Towards Modular Supercomputing

The DEEP and DEEP-ER take on Heterogeneous Cluster Architectures

♦

Norbert Eicker

Jülich Supercomputing Centre & University of Wuppertal

SC16 Salt Lake City, Utah | hpc matters.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (*FP7/2007-2013*) under *Grant Agreements* n° 287530 and n° 610476

Outline

- Motivation
  - Why heterogeneous systems
  - How to organize heterogeneity
- DEEP
  - General concept
  - Hardware architecture
  - Programming paradigm
- DEEP-ER
- Modular Supercomputing
- Summary

DEEP face-to-face Leuven

DEEP-ER kickoff Jülich

DEEP and DEEP-ER

EU-Exascale projects

20 partners

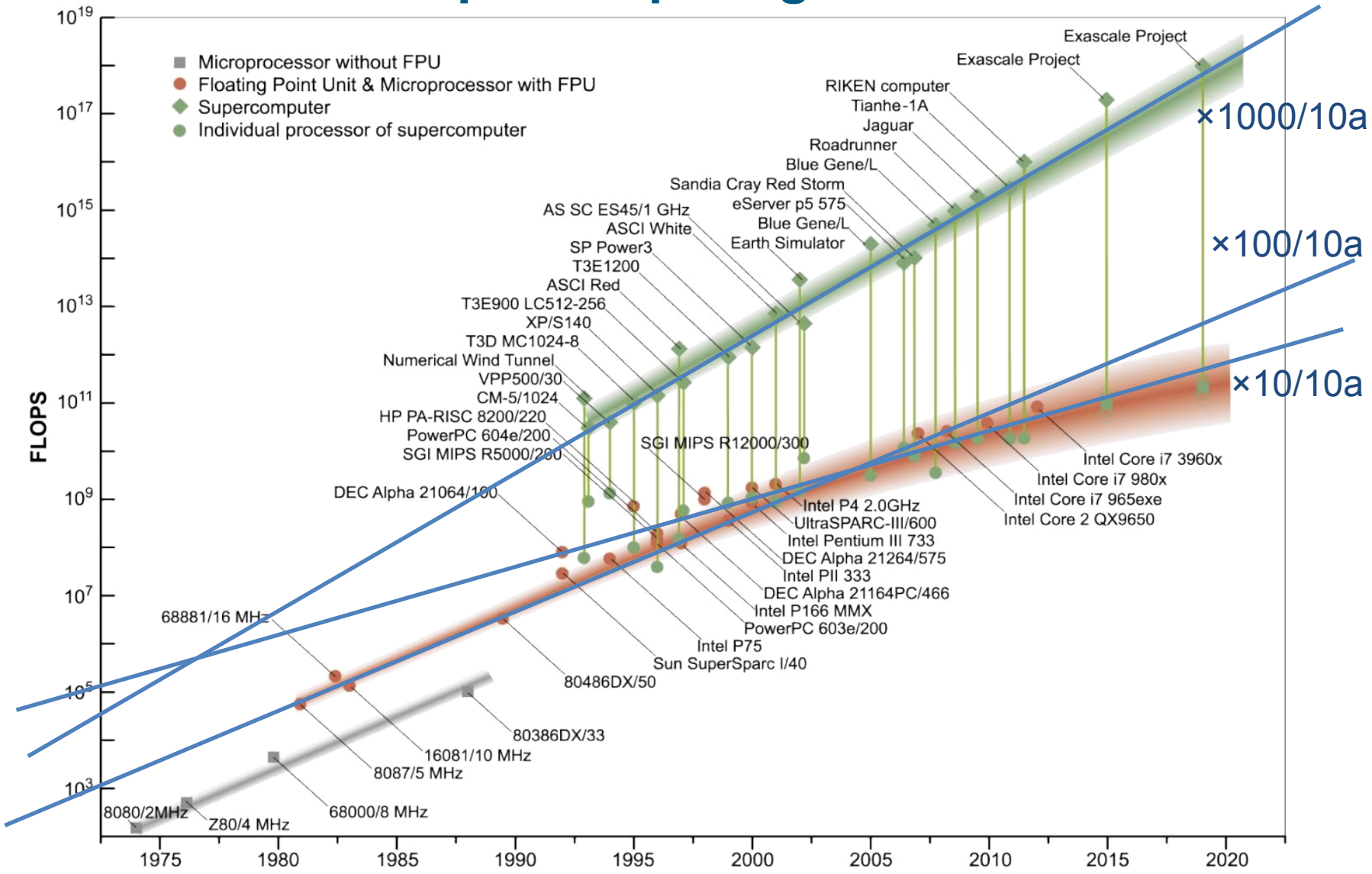
Total budget: 28,3 M€

EU-funding: 14,5 M€

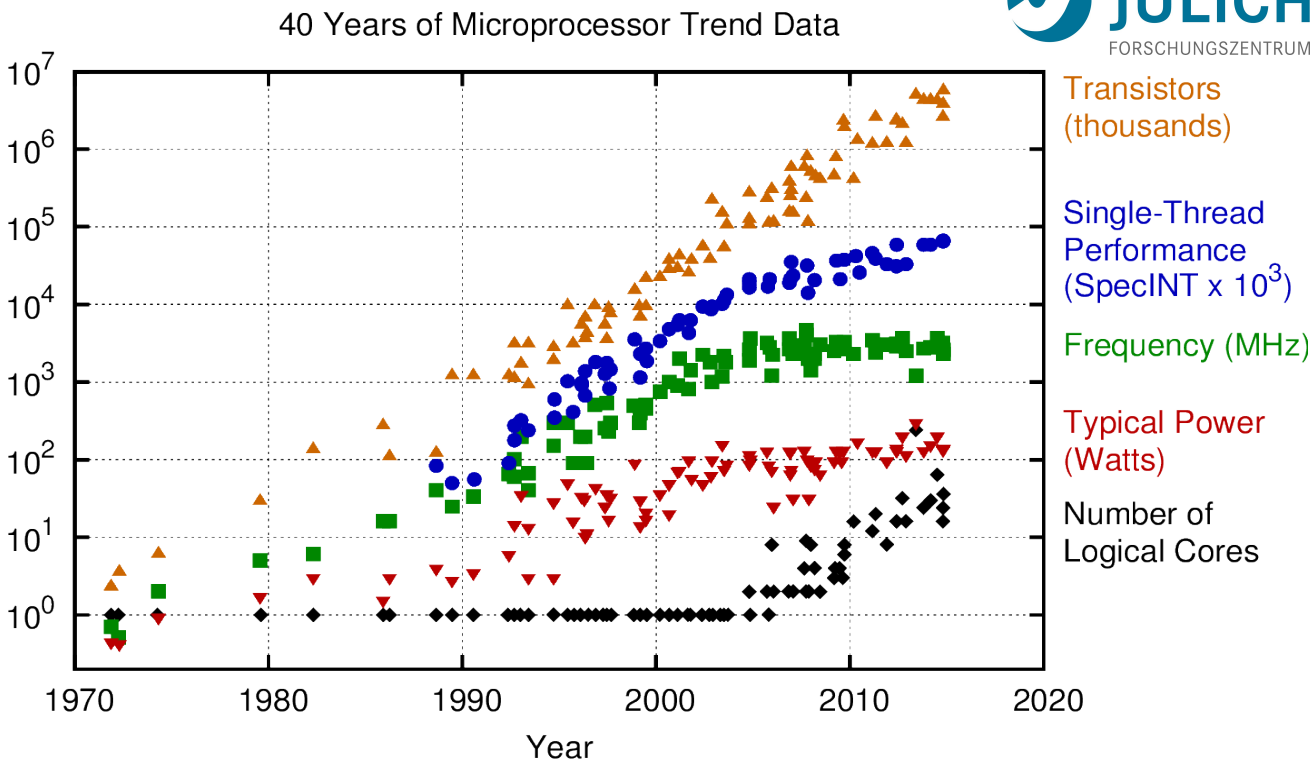
Nov 2011 – Mar 2017

www.deep-project.eu www.deep-er.eu

Evolution of Supercomputing



Moore's law



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

Observation

- Clock stagnates since 2002 at ~3 GHz
  - *Few exceptions: Power6, Power7, Gaming*
- # transistors still increases

Strategies for acceleration

- Multi-Core → Many-Core processors
- Vectorization

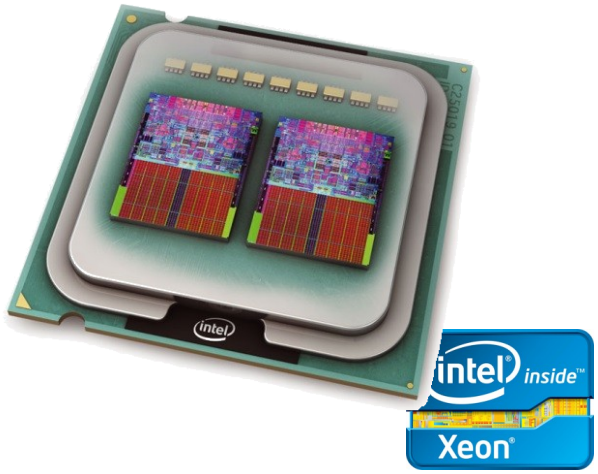
General purpose processors

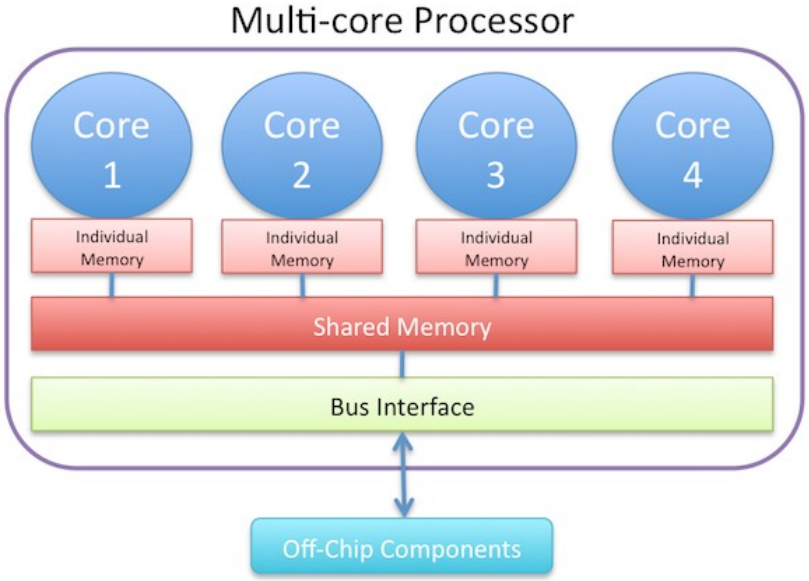
Characteristics

- Broad range of capabilities
- Today always multi-core
  - *Up to about 20 cores*
- High single thread performance
  - *High frequency*
  - *Out of order processing*
- Large memory per core
- Standard programming env.
  - *MPI, OpenMP, etc.*

Shortcomings

- Limited energy efficiency
- Larger $/FLOP





Accelerators

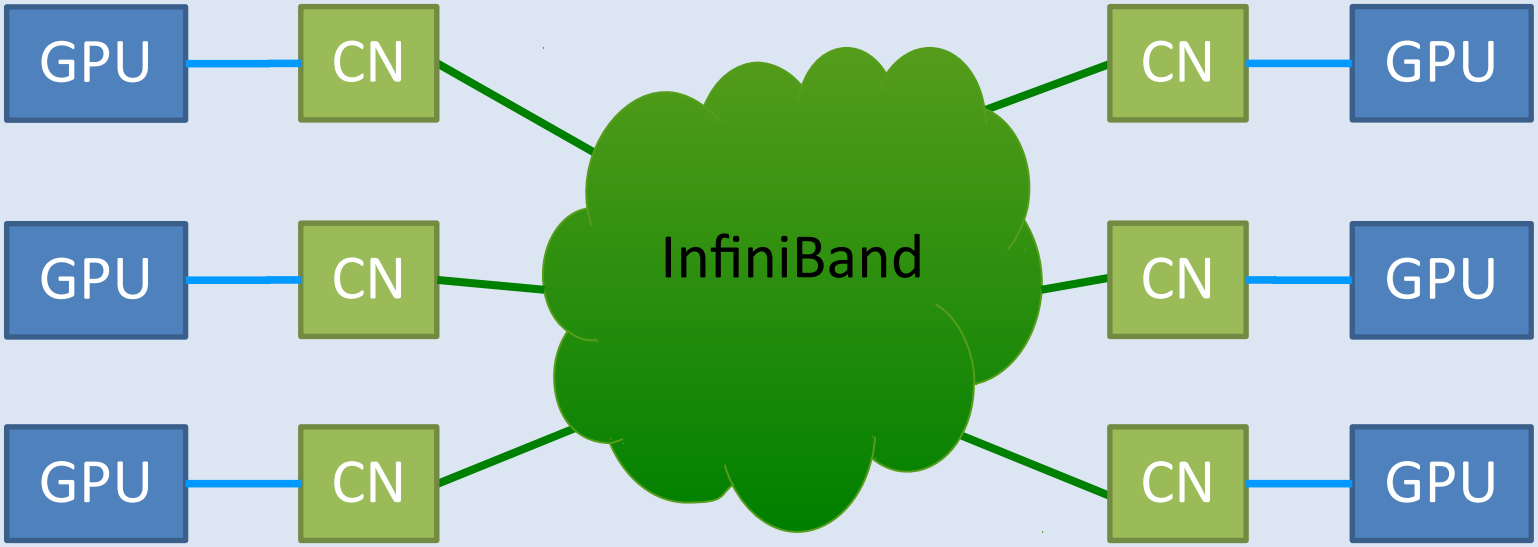
Many core (Intel Xeon Phi)

- 60 to 72 cores, 4 threads / core
- Limited single thread performance
  - *In-order architecture, low frequency*
- Energy efficient ($/FLOP)
- Std. Programming models (MPI, OpenMP, …)
- Might run autonomously (without host)

GPGPU (Graphic cards)

- Designed for graphics but evolved into general purpose
- Hundreds of (weak) computing cores
- Very energy efficient ($/FLOP)
- Spec. programming models (CUDA, OpenCL)
- Requires a host CPU

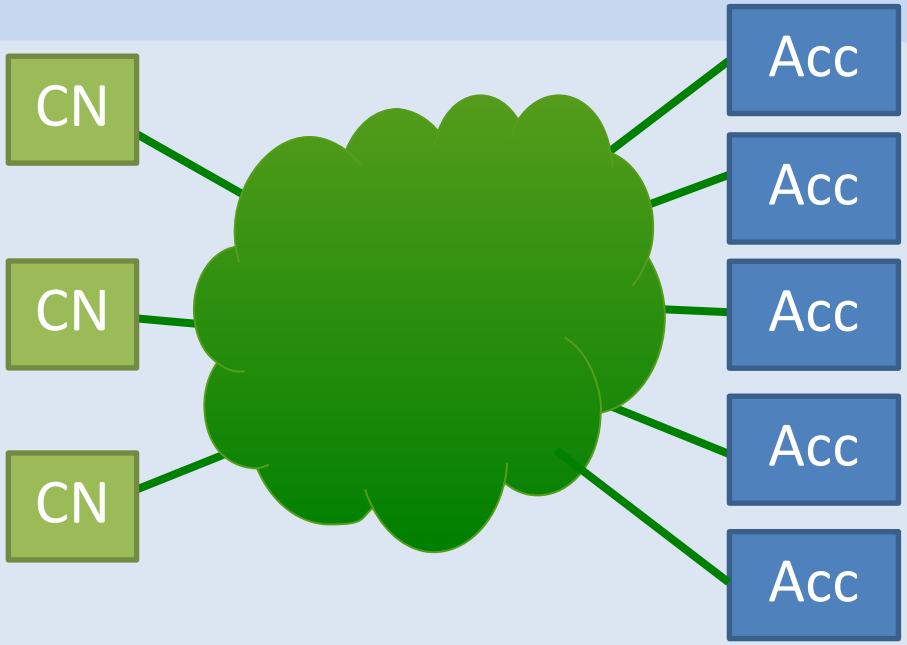
Heterogeneous Clusters



- Flat IB-topology
- Simple management of resources

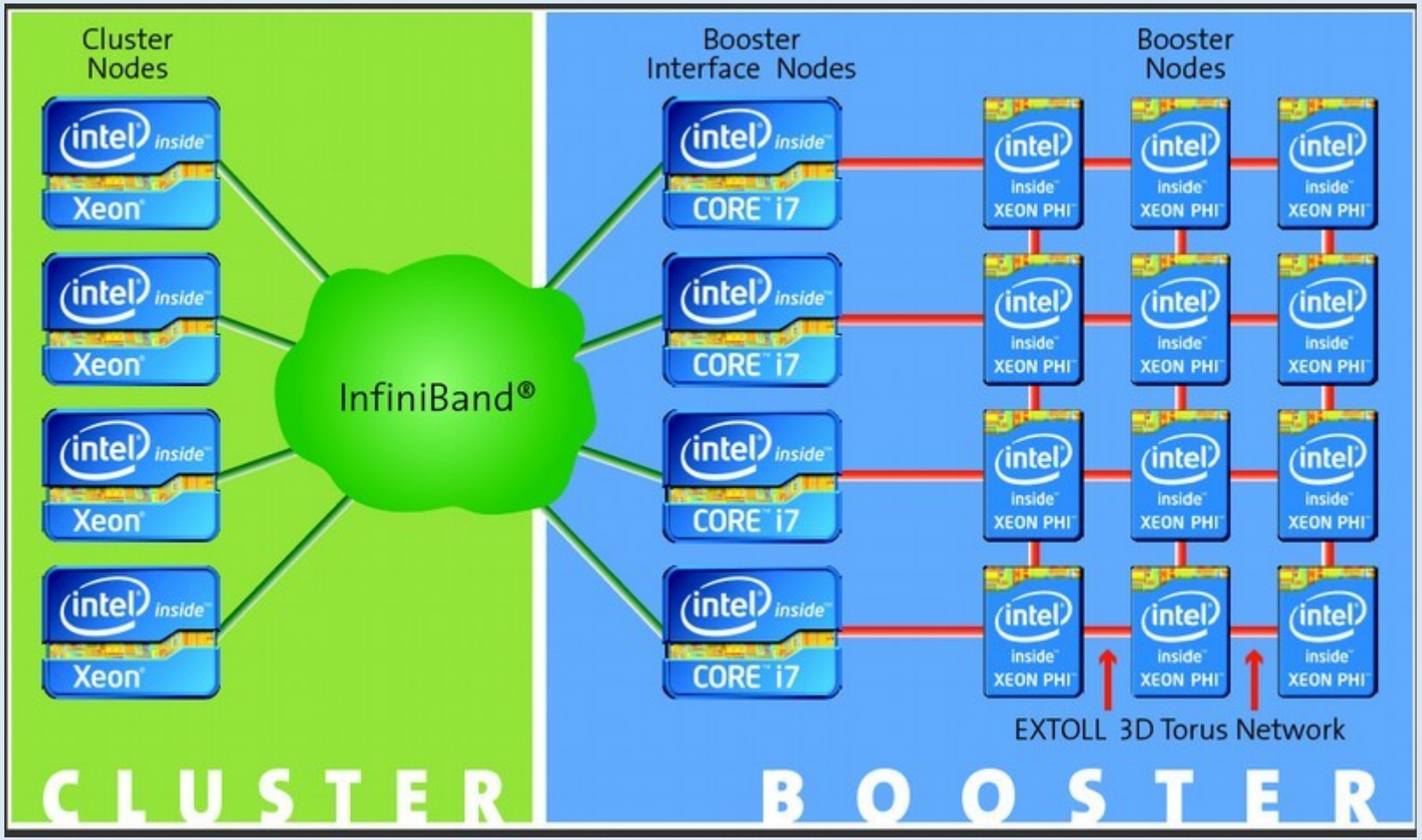
- Static assignment of CPUs to GPUs
- Accelerators not capable to act autonomously

Alternative Integration

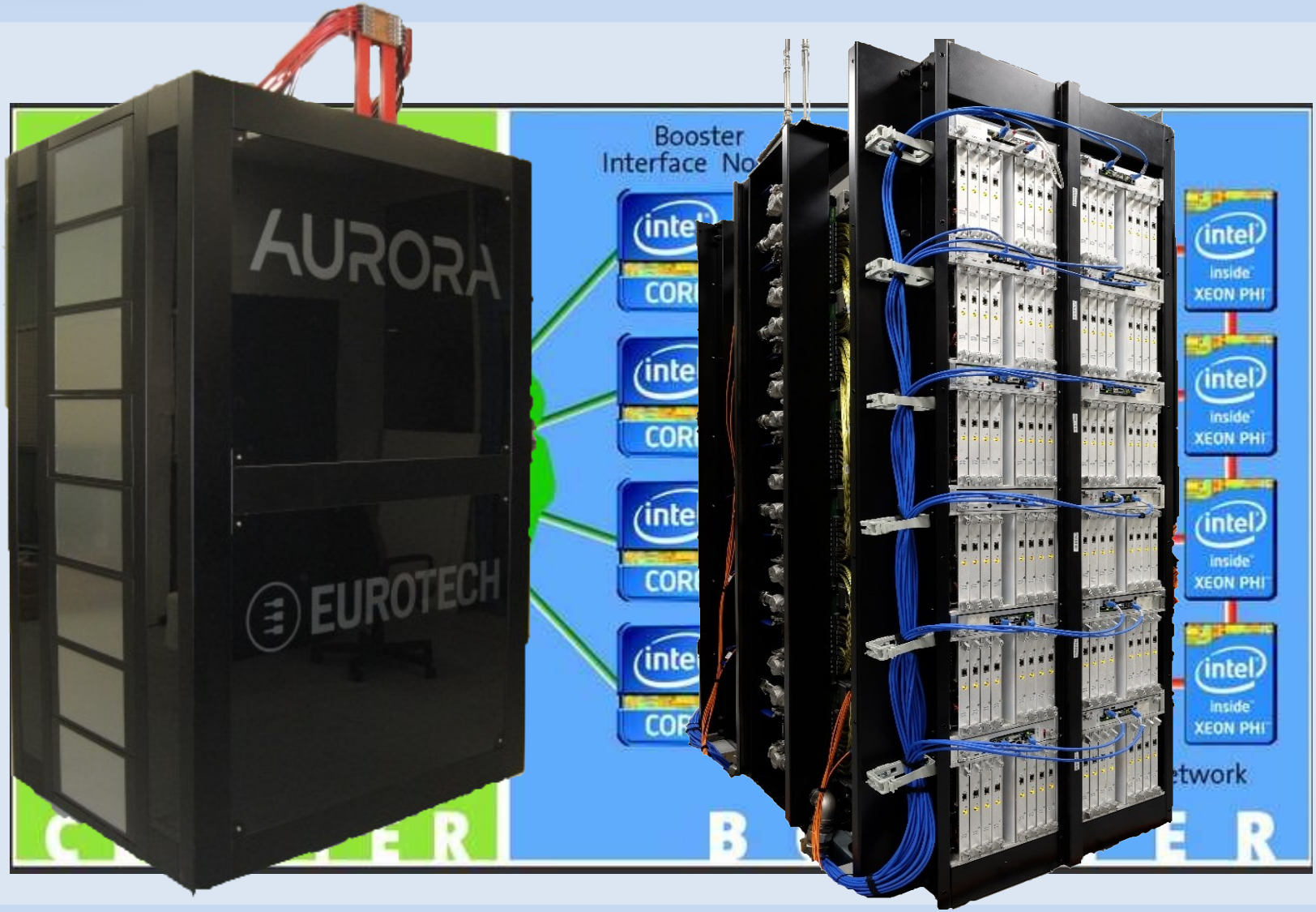


- Go for more capable accelerators (e.g. MIC)
- Attach all nodes to a low-latency fabric
- All nodes might act autonomously
- Dynamical assignment of cluster-nodes and accelerators
  - IB can be assumed as fast as PCIe besides latency
- Ability to off-load more complex (including parallel) kernels
  - communication between CPU and Accelerator less frequently
  - larger messages i.e. less sensitive to latency

Hardware Architecture

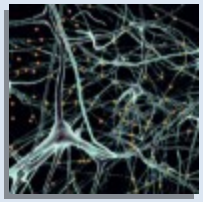


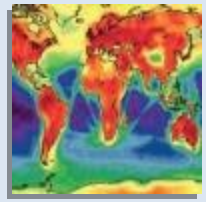
Hardware

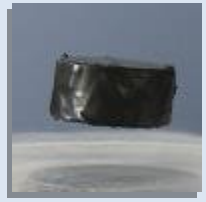


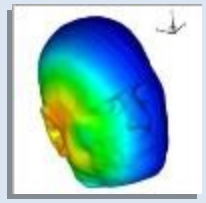
Guiding Applications

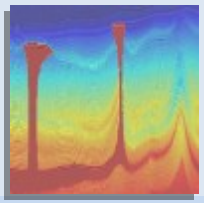
- Brain simulation (EPFL)
- Space weather simulation (KULeuven)
- Climate simulation (CYI)
- Computational fluid engineering (CERFACS)
- High T_c superconductivity (CINECA)
- Seismic imaging (CGG)
- Human exposure to electromagnetic fields (INRIA)
- Geoscience (BADW-LRZ)
- Radio astronomy (Astron)
- Oil exploration (BSC)
- Lattice QCD (UREG)



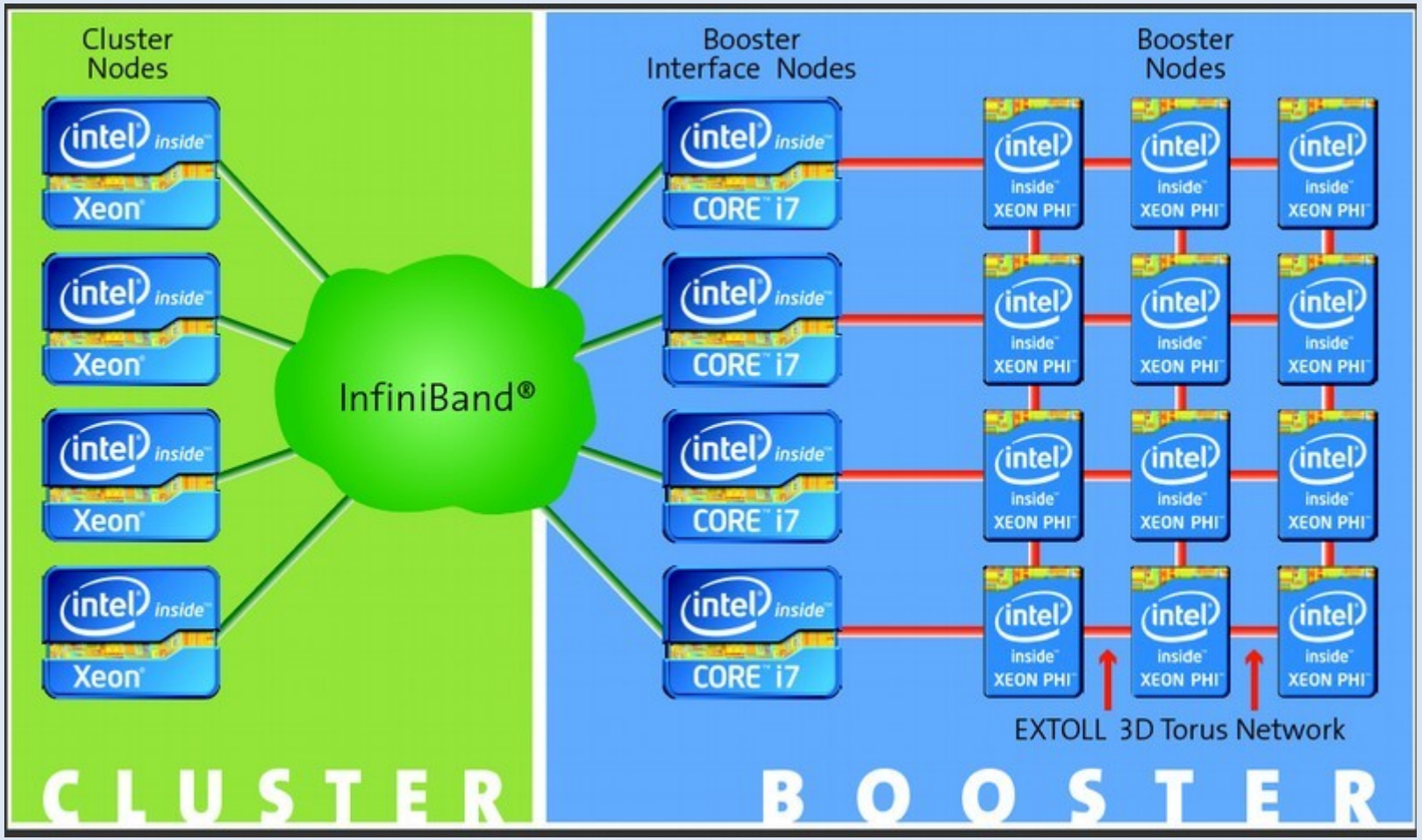




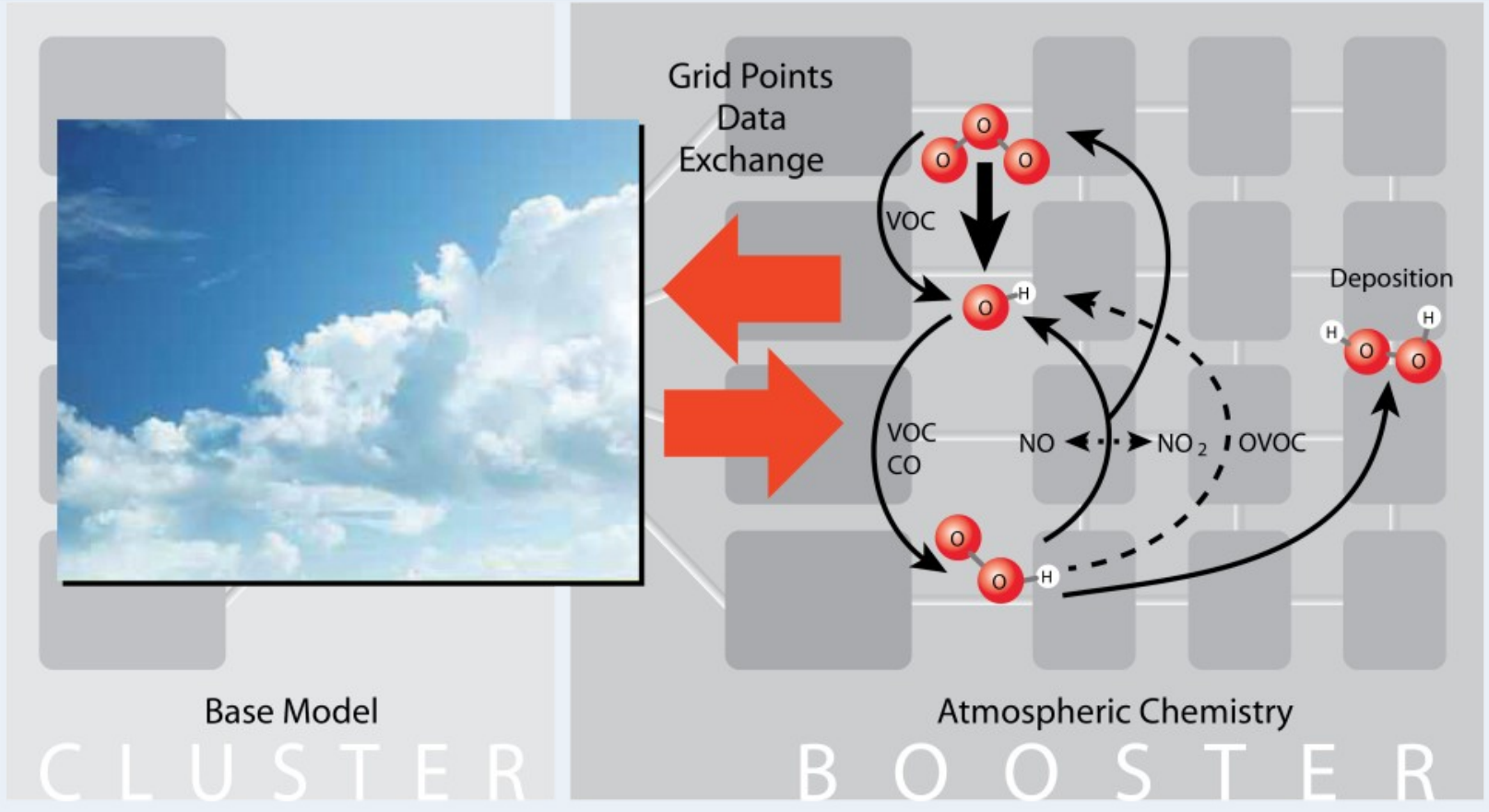


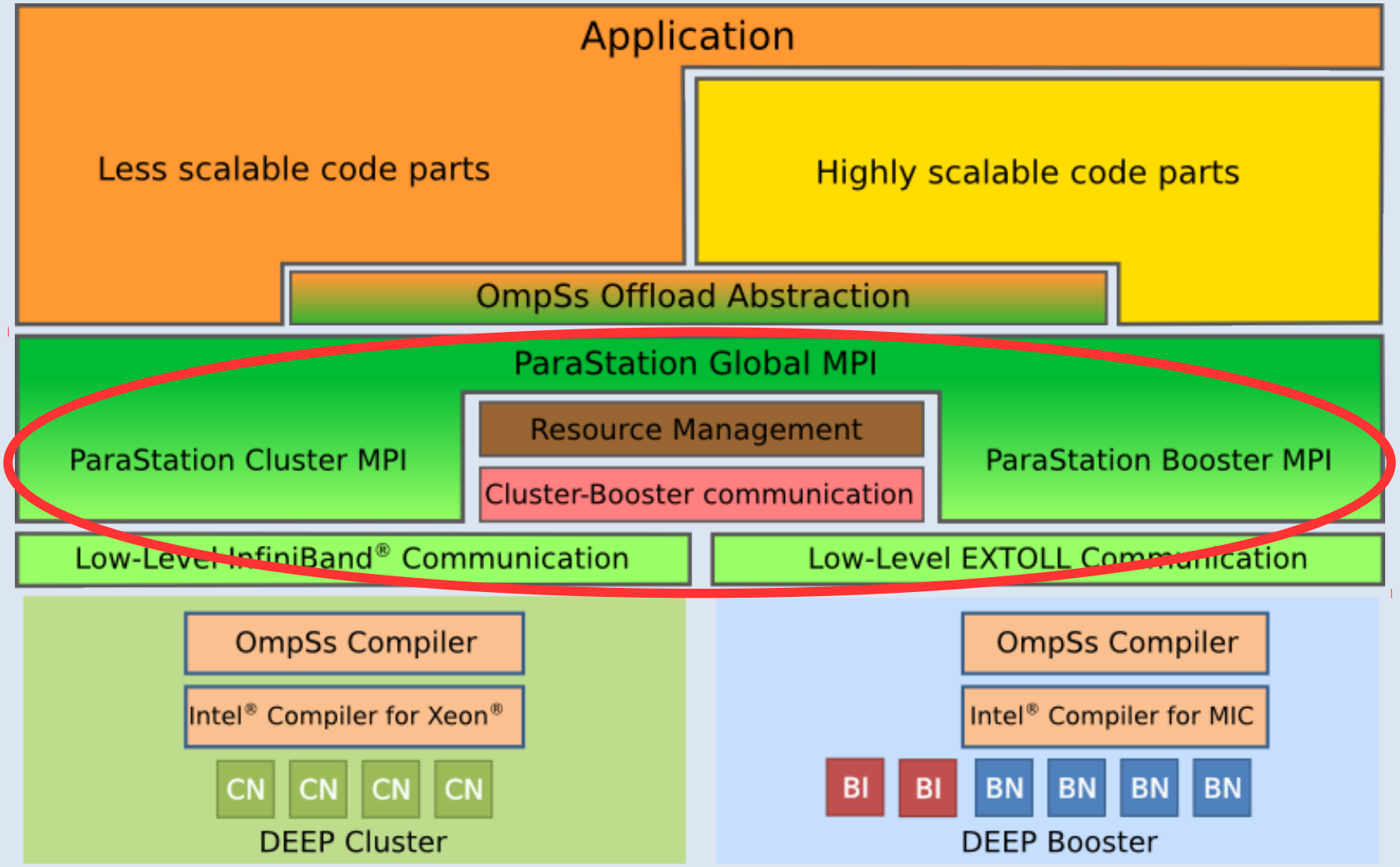

Hardware Architecture



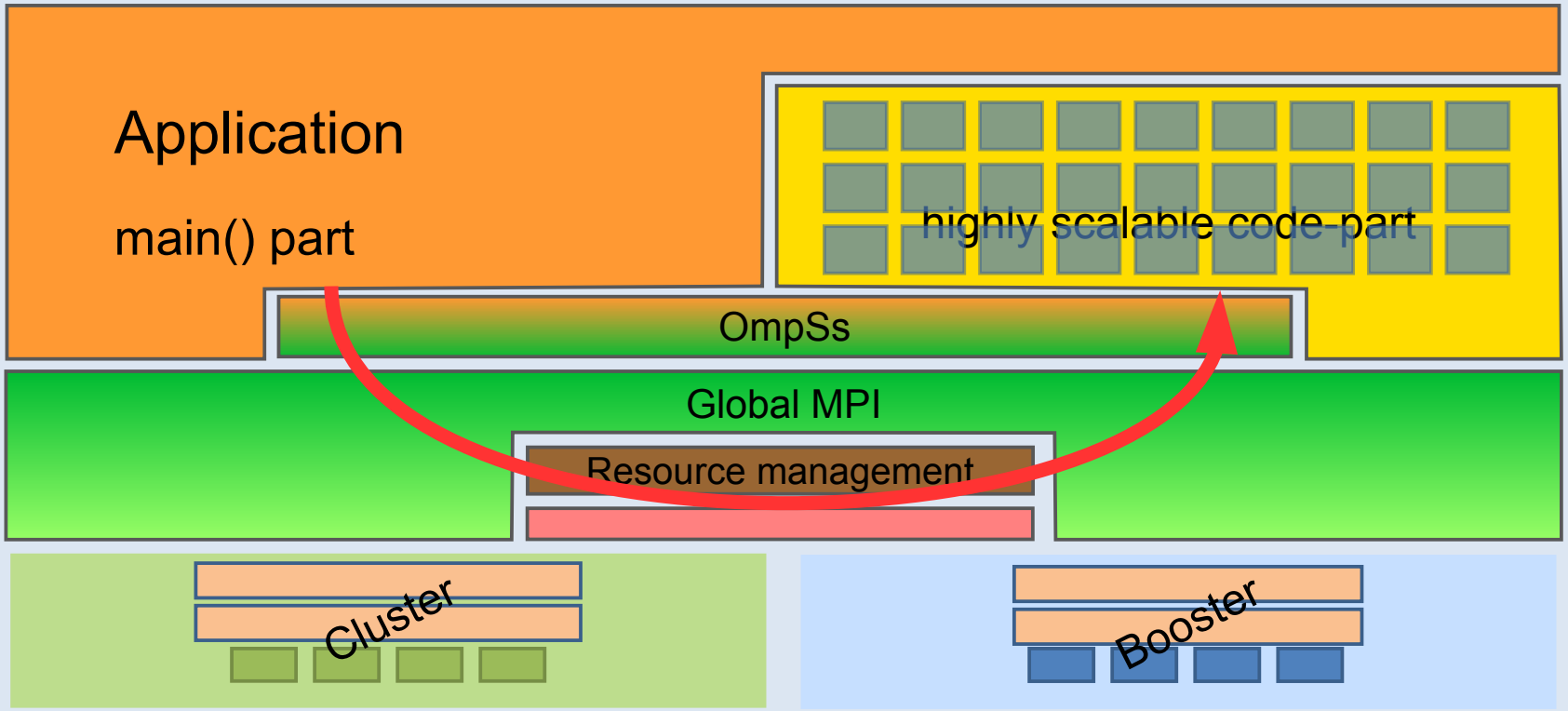
CyI – Climate Simulation



Software Architecture



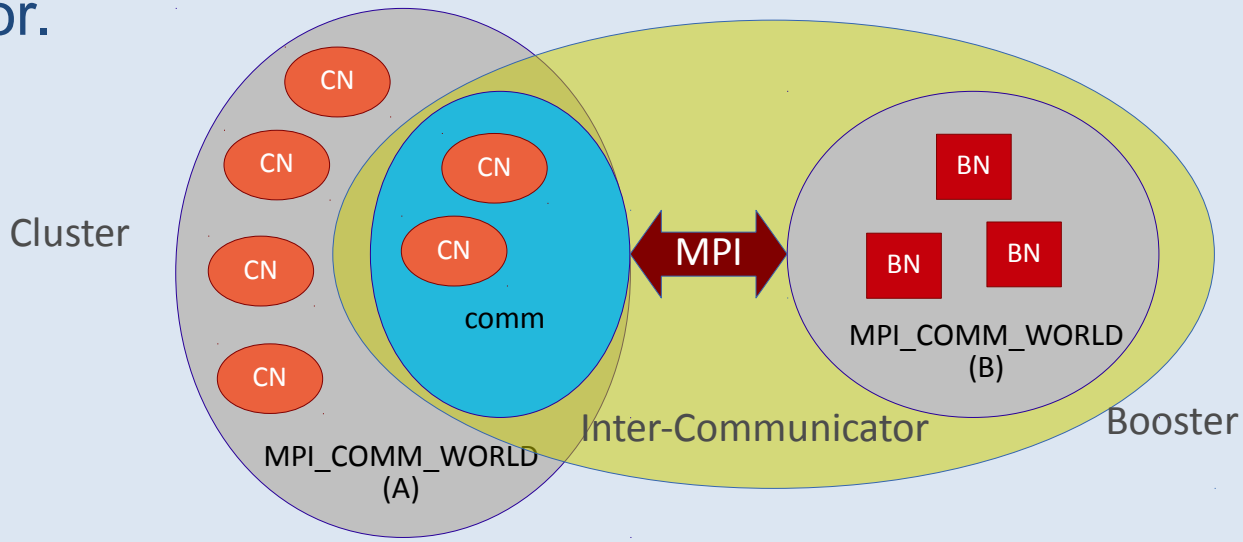
Application Startup



- Application's main()-part runs on Cluster-nodes (CN) only
- Actual spawn done via global MPI
- OmpSs acts as an abstraction layer
- Spawn is a collective operation of Cluster-processes
- Highly scalable code-parts (HSCP) utilize multiple Booster-nodes (BN)

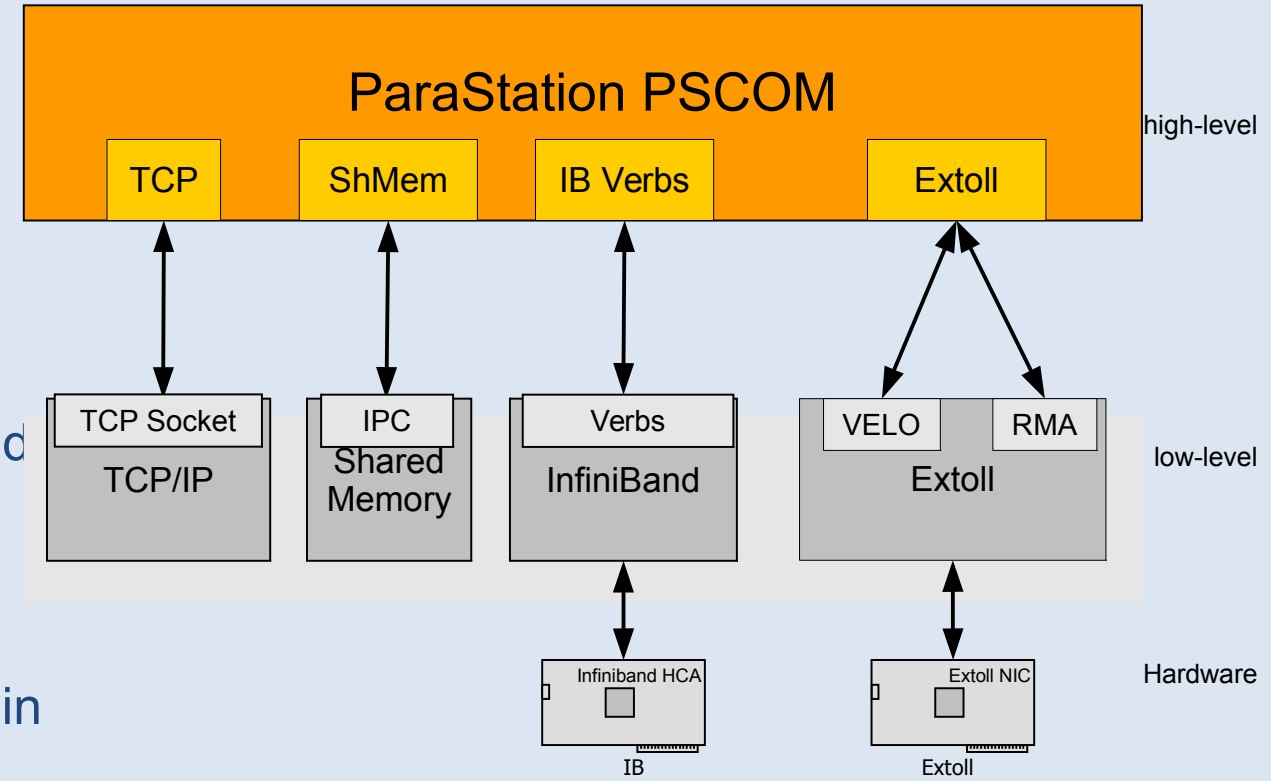
MPI Process Creation

- The inter-communicator contains all parents on the one side and all children on the other side.
  - Returned by MPI_Comm_spawn for the parents
  - Returned by MPI_Get_parent by the children
- Rank numbers are the same as in the the corresponding intra-communicator.

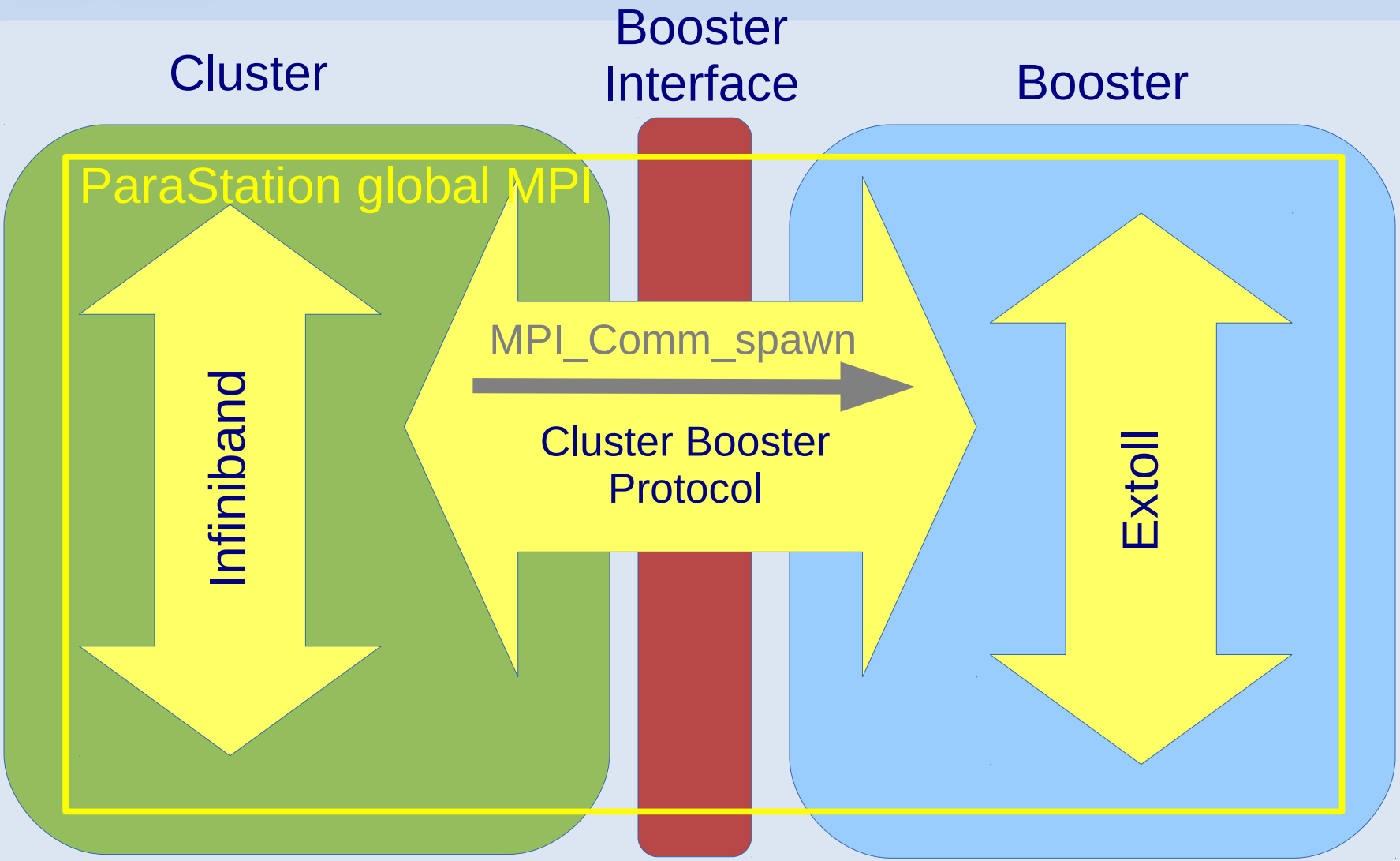


ParaStation pscom V5

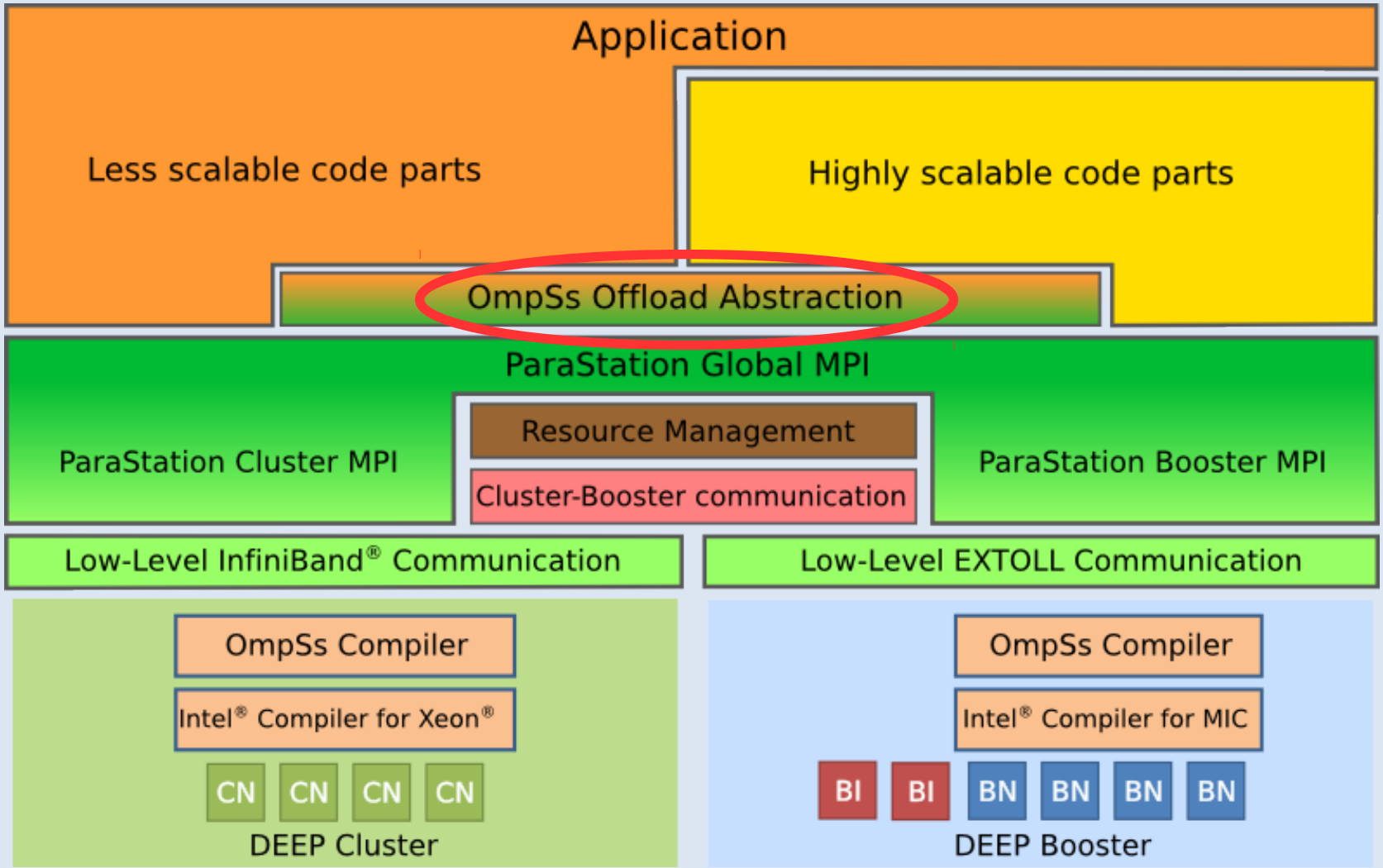
- Unified comm layer
- pscom plugins
  - Modular
  - Flexible to extend
  - Verbs plugin
  - VELO/RMA plugin
  - Easy enabling of Cluster-Booster protocol



Programming



Software Architecture



OmpSs Offload Abstraction

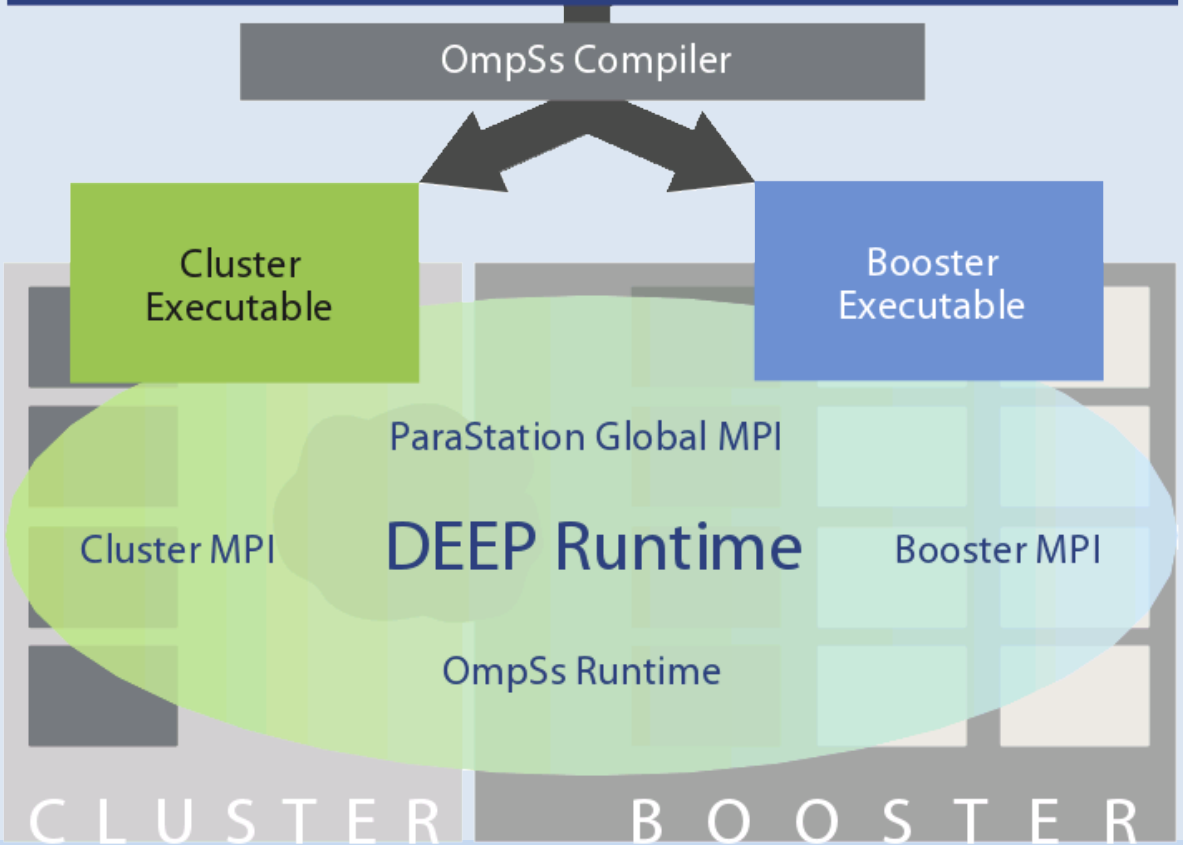
Source Code

Compiler

Application Binaries

DEEP Runtime

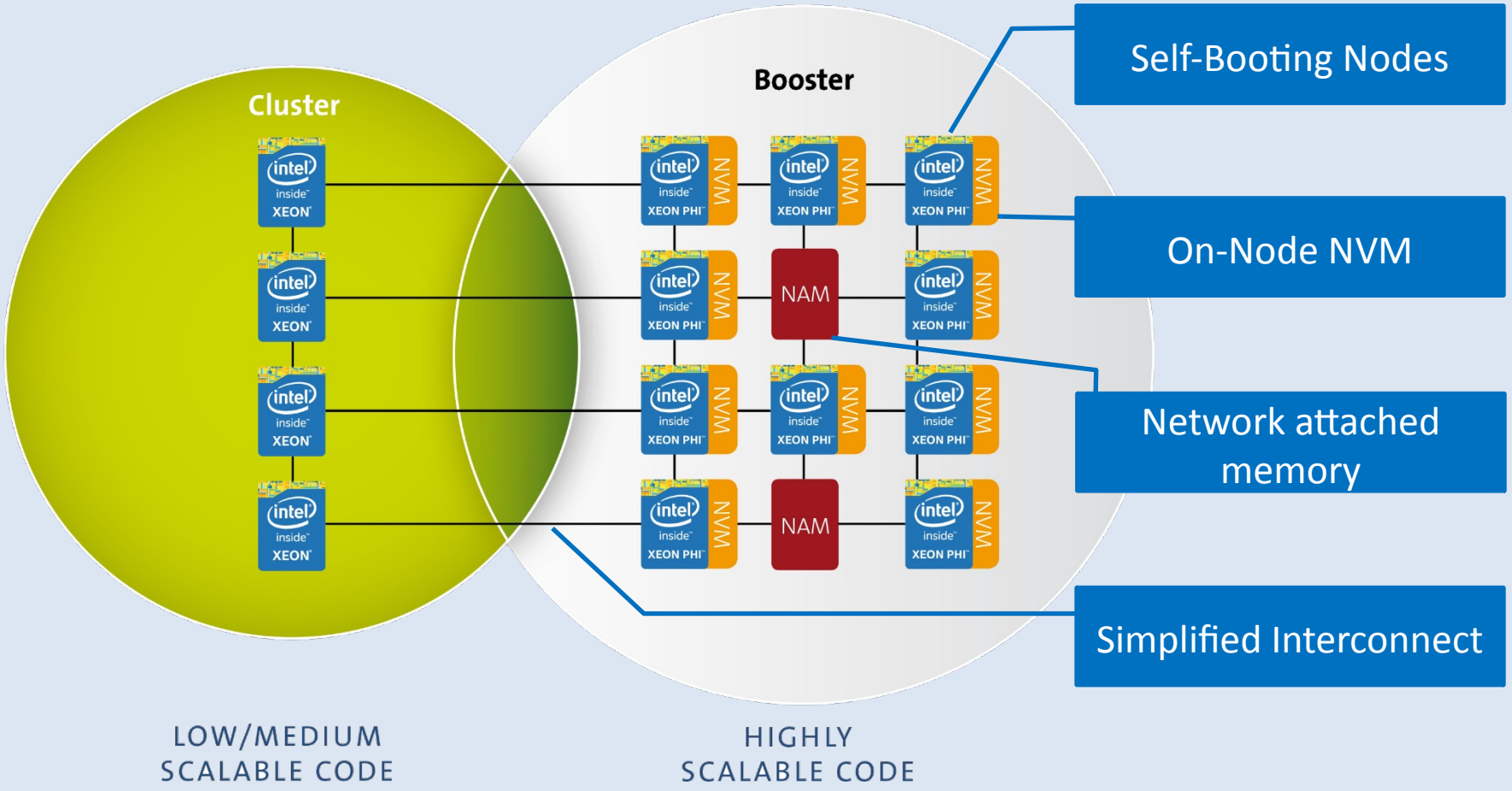
```
int main(int argc, char *argv[]){
    /*...*/
    for(int i=0; i<3; i++){
        #pragma target device (comm:size*rank+i) copy_deps
        #pragma omp task input(…) output(…)
        foo_mpi(i, …);}}
```



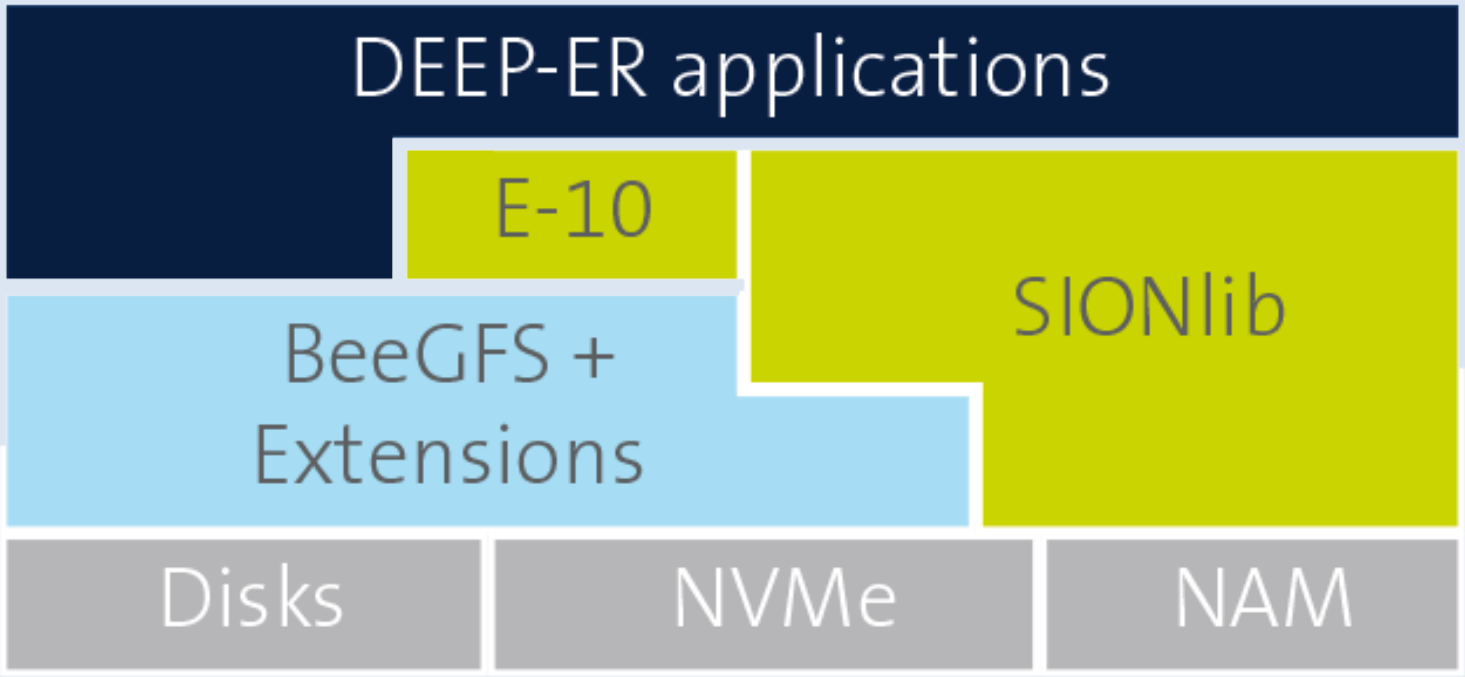
DEEP Advantages

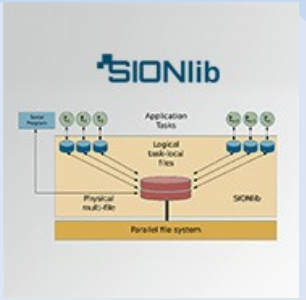
- DEEP is **more flexible** than a standard architecture → enables different usage models:
  1. Dynamic ratio of processors/coprocessors
  2. Use Booster as pool of accelerators (globally shared)
  3. Discrete use of the Booster
  4. Discrete use + I/O offload
  5. Specialized symmetric mode
- These usage models enable a **more efficient** use of system resources
  - But might require non-trivial changes on the application design mindset

Architecture Innovation



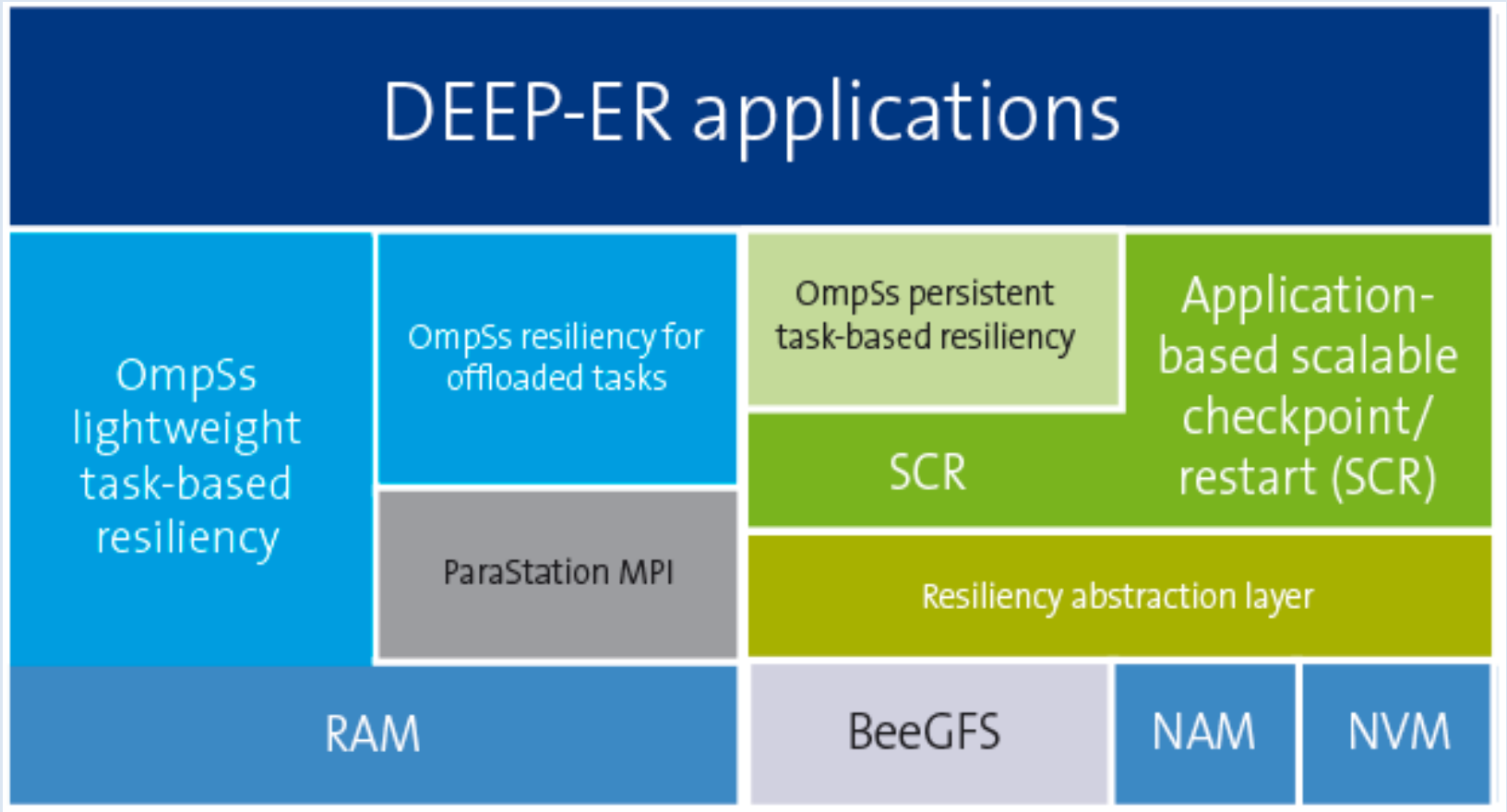
Scalable I/O



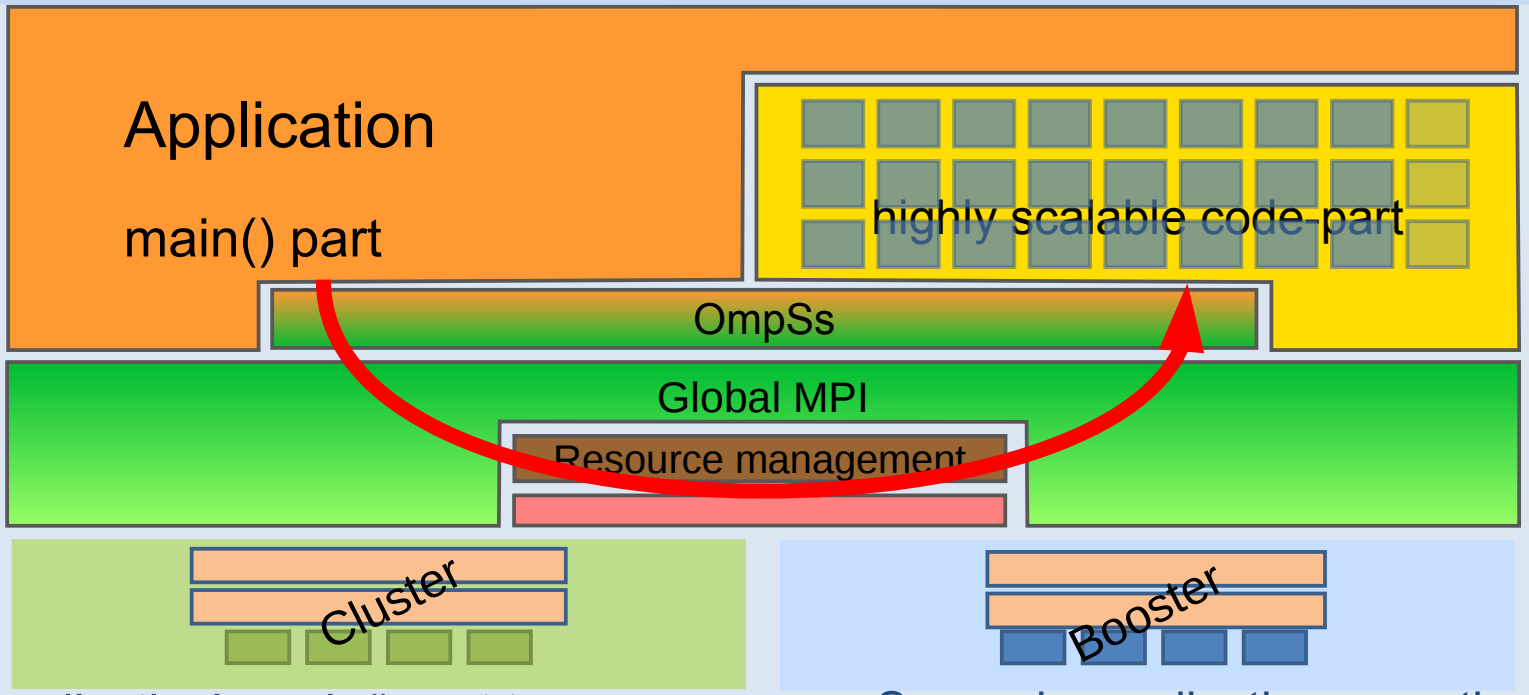


- Improve I/O scalability on all usage-levels
- Used also for checkpointing

Resiliency features



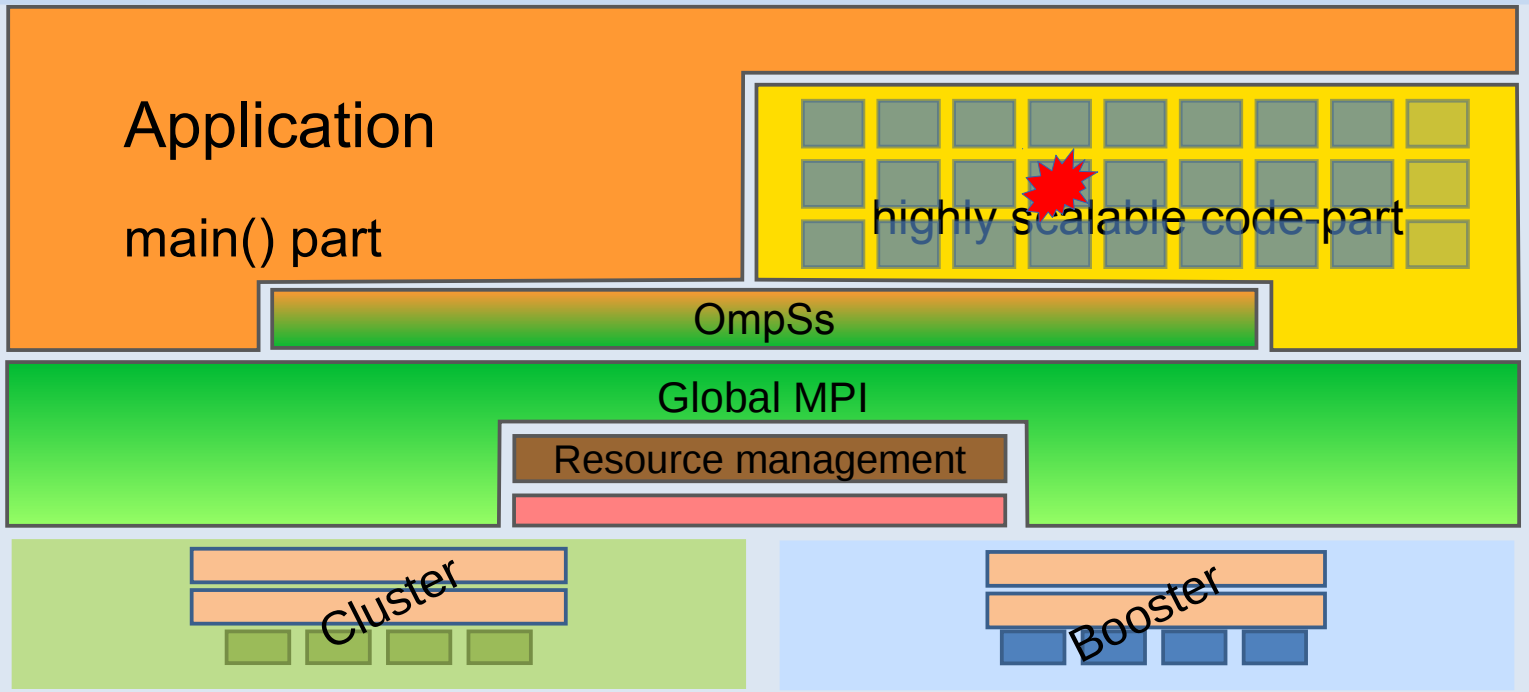
Startup in detail



- Application's main()-part to run on Cluster-nodes (CN) only
- Resources might be managed statically or dynamically
- OmpSs acts as an abstraction layer
- Actual spawn done via global MPI

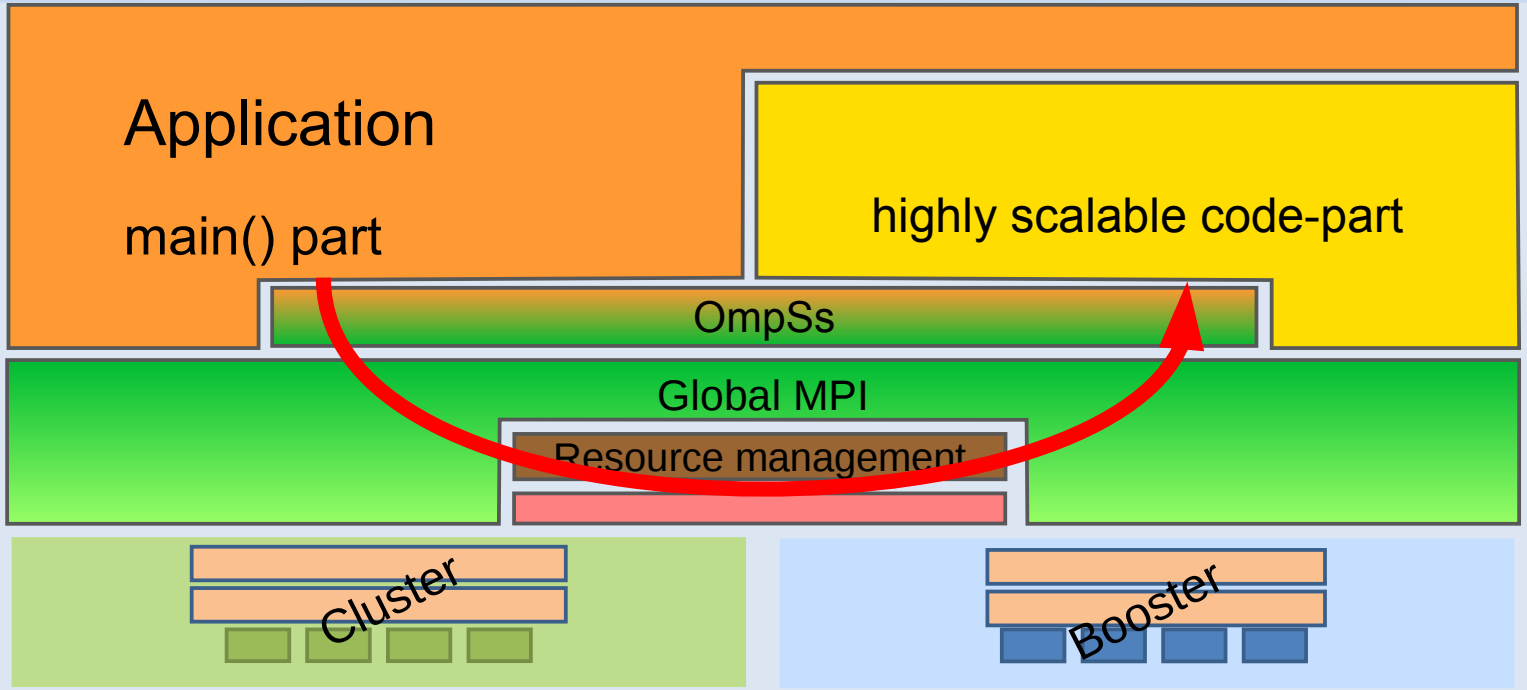
- Spawn is a collective operation of Cluster-processes
- Highly scalable code-parts (HSCP) utilize multiple Booster-nodes (BN)

Startup and failure



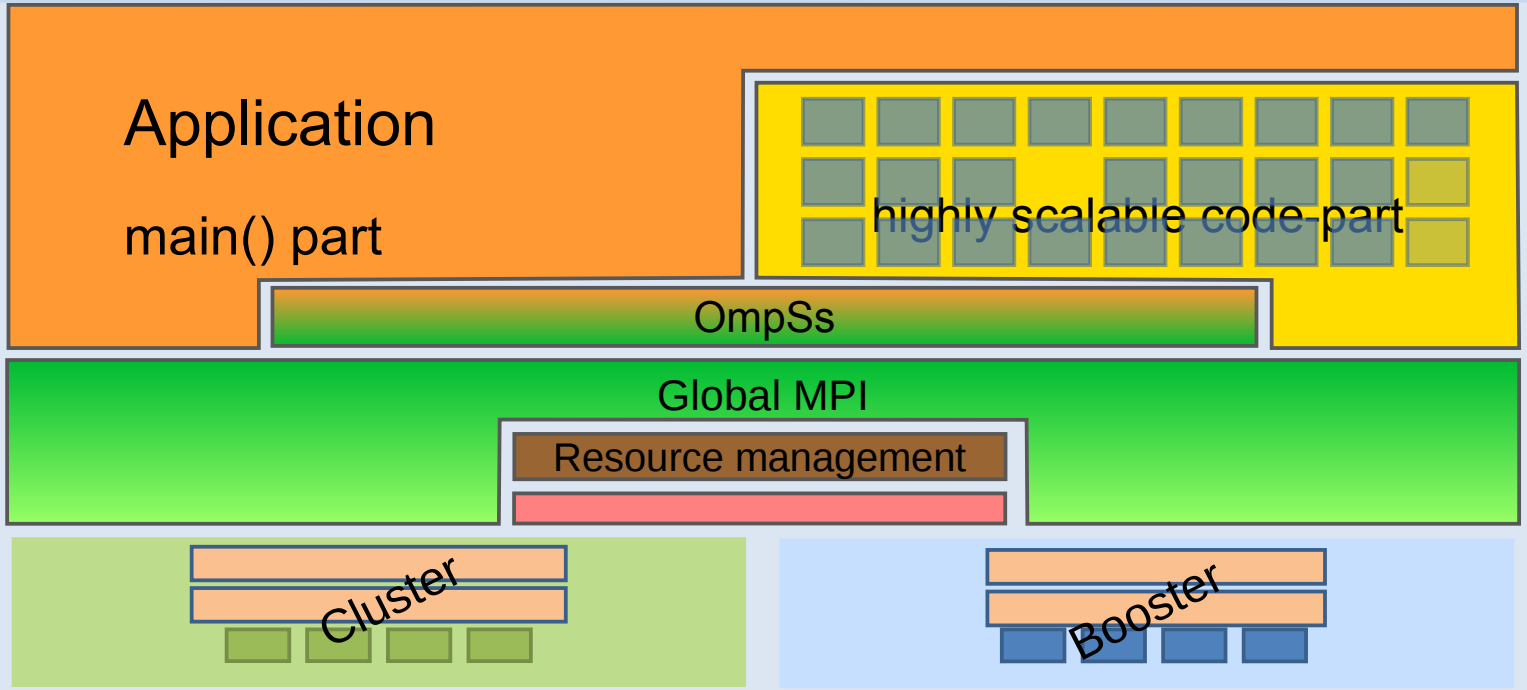
- HSCP might not make use of all resource on the Booster
  - Some spares might be held back to recover from failure
  - Depends on applications flexibility
- BN fails → HSCP not to survive
  - Booster-MPI to cleanup HSCP's resources
  - Global-MPI to ensure main()-parts survival

Restart



- main()-part is signaled by MPI on HSCP-failure
  - Re-start HSCP
  - Use spare-resources or rely on dynamic allocation
- Abstraction via OmpSs helps
  - Can keep track on how to start HSCP
  - Has control on HSCP's status

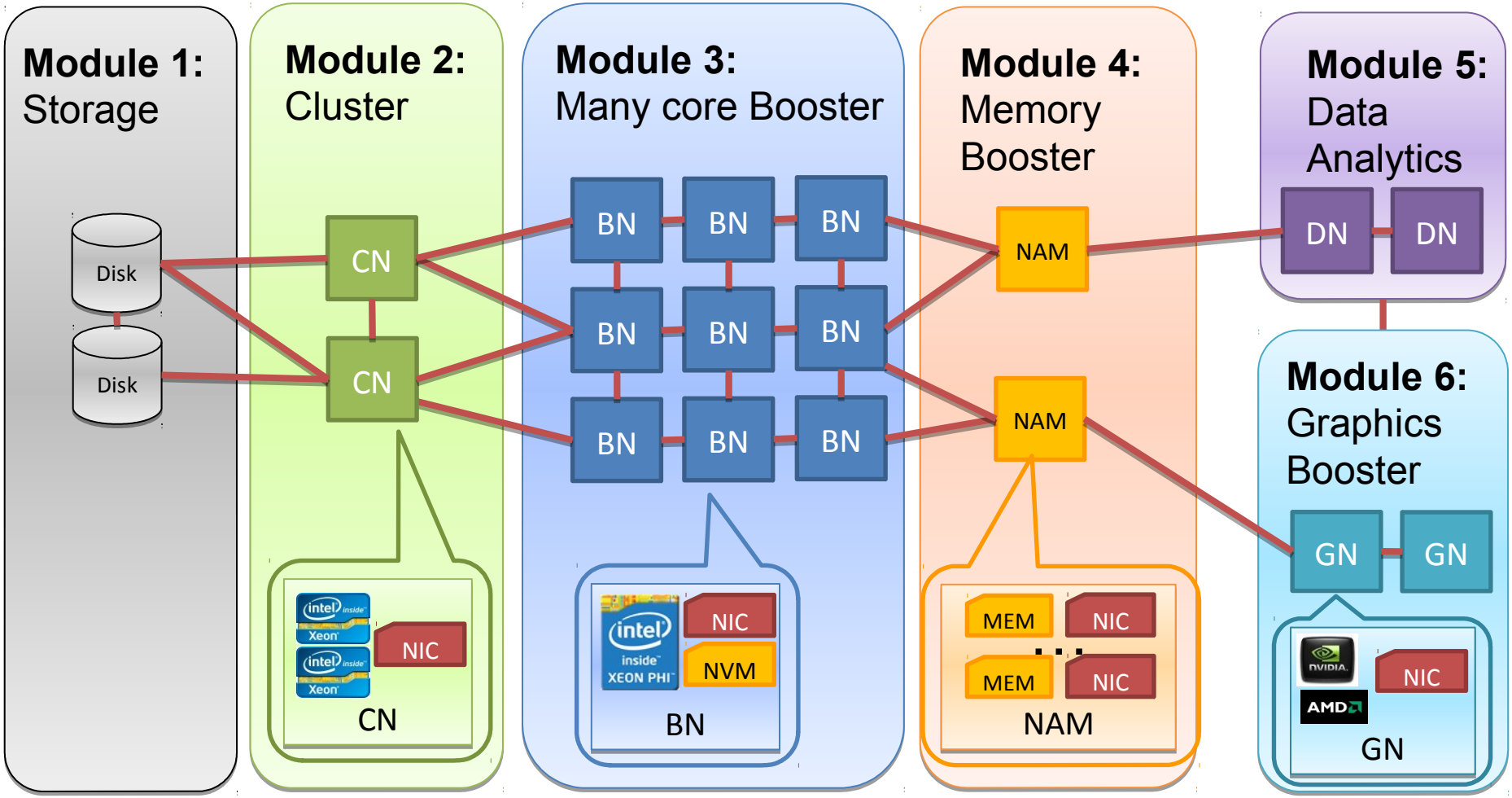
Restart



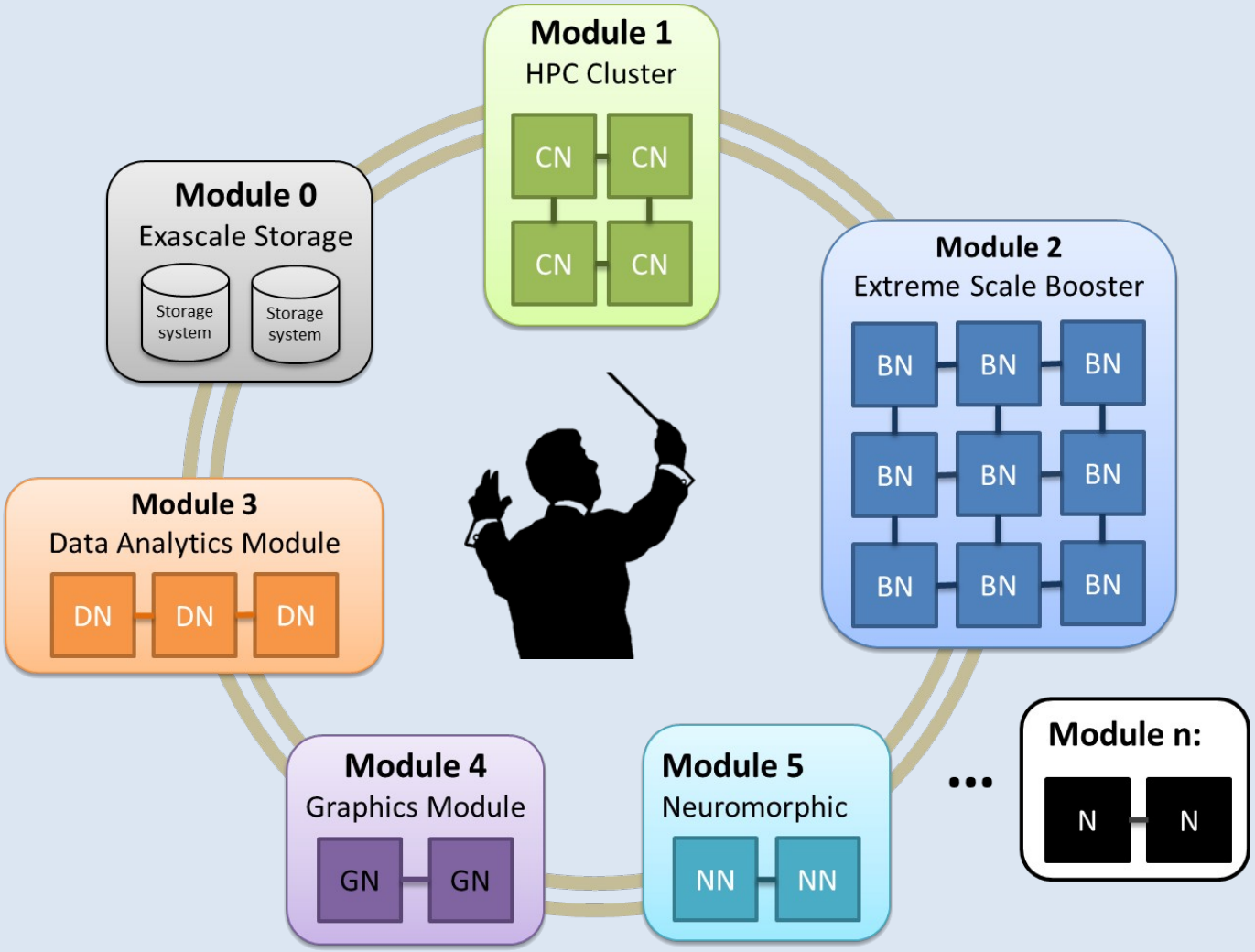
- HSCP now on diff. set of BNs
- Run-time of last HSCP is lost
  - App's main()-part to survive
  - Keep track of all info required for re-start
- OmpSs-abstraction helps again
  - Input-parameters are known
  - Has control on HSCP's output
  - Special handling of in-out parameters required

Modular Supercomputing

Generalization of the Cluster-Booster concept

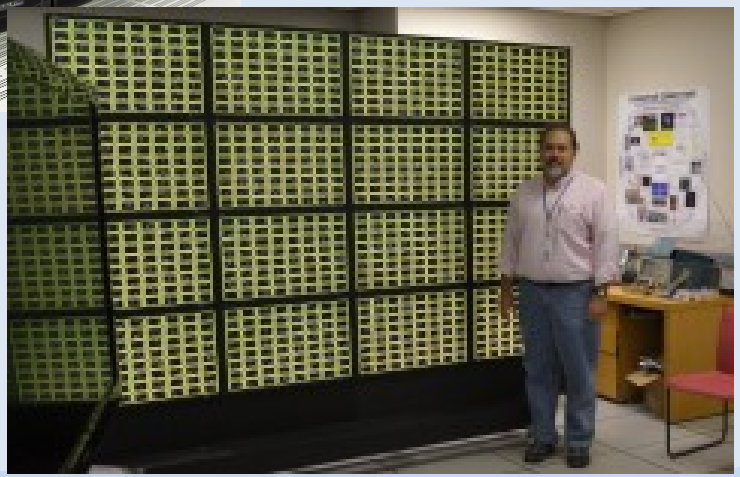


Towards Modular Computing




D:wave


JURE



Take aways

- DEEP and DEEP-ER explore new ways to use and manage heterogeneity
  - Cluster Booster Architecture and it's implementation
  - Programming Model
- Both projects are application driven

- DEEP-ER extends the concept towards I/O, Resiliency and innovative memory technologies
- Jülich plans to extend its JURECA Cluster by a 10 PF/s Booster in 2017 (based on KNL)
- More Modules to come...
- More info: http://www.deep-project.eu
  http://www.deep-er.eu