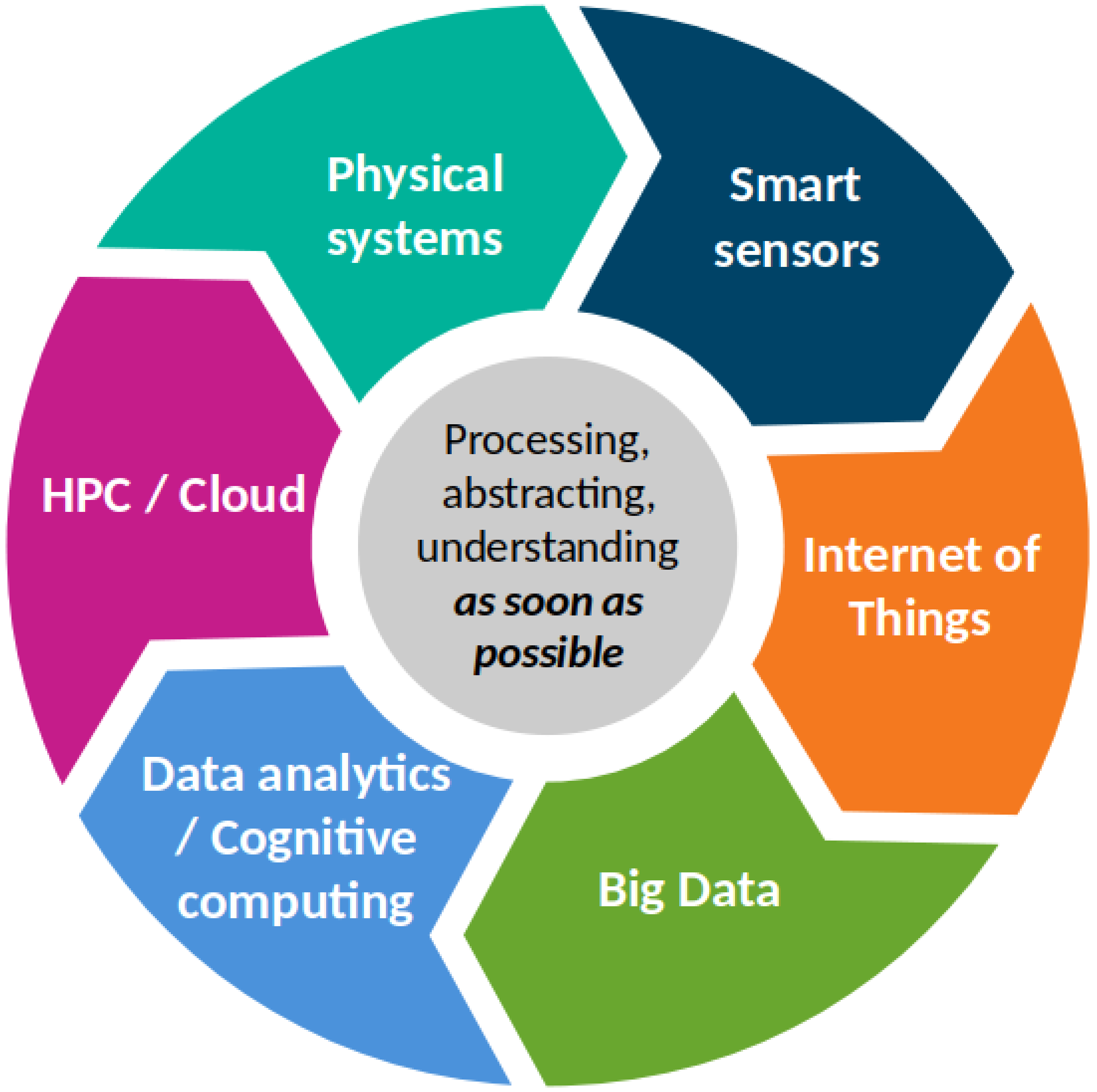
- Front-end: Computation specified by domain scientist using high-level languages
- Back-end: Management of computer resources by HPC experts

Other benefits

- Descriptive frontend could link to different back-ends → performance portability
- Efforts for implementing back-ends could become community effort
  - Allow domain scientists compete on frontend

FUTURE HPC INFRASTRUCTURES

New HPC Usage Models Emerging


Physical systems
Smart sensors
Internet of Things
Big Data
Data analytics / Cognitive computing
HPC / Cloud
Processing, abstracting, understanding *as soon as possible*

Need for designing e-infrastructures including supercomputers

- Supercomputers cannot be designed as silos

[ETP4HPC, 2020]

Opening HPC Infrastructures: Expected Developments

Federated user management

- Few technical challenges
- Major organisational challenges

Setup clusters for deploying Cloud-type services with path to HPC world

- But: HPC will remain a protected region

Option to deploy services in HPC world

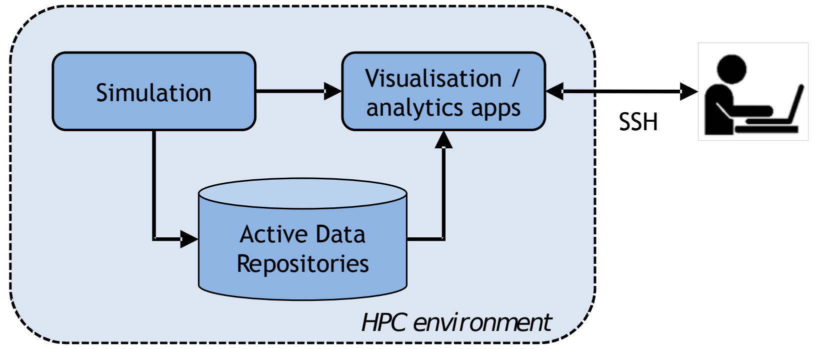
- Examples: databases, workflow schedulers

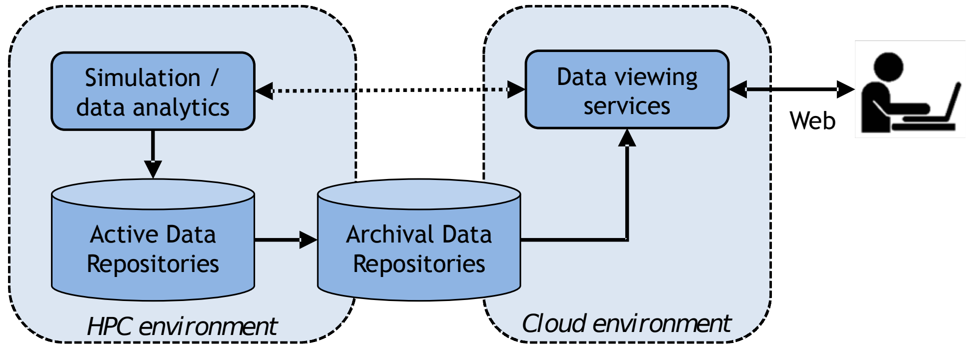
Improve on interactivity

- Support of interactive frameworks like Jupyter notebooks

Allow for new allocation models

- Depends largely on funding agencies





ROLE OF MATHEMATICAL METHODS AND ALGORITHMS

Role of Mathematical Methods and Algorithms: ETP4HPC's view

Robust methods and algorithms enabling extreme scalability

- Need for more parallelism at all levels, support of communication avoiding
- Support of reduced/mixed-precision calculations (trade performance for accuracy)

Methods for (scalable) data analytics and artificial intelligence

- Shift from a compute-centric view to a more data-centric view
- Algorithms for data discovery

Algorithms reducing energy-to-solution

- Characterisation of algorithms wrt energy requirements
- Minimisation of data movement and improved hardware utilisation

Vertical integration and validation of mathematical methods and algorithms

- Tuning of algorithm parameters for exascale problems
- Integration into diverse HPC and beyond software stacks

SUMMARY, CONCLUSIONS AND OUTLOOK

Summary and Conclusions

Paths to exascale architectures established

- Mainly based on compute accelerators
- Use of high-bandwidth memories mandatory for getting to >0.1 Ebyte/s

Major challenges for users

- Cope with heterogeneity and diversity
  - Different processor architectures (x86, Arm)
  - Variety of GPUs
- Exploit deeper memory hierarchies
- Need for increased efforts in code modernisation aiming for split of concerns between domain scientists and HPC expertss

Critical role of mathematical methods and algorithms

JÜLICH Forschungszentrum

Outlook on Supercomputers at JSC

A turbocharger for the supercomputer JUWELS

Forschungszentrum Jülich, Atos and ParTec with NVIDIA and Mellanox plan expansion of the Jülich supercomputer

Jülich, 14 November 2019 – The Jülich supercomputer JUWELS will have a big brother, a so-called booster module, as Forschungszentrum Jülich, Atos, and ParTec have agreed. The module, equipped with several thousand graphics processors, is designed for extreme computing power and artificial intelligence tasks. It is designed as a Franco-German project together with NVIDIA and Mellanox using the co-design process. With the launch of the booster in 2020, the computing power of JUWELS will be increased from currently 12 to over 70 petaflops. This is equivalent to 70 trillion computing operations per second or the power of more than 300,000 modern PCs – no computer in Europe currently calculates faster.