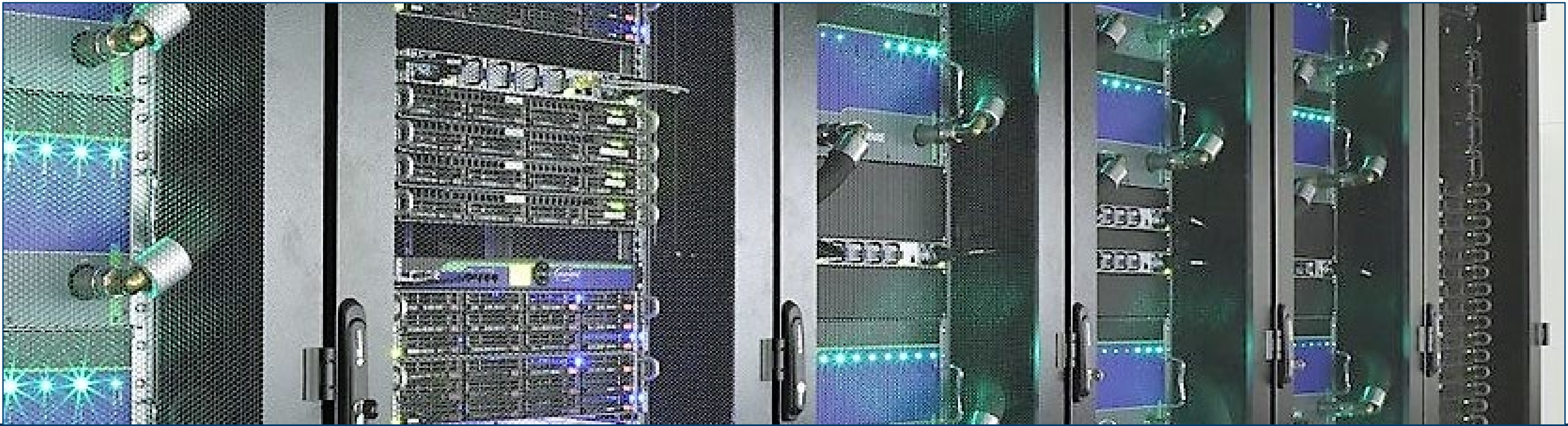


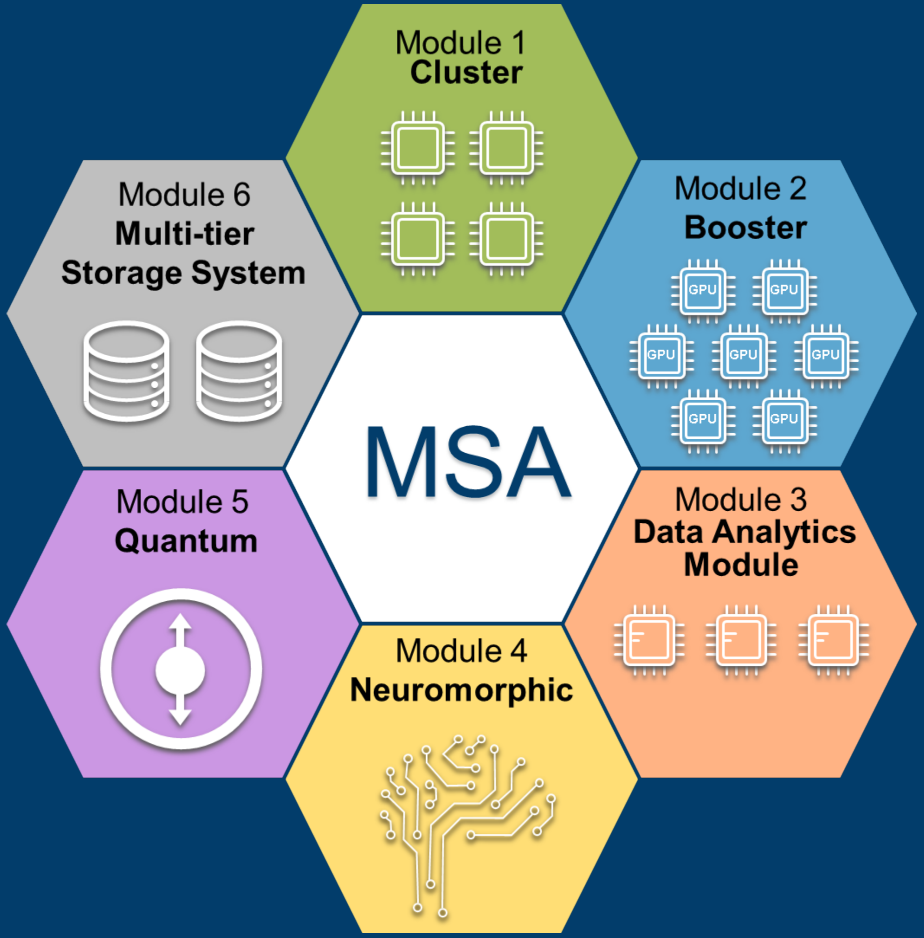
MODULAR SUPERCOMPUTING:

enabling application diversity in HPC

10.10.2023 I ECMWF Workshop I Estela Suarez (FZJ/JSC & UniBonn)

OUTLINE

- JSC and its users
- System Architecture
- Software Stack
- Application Experience
- Summary



Jülich Supercomputing Centre

Where we are

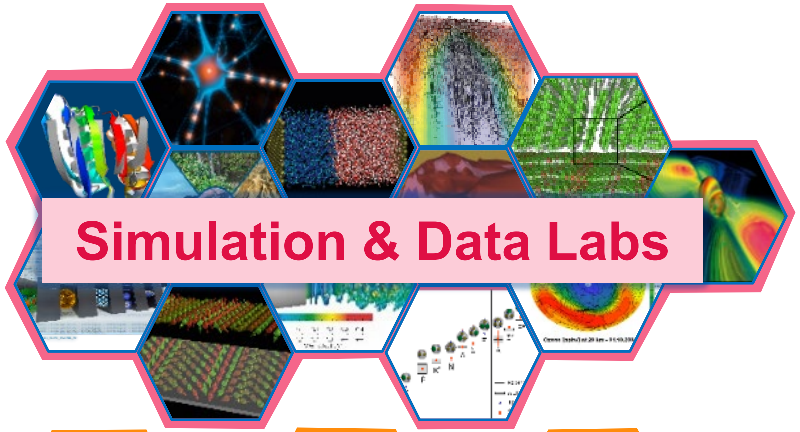
JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

Jülich Supercomputing Centre

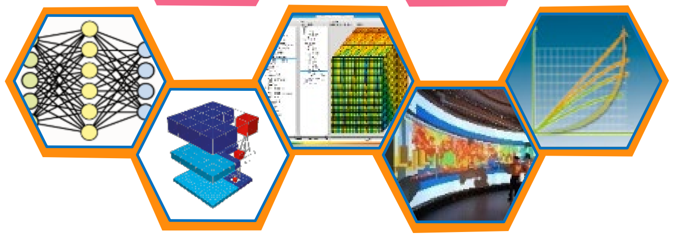
What we do („We“ are >300 people at JSC)

- **System Operation**

- **Support**



Simulation & Data Labs



Algorithms, Tools & Methods Labs

- **Research**

DEEP Projects

SEA Projects

epi

MAELSTROM

EUPEX European Pilot for Exascale

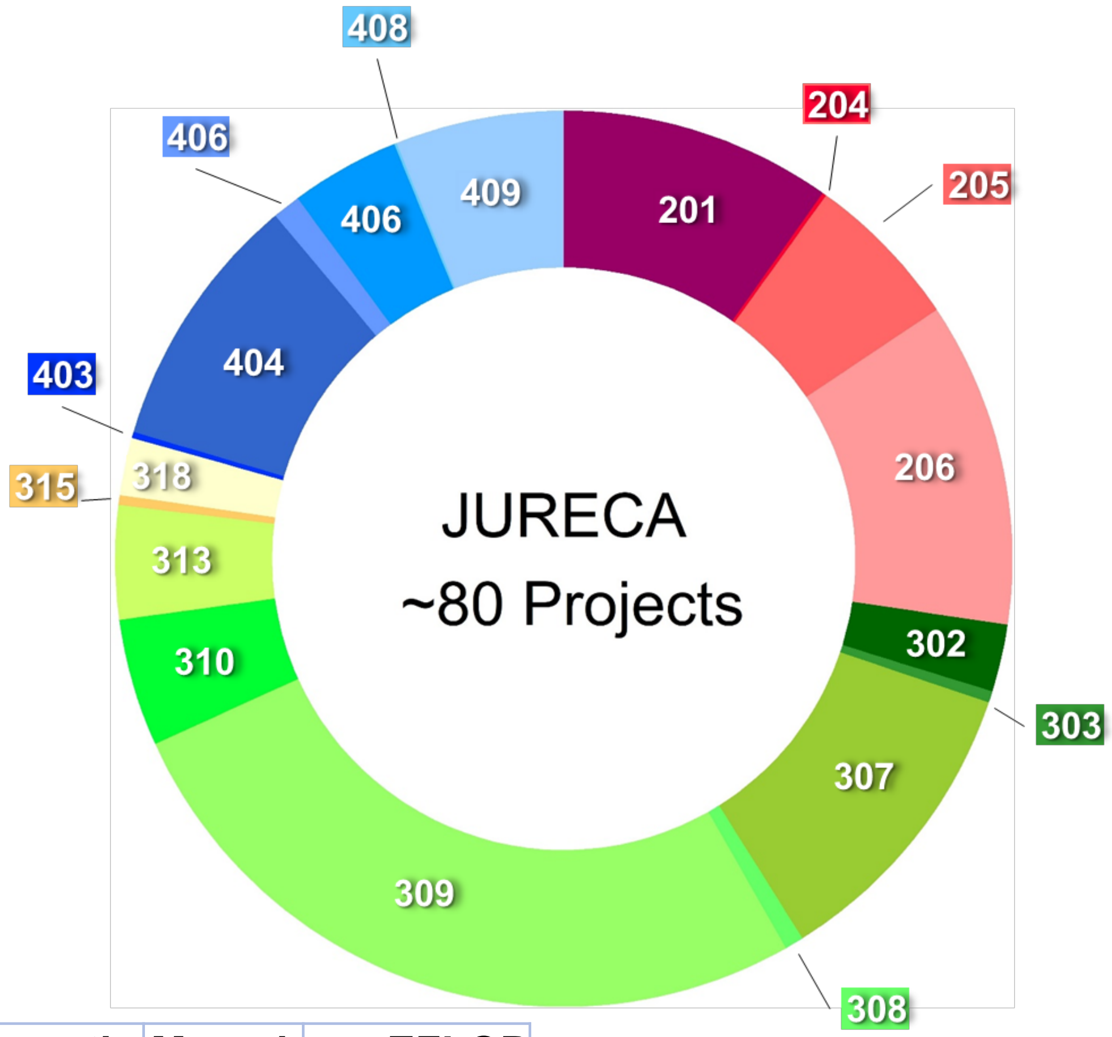
THE EUPILOT

HPCQS

...

JSC Users

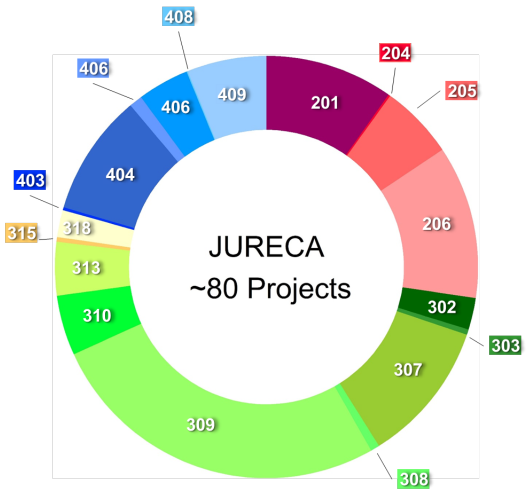
May - October 2023



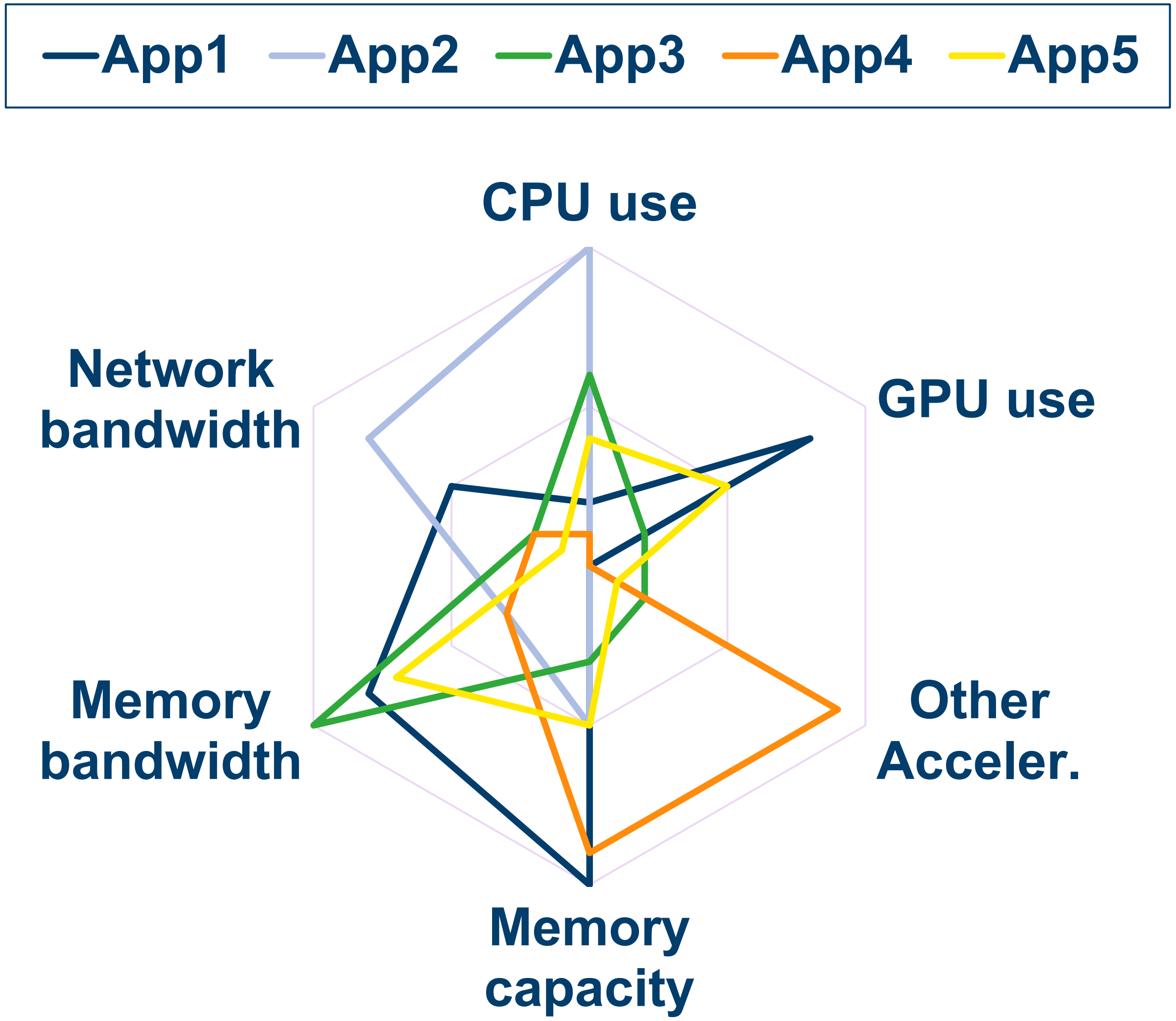
6-month	Mcoreh	EFLOP
JURECA	350	550,000

201	Basic Biological and Medical Research
204	Microbiology, Virology and Immunology
205	Medicine
206	Neurosciences
207	Agriculture, Forestry and Veterinary Medicine
302	Chemical Solid State and Surface Research
303	Physical and Theoretical Chemistry
307	Condensed Matter Physics
308	Optics, Quantum Optics and Physics of Atoms, Molecules and Plasmas
309	Particles, Nuclei and Fields
310	Statistical Physics, Soft Matter, Biological Physics, Nonlinear Dynamics
311	Astrophysics and Astronomy
312	Mathematics
313	Atmospheric Science, Oceanography and Climate Research
315	Geophysics and Geodesy
316	Geochemistry, Mineralogy and Crystallography
318	Water Research
402	Mechanics and Constructive Mechanical Engineering
403	Process Engineering, Technical Chemistry
404	Heat Energy Technology, Thermal Machines, Fluid Mechanics
405	Materials Engineering
406	Materials Science
407	Systems Engineering
408	Electrical Engineering and Information Technology
409	Computer Science

How to serve diverse requirements with one single system?



Diverse Requirements



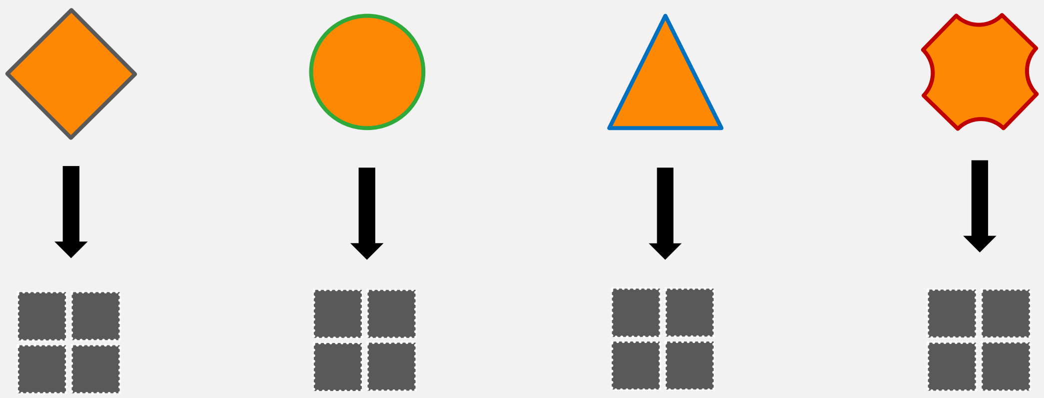
Node design

?

Monolithic vs. Modular Supercomputing Architectures

Monolithic
Supercomputing Architecture

APPLICATION & WORKLOAD CHARACTERISTICS

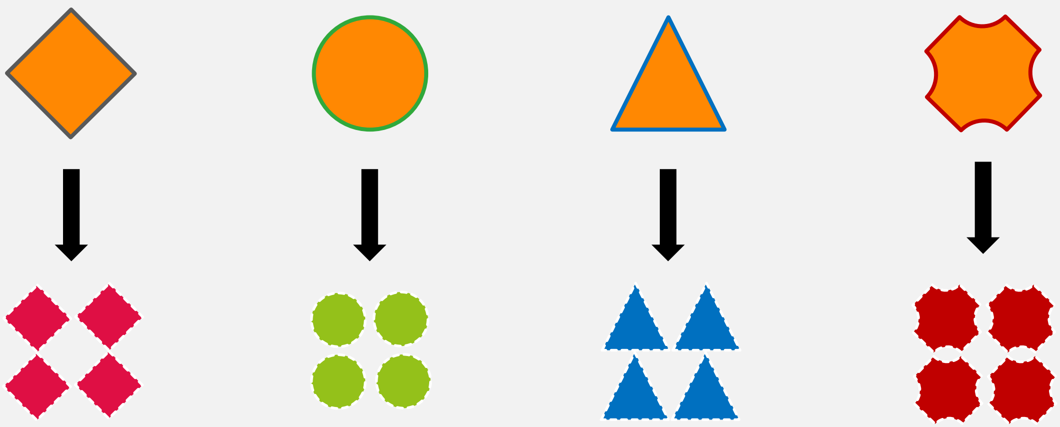


MONOLITHIC HARDWARE

SINGLE MODULE –
WITH ALL NODES THE SAME

Modular
Supercomputing Architecture

APPLICATION & WORKLOAD CHARACTERISTICS

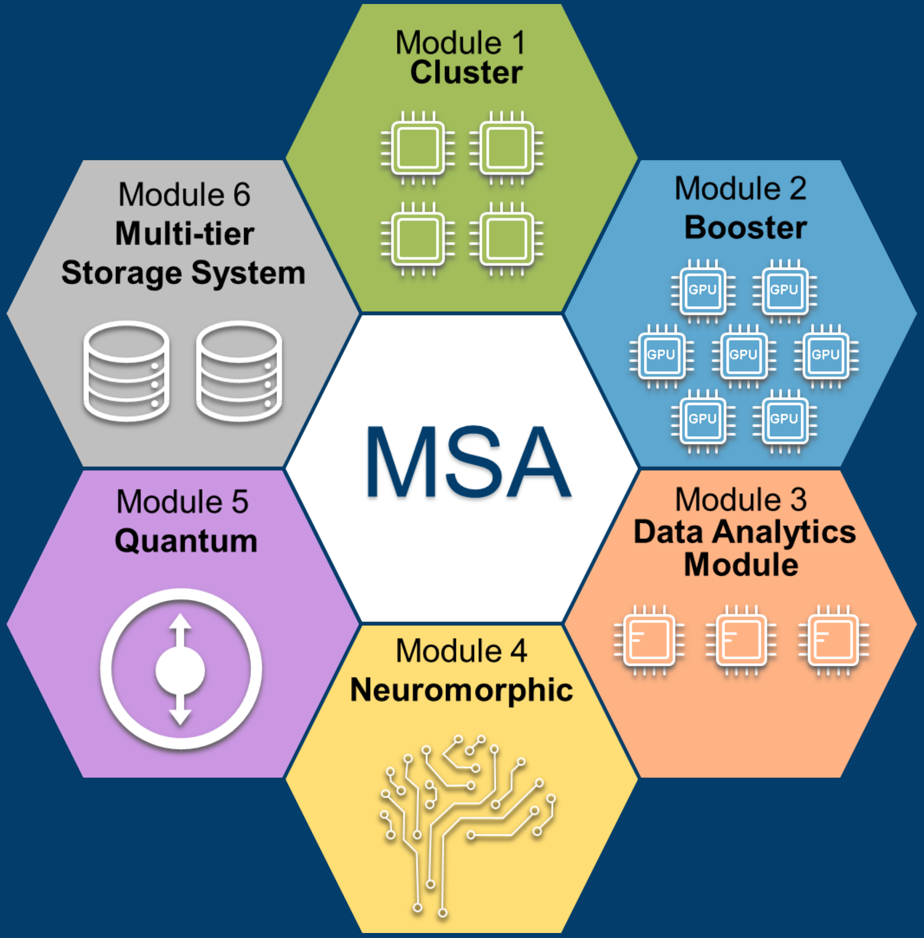


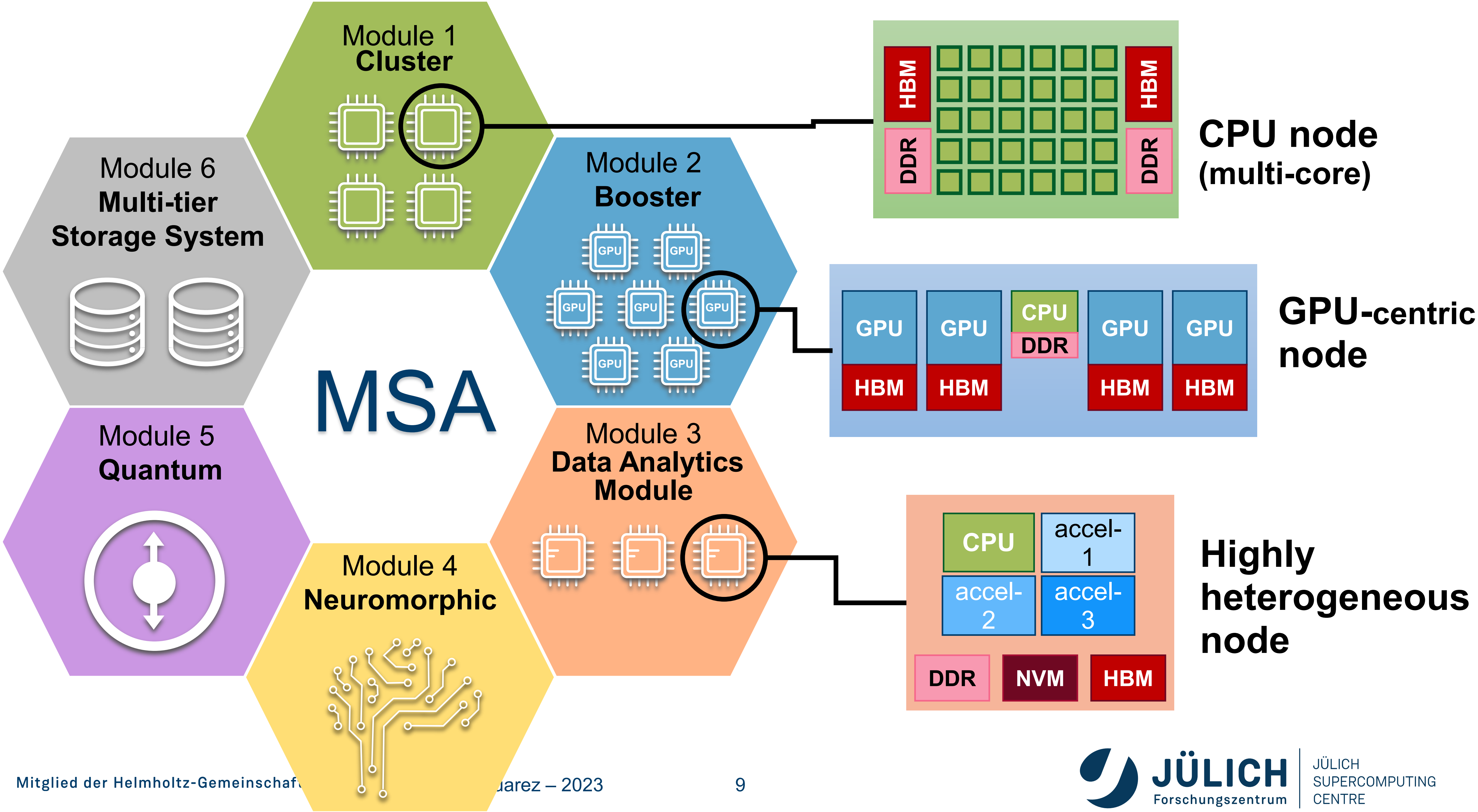
MODULAR HARDWARE

MULTIPLE MODULES -
EACH WITH TARGETED (DIFFERENT) NODES

OUTLINE

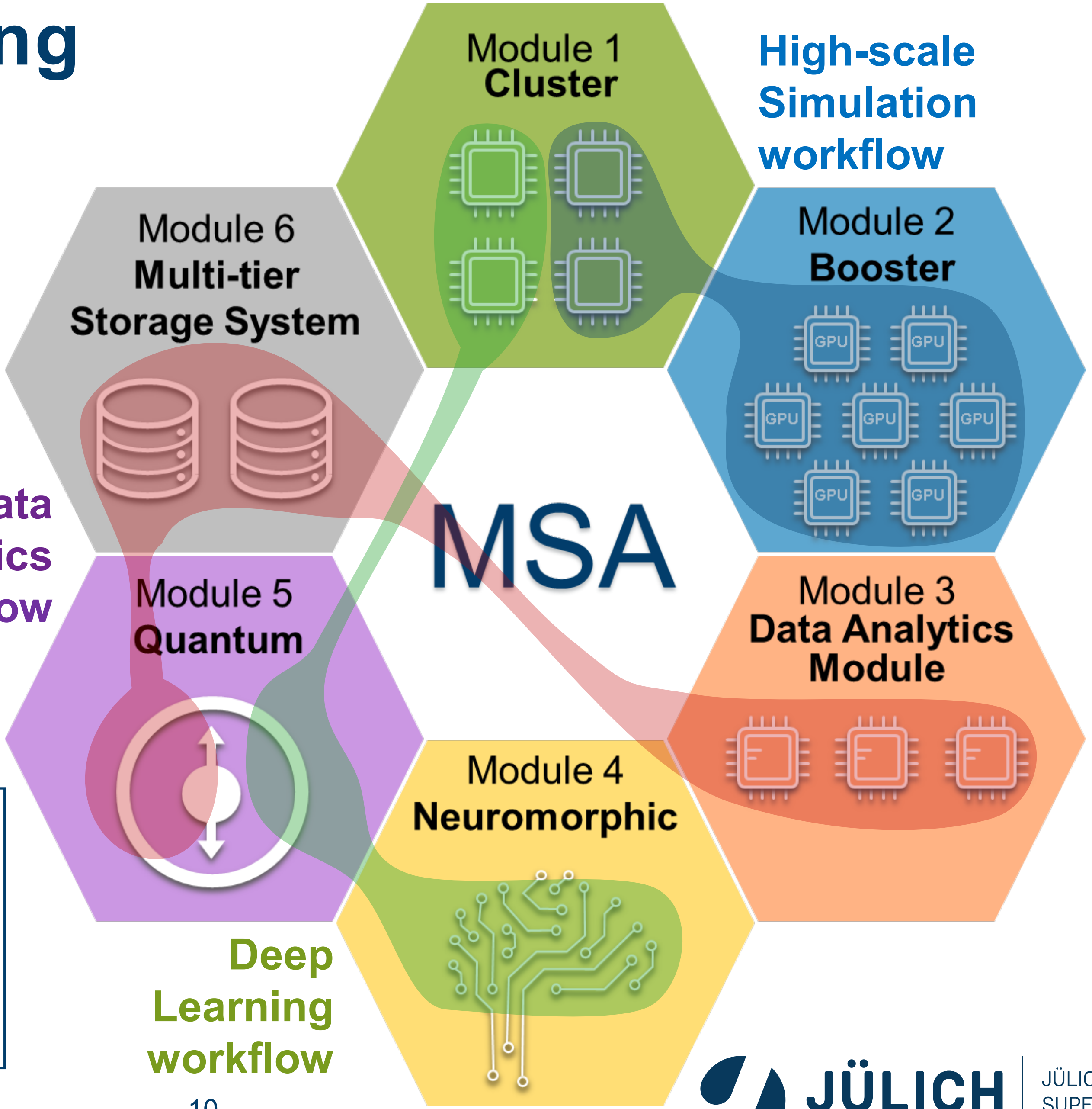
- JSC and its users
- System Architecture
- Software Stack
- Application Experience
- Summary




Module 1
Cluster
Module 2
Booster
GPU
Module 3
Data Analytics Module
Module 4
Neuromorphic
Module 5
Quantum
Module 6
Multi-tier Storage System
MSA
HBM
DDR
CPU node
(multi-core)
GPU
CPU
DDR
HBM
GPU-centric node
CPU
accel-1
accel-2
accel-3
DDR
NVM
HBM
Highly heterogeneous node

Modular Supercomputing Architecture

Serve diverse applications with composable heterogeneous resources



- **Suarez** et al. *"Modular Supercomputing Architecture – A Success Story of European R&D",* ETP4HPC White Paper. (2022) Available at https://www.etp4hpc.eu/white-papers.html#msa.
- **Suarez** et al., "*Modular Supercomputing Architecture: from idea to production*", Chapter 9 in Contemporary High Performance Computing: from Petascale toward Exascale, Volume 3, p 223-251, CRC Press. (2019)

The DEEP Prototypes

DEEP Projects

© FZJ

2015

DEEP Prototype
128 Xeon + 284 KNC nodes
InfiniBand + 1.5Gbit Extoll
550 TFlop/s

2016

DEEP-ER Prototype
16 Xeon + 8 KNL nodes
100Gbit Extoll
40 TFlop/s

2020

DEEP-EST Prototype
55 Cluster + 75 Booster + 16 Data Analytics
100 Gbit Extoll + InfiniBand + Eth
800 TFlop/s

Modular Supercomputer JUWELS

Entry in Nov'20 TOP500 The List.

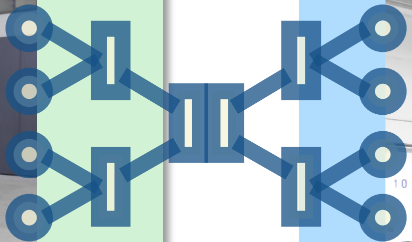
JUWELS Cluster #44

Intel Xeon (Skylake) processor

InfiniBand EDR network

2,500 compute nodes

10 PFLOP/s peak (CPU-based)



JUWELS Booster #7

AMD EPYC Rome 7402 processor

3,700 NVIDIA A100 GPUs

InfiniBand HDR DragonFly+

70 PFLOP/s peak (GPU-based)

Funded through SiVeGCS (BMBF, MWK-NRW)

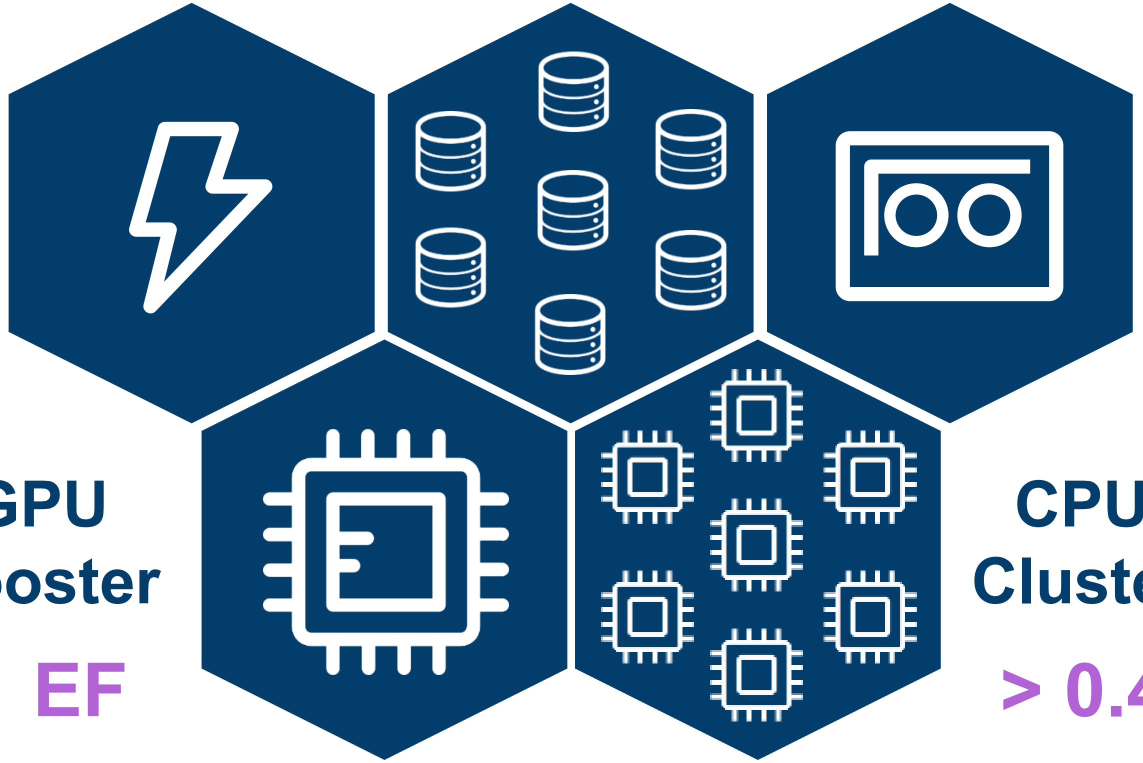
JUPITER – Modular Exascale Computer

> 1 EB

Parallel
High Bandwidth
Flash Module

Parallel
High Capacity
Data System

High Capacity
Backup/Archive
System



GPU Booster

> 1 EF

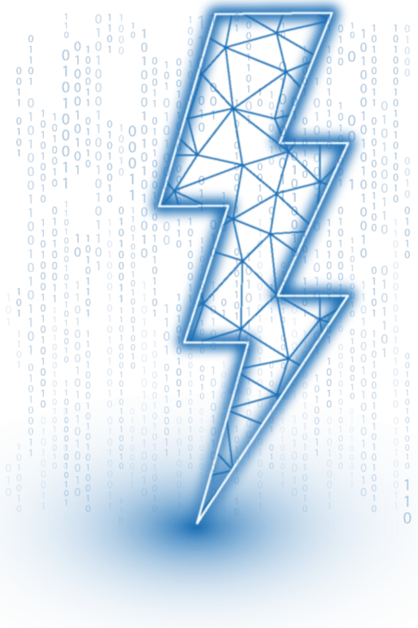
CPU Cluster

> 0.4 B/FLOP

Basis Configuration

Target >20×

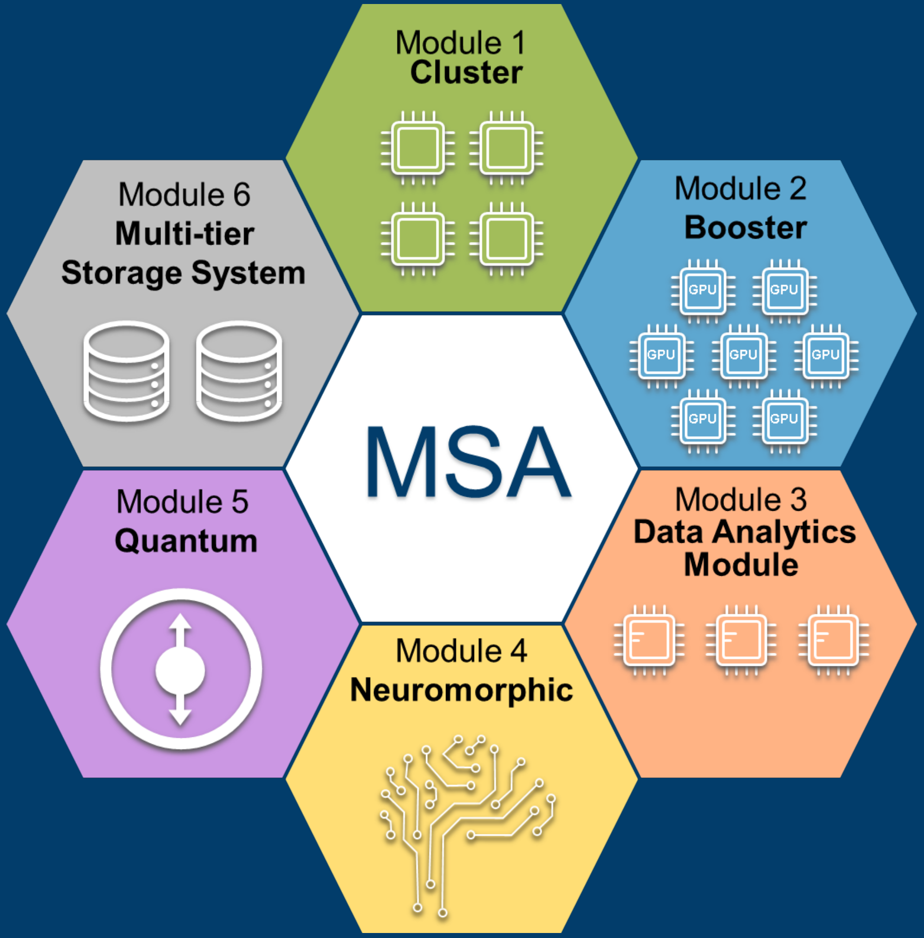
application performance compared to JUWELS Booster



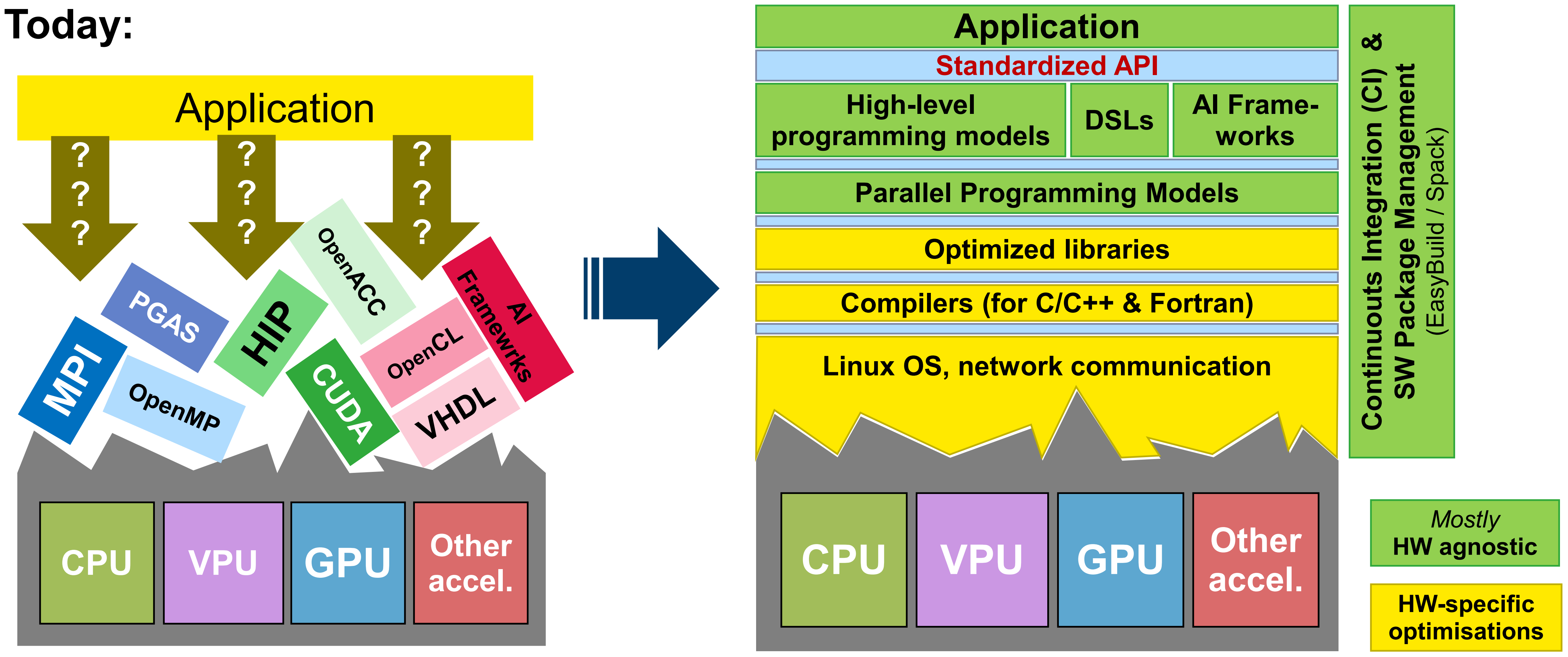
fz-juelich.de/jupiter

OUTLINE

- JSC and its users
- System Architecture
- Software Stack
- Application Experience
- Summary

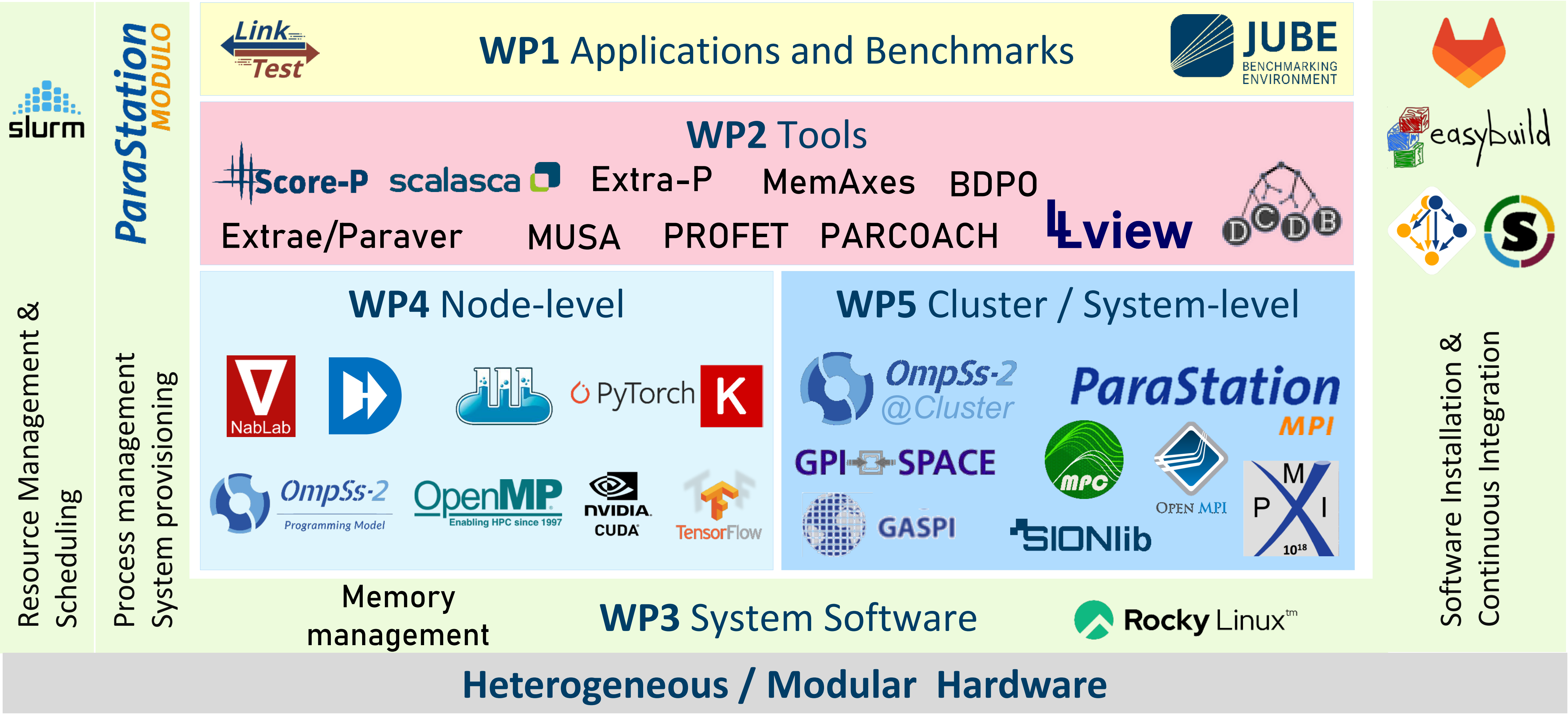


Software Environment on Heterogeneous systems

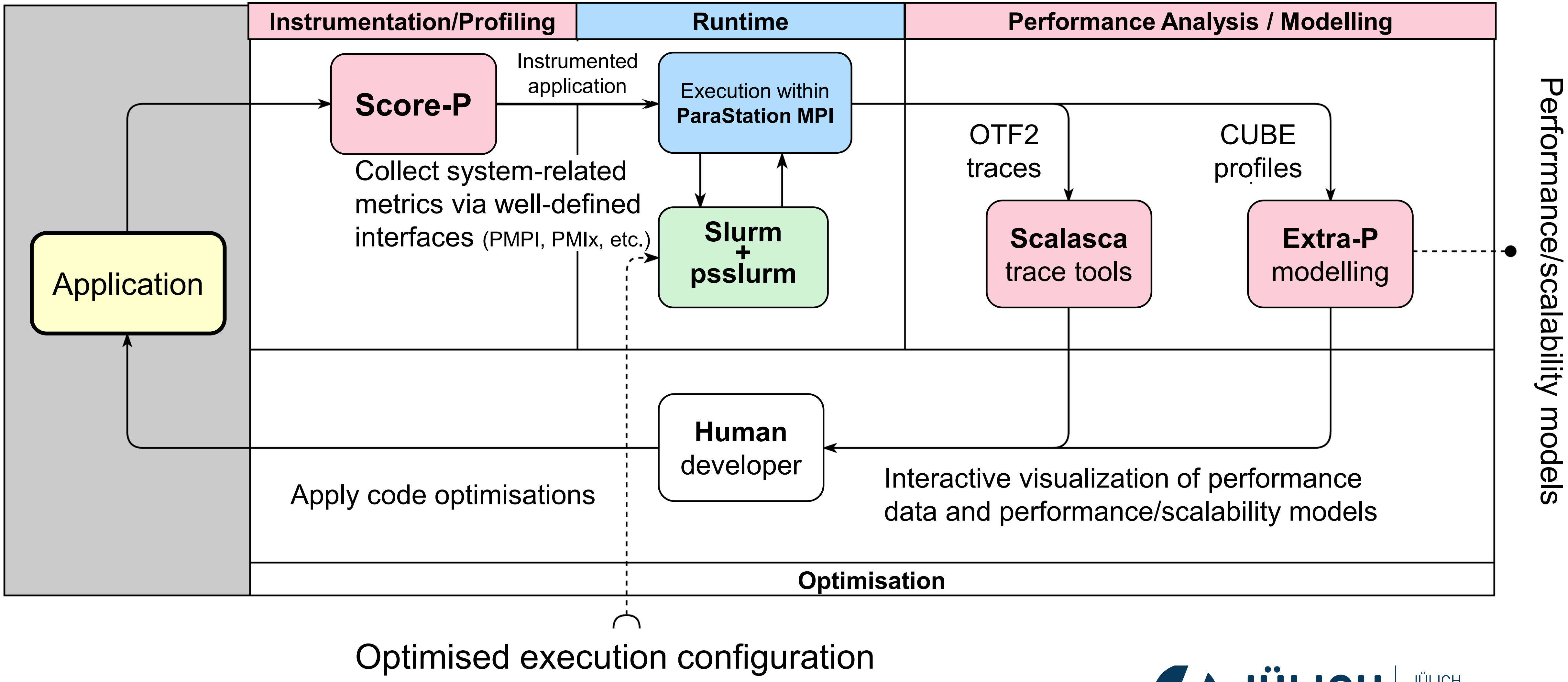


Integrated SW Stack

DEEP-SEA



Optimisation Cycles – e.g. MSA-related OC

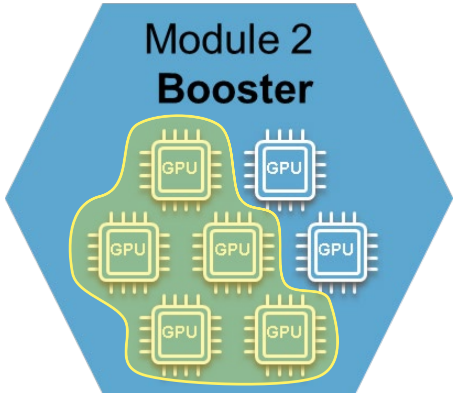


How do applications run on the MSA?

DEEP Projects

A) Only on one module

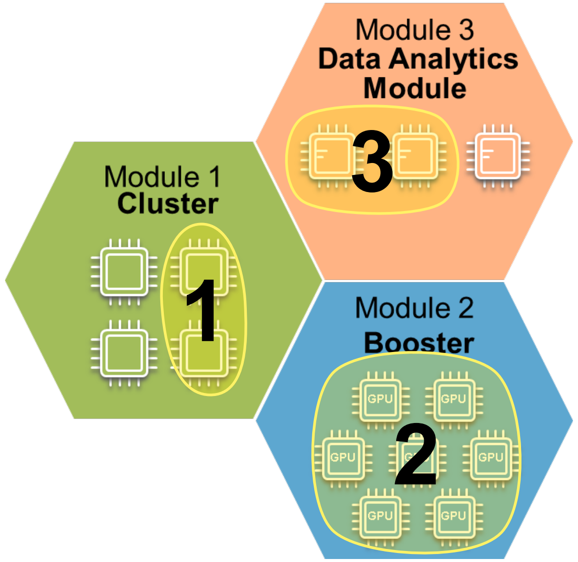
(e.g. the Booster)



Typical for **tightly-coupled**, high-scaling **applications**

(e.g. dense M×M)

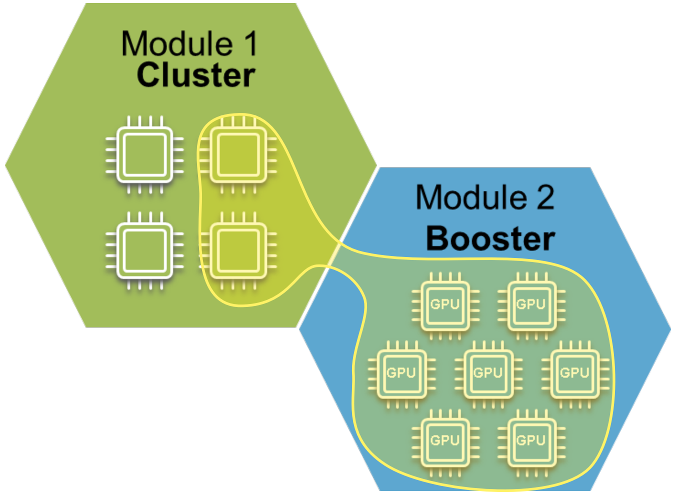
B) Job chain



Typical of complex **application workflows**

(MPMD applications, e.g. pre-processing, simulation, data analysis)

C) MPI across modules



Typical for **multi-physics** or multi-scale applications

(e.g. coupled climate models)

How do applications run on the MSA?

DEEP Projects

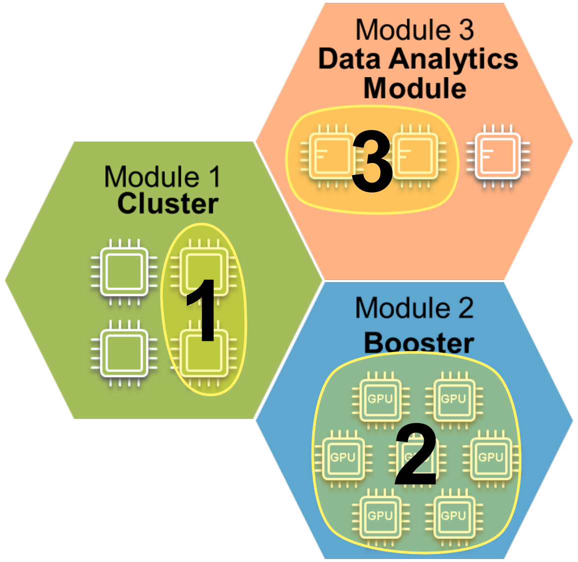
A) Only on one module

(e.g. the Booster)

Typical for **tightly-coupled**, high-scaling **applications**

(e.g. dense M×M)

B) Job chain



Typical of complex **application workflows**

(MPMD applications, e.g. pre-processing, simulation, data analysis)

C) MPI across modules

Typical for **multi-physics** or multi-scale applications

(e.g. coupled climate models)

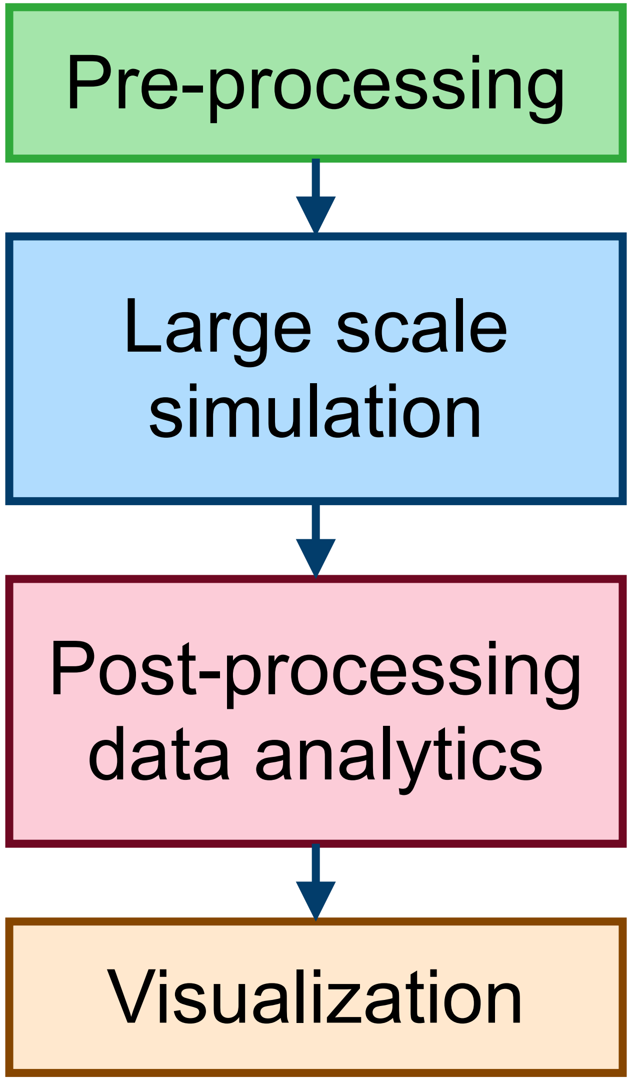
B) Inter-module Workflow execution

- Application workflows execute various distinct steps:
  - **Each step is a stand-alone code** and executable
  - Each of those codes might run best on different HW
  - The result is an heterogeneous job

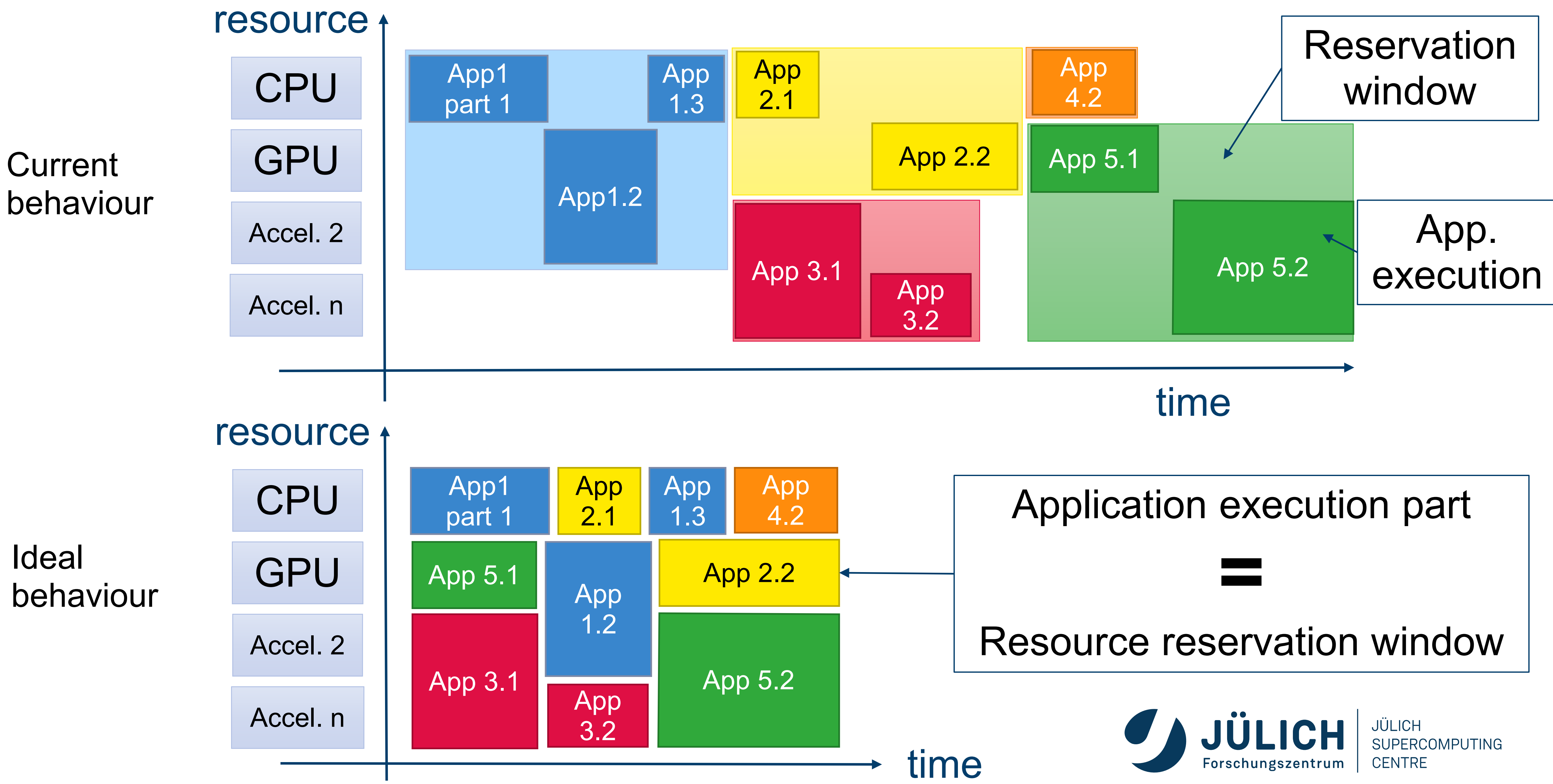
- **Slurm** scheduler is used to submit heterogeneous jobs forming a **job pack** allocation using colon notation for **salloc, sbatch, srun**, e.g.:

```
$ srun  -N 1 -p cluster ./exe1 \
      : -N 2 -p booster ./exe2
```

  - allows even different **executables** (even diff. project accounts)
  - for each job in the job pack, resources specified individually



Malleability and Dynamic Scheduling



JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

How do applications run on the MSA?

A) Only on one module

(e.g. the Booster)

Typical for **tightly-coupled**, high-scaling **applications**

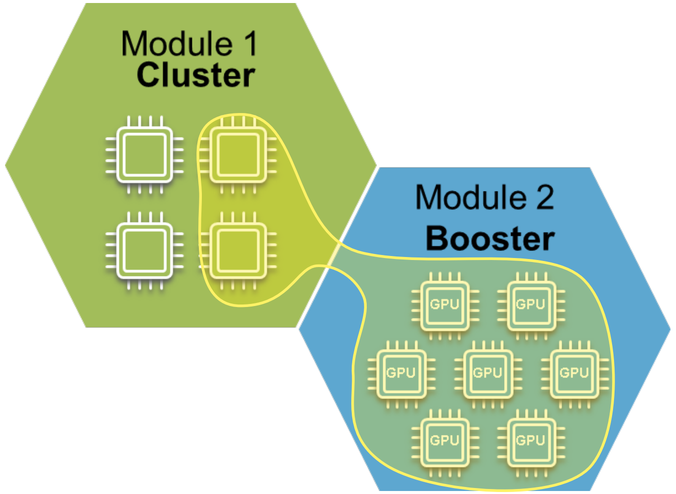
(e.g. dense M×M)

B) Job chain

Typical of complex **application workflows**

(MPMD applications, e.g. pre-processing, simulation, data analysis)

C) MPI across modules

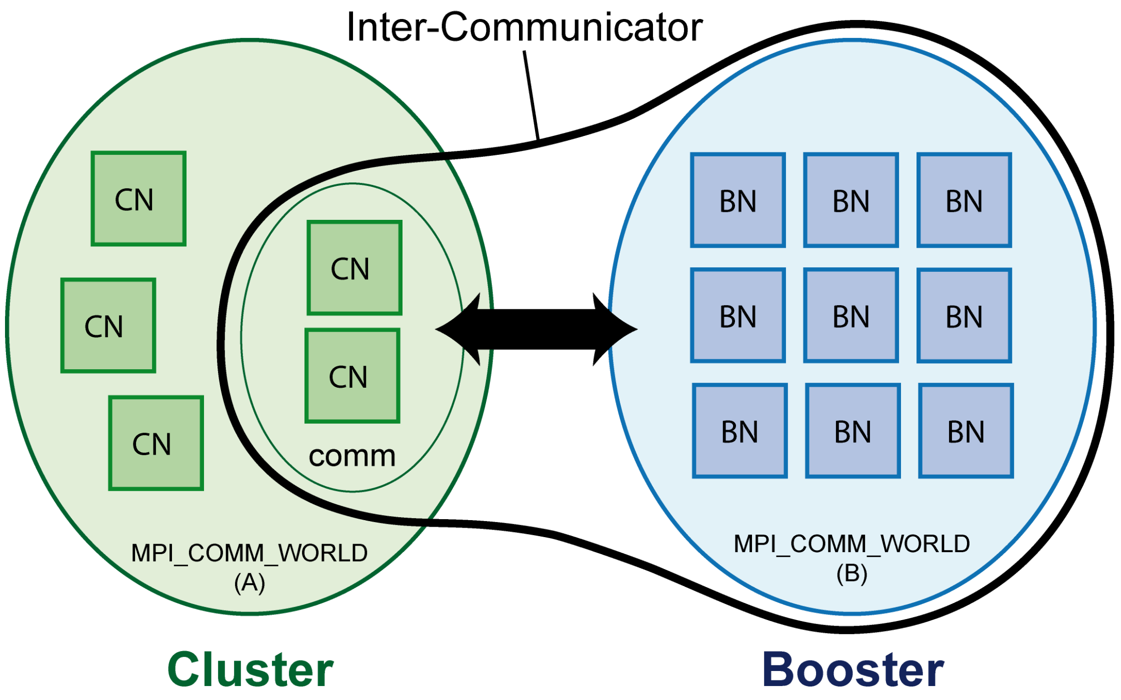


Typical for **multi-physics** or multi-scale applications

(e.g. coupled climate models)

C) Inter-module MPI communication

- **Collective offload process**
- Connect two MPI worlds via an intercommunicator
  - *MPI_Comm_spawn()*
  - *MPI_Connect()*
  - *MPI_Comm_split()*
- Transparent data exchange via MPI

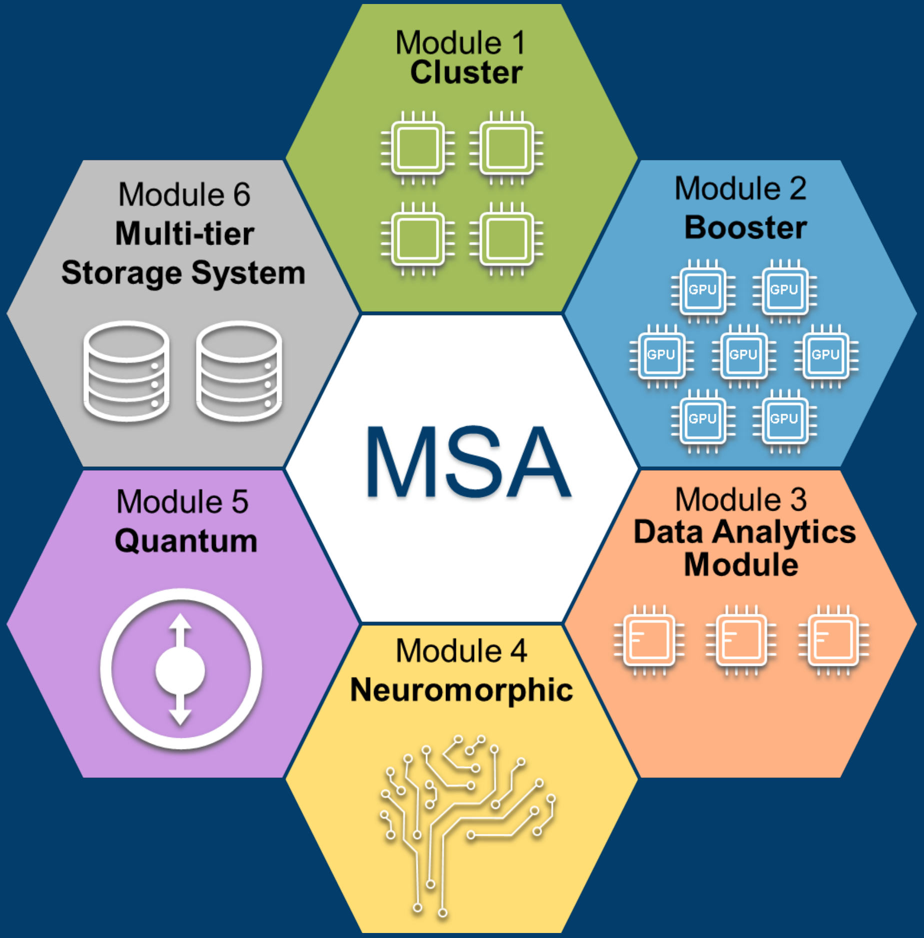


ParaStation MPI

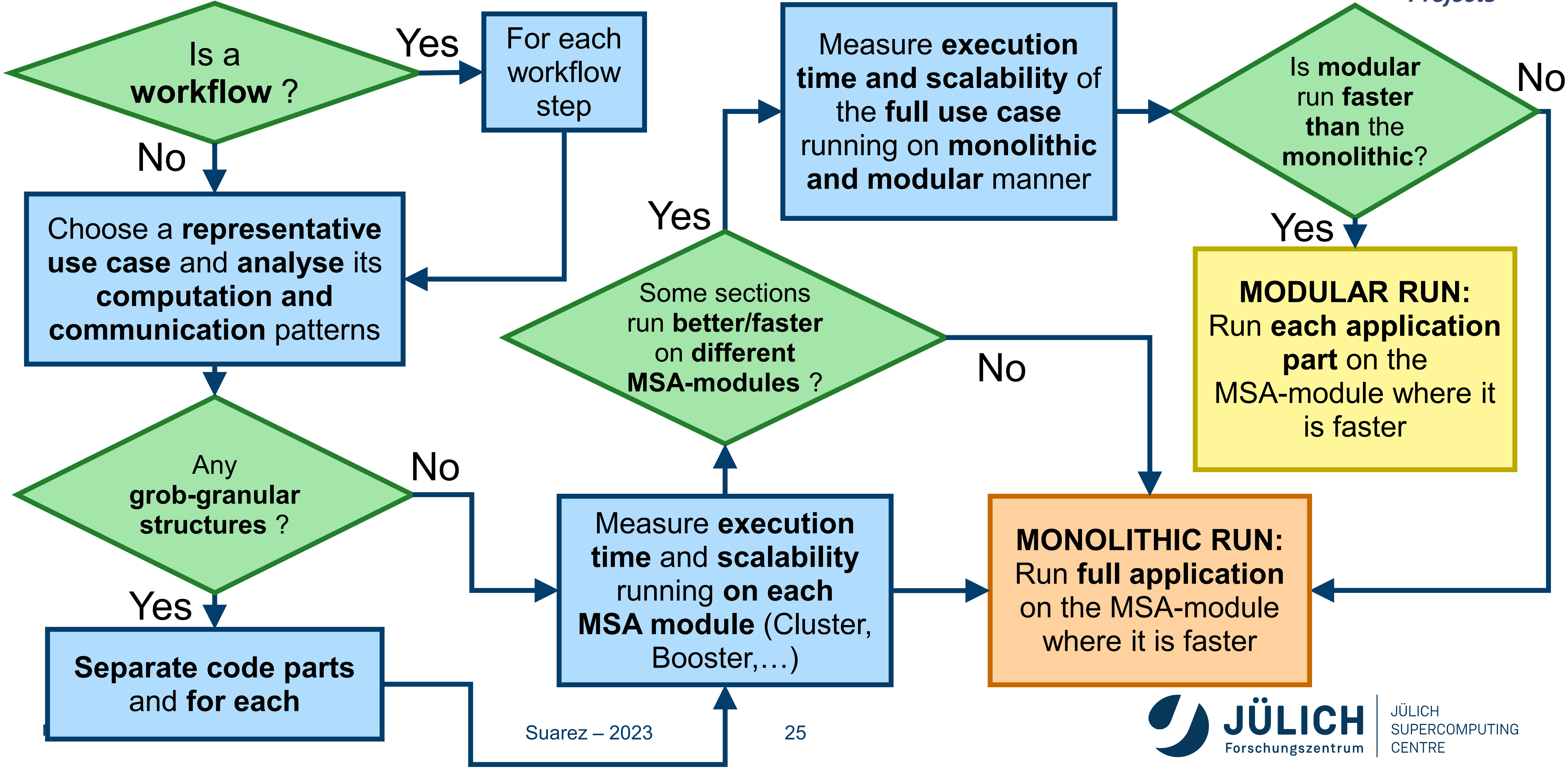
- **Clauss** et al., *Dynamic Process Management with Allocation-internal Co-Scheduling towards Interactive Supercomputing*, COSH@HiPEAC, (2016)

OUTLINE

- **System Architecture**
- **Software Stack**
- **Application Experience**
- **Summary**



Deciding how to run my Application on MSA



How do applications run on the MSA?

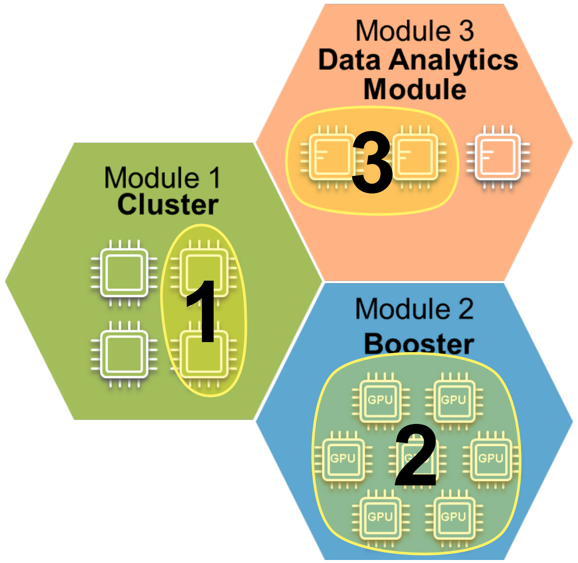
A) Only on one module

(e.g. the Booster)

Typical for **tightly-coupled**, high-scaling **applications**

(e.g. dense M×M)

B) Job chain



Typical of complex

application workflows

(MPMD applications, e.g. pre-processing, simulation, data analysis)

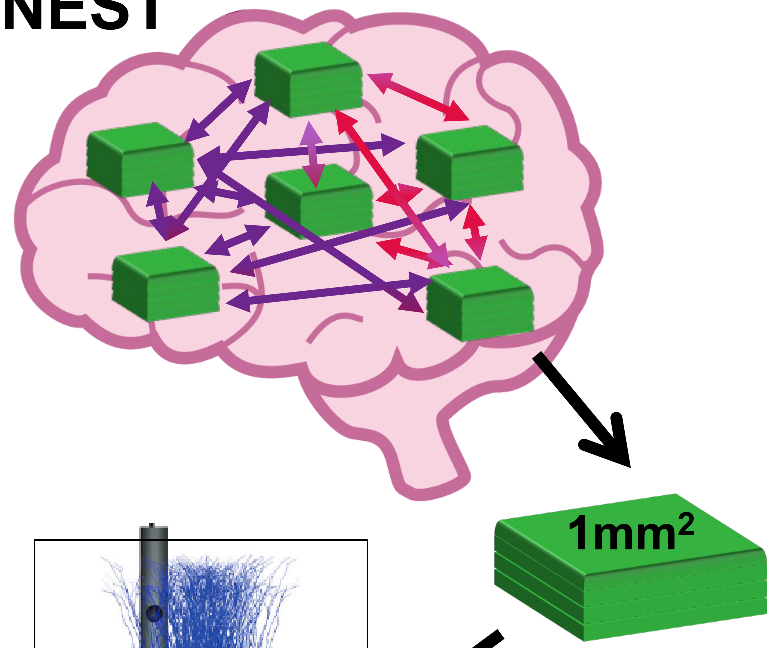
C) MPI across modules

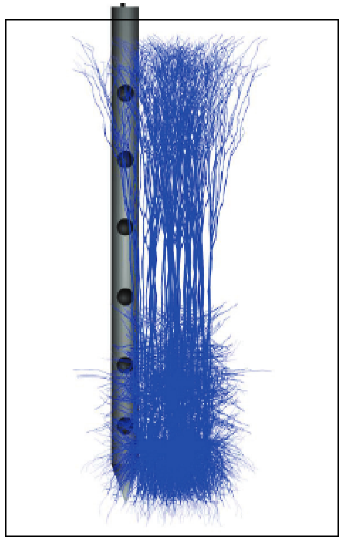
Typical for **multi-physics** or multi-scale applications

(e.g. coupled climate models)

B) Example Workflow: Brain Simulation

NEST





Arbor

- **NEST**: multi-area model (large-scale network simulation)
  - relatively **low computational cost**
    - updates only simple model neurons
  - frequent and unpredictable exchange of neuronal signals
  - **communication and memory bound**

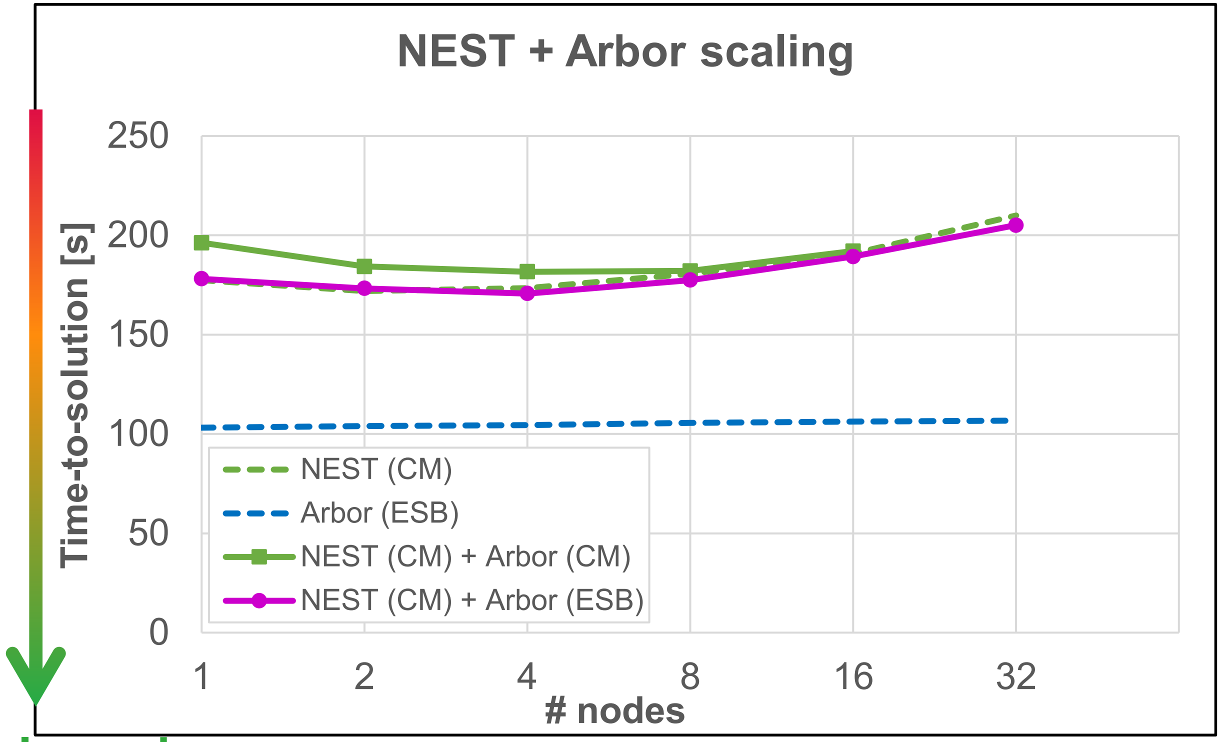
Cluster

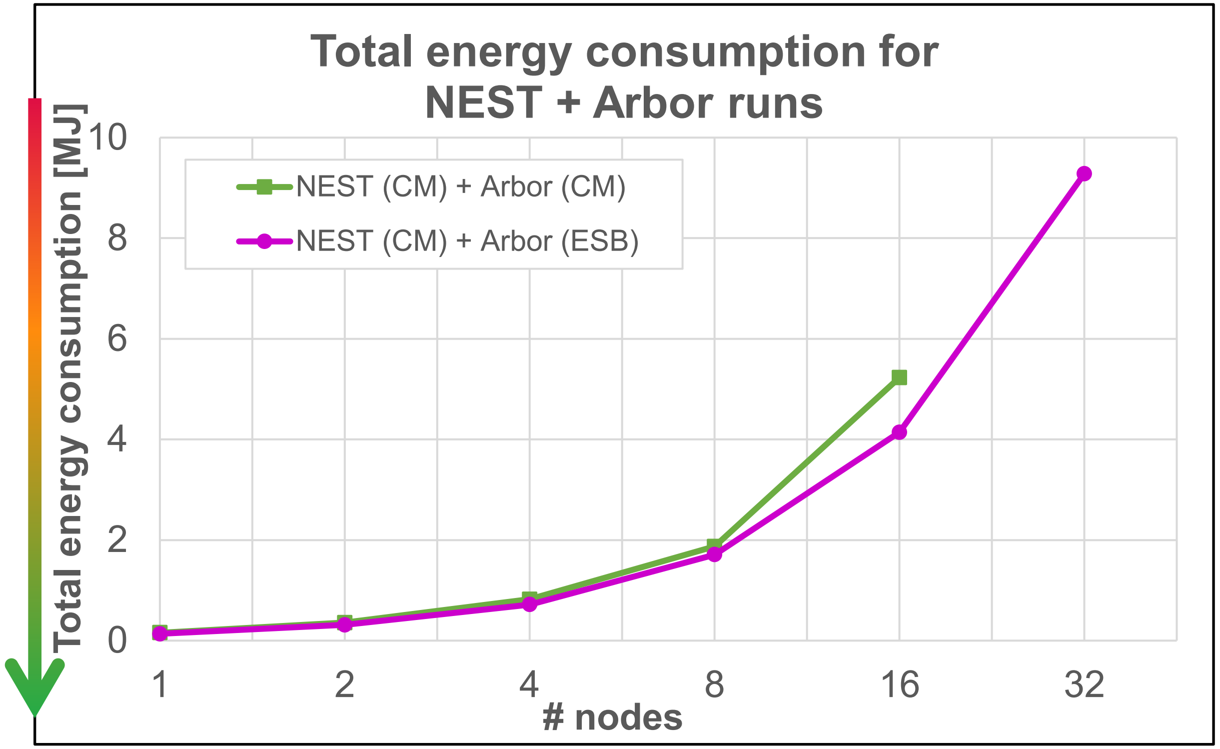
- **Arbor**: detailed multi-compartment neuron simulation
  - high **computational cost** per neuron, few communication
  - **compute bound**
  - **designed for vectorised architectures**

Booster

Suarez et al. *Modular Supercomputing for Neuroscience*, Lecture Notes in Computer Science 12339, 63-80 (2021), http://hdl.handle.net/2128/28352

B) Example Workflow: NEST + Arbor





lower is better

Plesser et al. *Neuroscience with NEST, Arbor and Elephant,* IAS Series 48, 27-46 (2021) http://hdl.handle.net/2128/30531

- Extended NEST to a co-simulation with Arbor on MSA
  - **no runtime penalty, and lower energy consumption**

How do applications run on the MSA?

A) Only on one module

(e.g. the Booster)

Typical for **tightly-coupled**, high-scaling **applications**

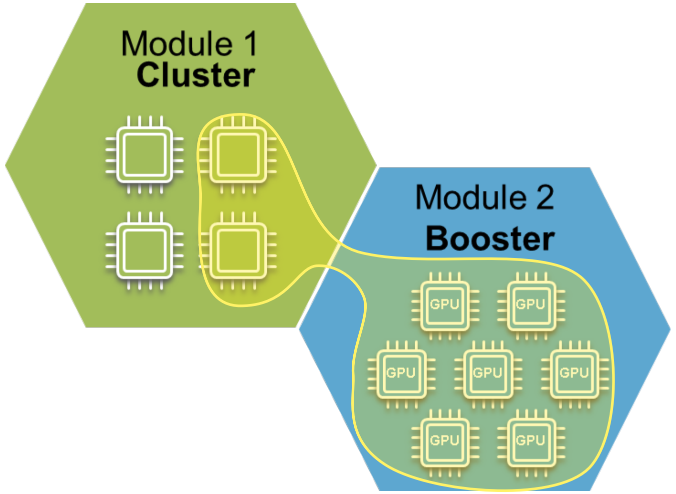
(e.g. dense M×M)

B) Job chain

Typical of complex **application workflows**

(MPMD applications, e.g. pre-processing, simulation, data analysis)

C) MPI across modules



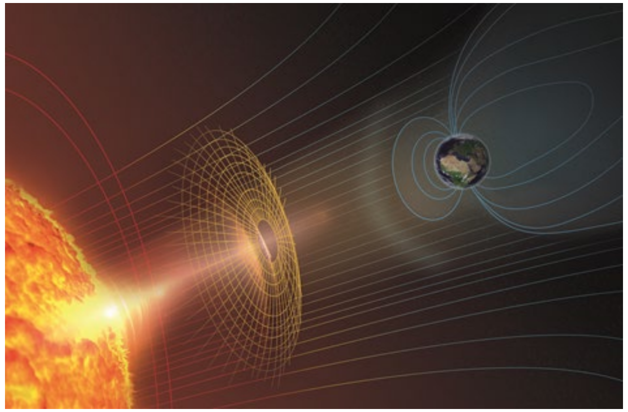
Typical for **multi-physics** or multi-scale applications

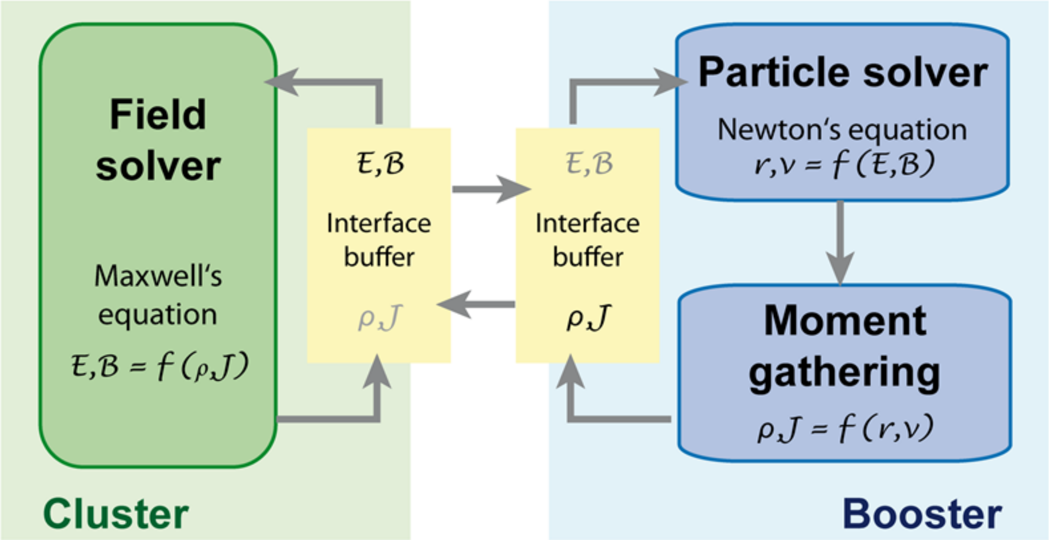
(e.g. coupled climate models)

C) Example MPI coupling: xPic

- **Space Weather simulation**
  - Simulates plasma produced in solar eruptions and its interaction with the Earth magnetosphere
  - Particle-in-Cell (PIC) code from KULeuven
- **Two solvers:**
  - **Field solver**: Computes electromagnetic (EM) field evolution
    - Limited code scalability
    - Frequent, global communication
  - **Particle solver**: Calculates motion of charged particles in EM-fields
    - Highly parallel
    - Billions of particles
    - Long-range communication

xPic





- **Kreuzer**, et al, *"Application performance on a Cluster-Booster system"*, IPDPSW, HCW (2018) [doi: 10.1109/IPDPSW.2018.00019]

C) xPic – Original Configuration

```
for (auto i=beg+1; i<=end; i++){
  fld.solver->calculateE();
  fld.cpyToArr_F();



  pcl.cpyFromArr_F();
  for (auto is=0; is<nspec; is++) {
    pcl.species[is].ParticlesMove();
    pcl.species[is].ParticleMoments();
  }
  pcl.cpyToArr_M();



  fld.solver->calculateB();
  fld.cpyFromArr_M();
}
```

`fld`: Field Solver

Copy information between solvers

`plc`: Particle Solver

C) xPic – Code Partition

```
#ifdef __CLUSTER__
for (auto i=beg+1; i<=end; i++){
  fld.solver->calculateE();
  fld.cpyToArr_F();
  ClusterToBooster();
  // Auxiliary computations
  ClusterWait();







BoosterToCluster();

BoosterWait();
  fld.solver->calculateB();
  fld.cpyFromArr_M();
}
#endif
```

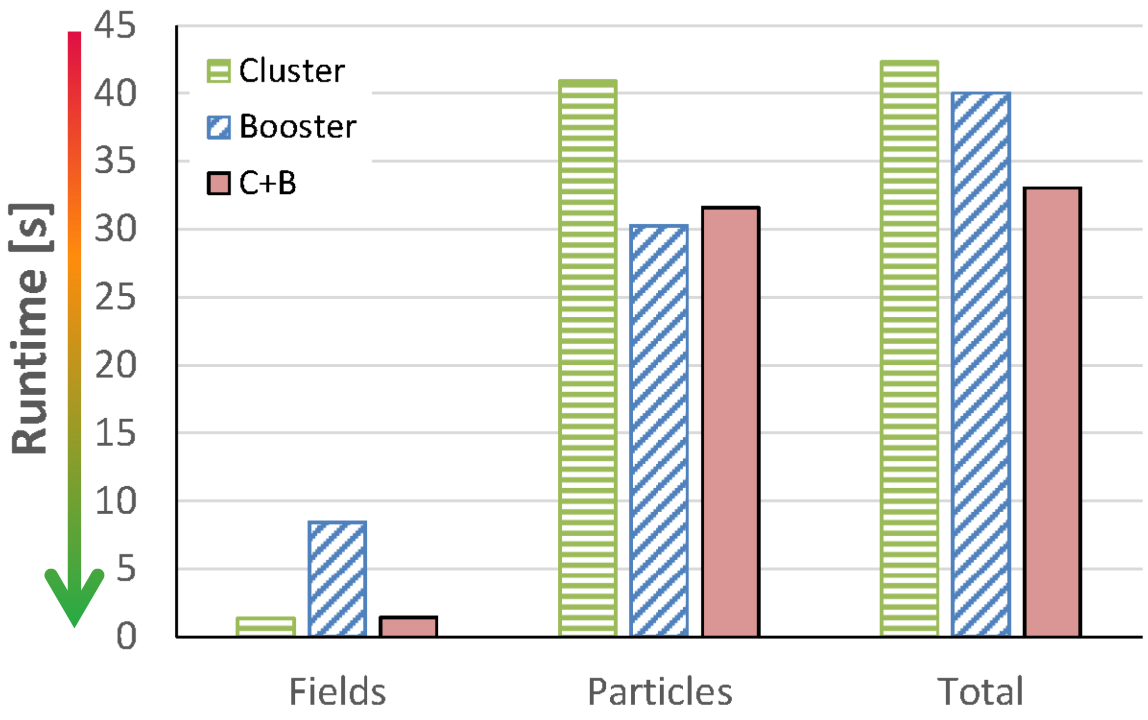
```
#ifdef __BOOSTER__
for (auto i=beg+1; i<=end; i++){


  ClusterToBooster();

  ClusterWait();
  pcl.cpyFromArr_F();
  for (auto is=0; is<nspec; is++) {
    pcl.species[is].ParticlesMove();
    pcl.species[is].ParticleMoments();
  }
  pcl.cpyToArr_M();
  BoosterToCluster();
  // I/O and auxiliary computations
  BoosterWait();



}
#endif
```

C) xPic – Performance Results

- **Field solver**: 6× faster on Cluster
- **Particle solver**: 1.35 × faster on Booster
- **Overall performance gain:**

1× node	**28% × gain** compared to Cluster alone **21% × gain** compared to Booster alone
8× nodes	**38% × gain** compared to Cluster only **34% × gain** compared to Booster only

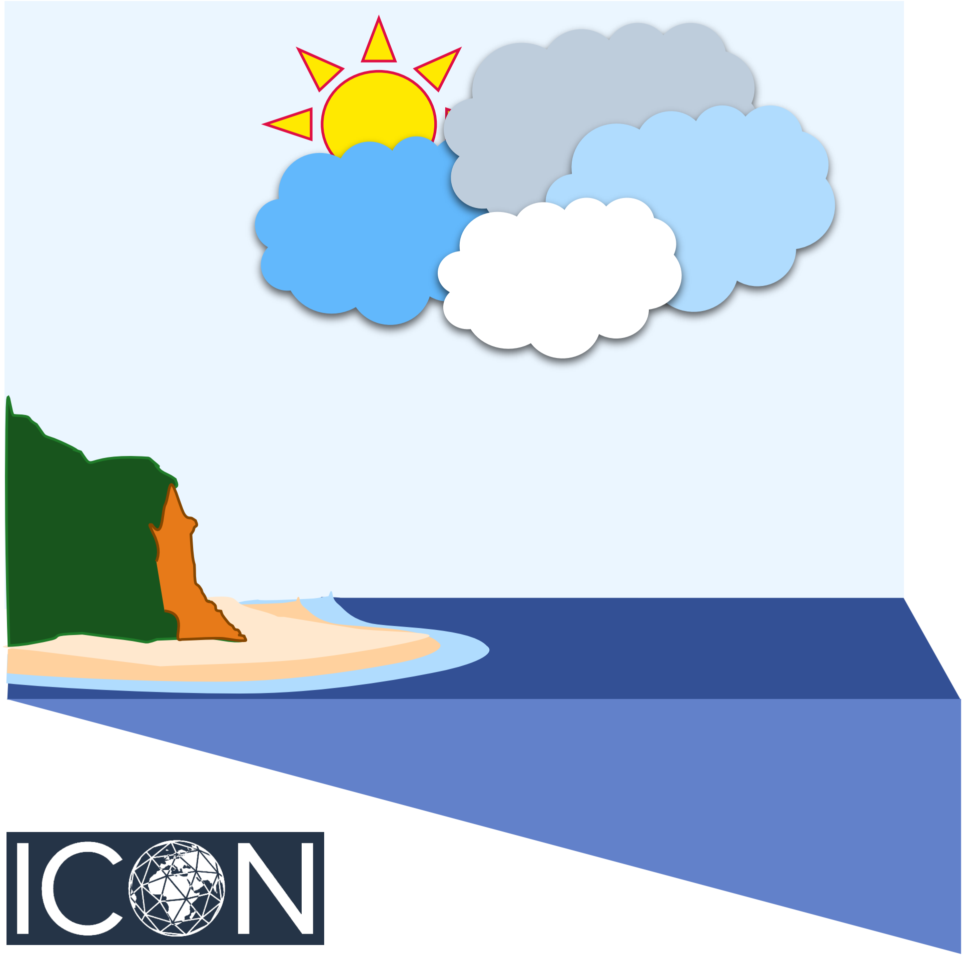
– 3%-4% overhead per solver for C+B communication (point to point)



#cells per node	4096
#particles per cell	2048
Compilation flags	-openmp, -mavx (Cluster) -xMIC-AVX512 (Booster)

- **Kreuzer**, et al, *"Application performance on a Cluster-Booster system"*, IPDPSW, HCW (2018) [doi: 10.1109/IPDPSW.2018.00019]

C) Example MPI coupling: ICON



See presentation from Olaf Stein this afternoon!

ICON-A:
Atmospheric model
(GPU optimised)

→ Booster

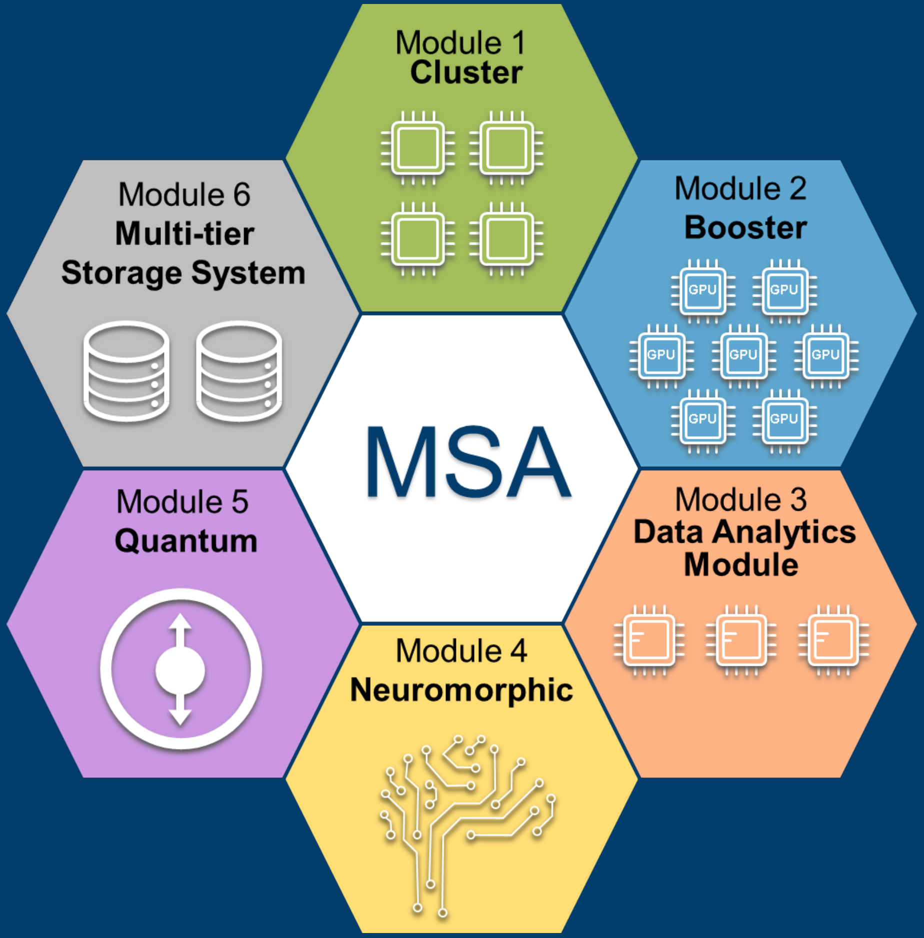
ICON-O:
Ocean model

→ Cluster

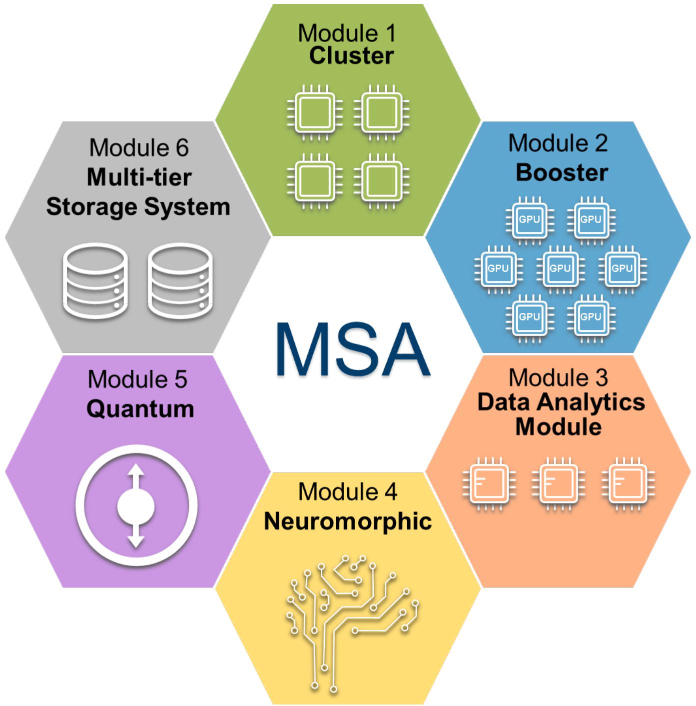
- **Bishnoi,** et al. *Earth system modeling on Modular Supercomputing Architectures: coupled atmosphere-ocean simulations with ICON 2.6.6-rc*, EGUsphere [preprint], https://doi.org/10.5194/egusphere-2023-1476, (2023)

OUTLINE

- System Architecture
- Software Stack
- Application Experience
- Summary



SUMMARY – Modular Supercomputing



- **Hardware heterogeneity at system level**
  - Segregated / disaggregated compute modules
  - Scale modules independently ➔ energy efficiency
  - Gradual integration of disruptive technologies
- **to support application diversity**
  - Adapt system to user portfolio
  - Choose the appropriate mix of resources for each use case
  - Speed up each application part with the appropriate hardware
- **and maximise throughput**
  - Efficient resource sharing between workloads
  - Dynamic resource allocation and malleability

THANK YOU!

www.deep-projects.eu
@DEEPprojects
@deep-projects

The **DEEP Projects** have received funding from the European Commission's FP7, H2020, and EuroHPC Programmes, under Grant Agreements n° 287530, 610476, 754304, and 955606.

The EuroHPC Joint Undertaking (JU) receives support from the European Union's Horizon 2020 research and innovation programme and Germany, France, Spain, Greece, Belgium, Sweden, United Kingdom, Switzerland