Running DEEP

Operating Heterogeneous Clusters in the Many-Core Era

♦

Norbert Eicker

Jülich Supercomputing Centre & University Wuppertal

ROME Workshop

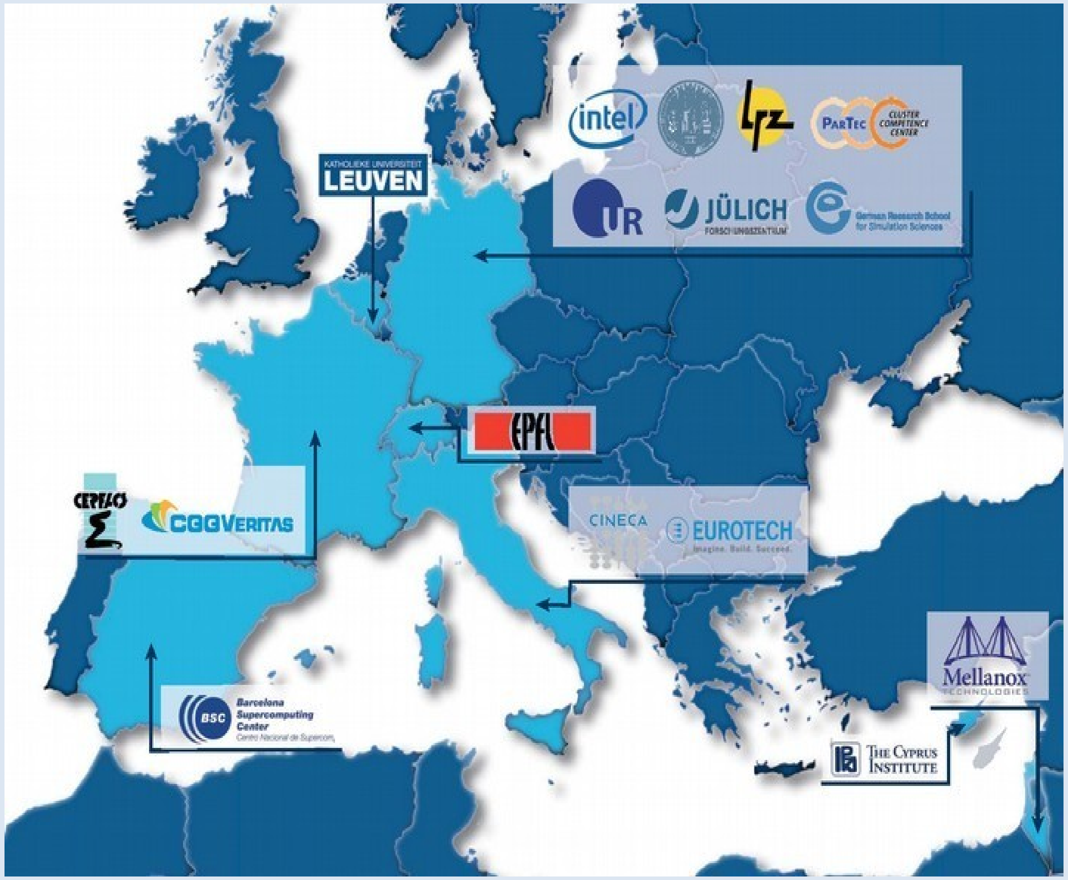
26th August 2014

Porto

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

DEEP Project

- 16 Partners
  - 4 PRACE hosts
  - 3 Research Centers
  - 5 Industry Partners
  - 4 Universities
  - Coordinator JSC
- 8 Countries
- Duration: 3.5 years
- Budget: 18.3 M€
- EU funding: 8.03 M€



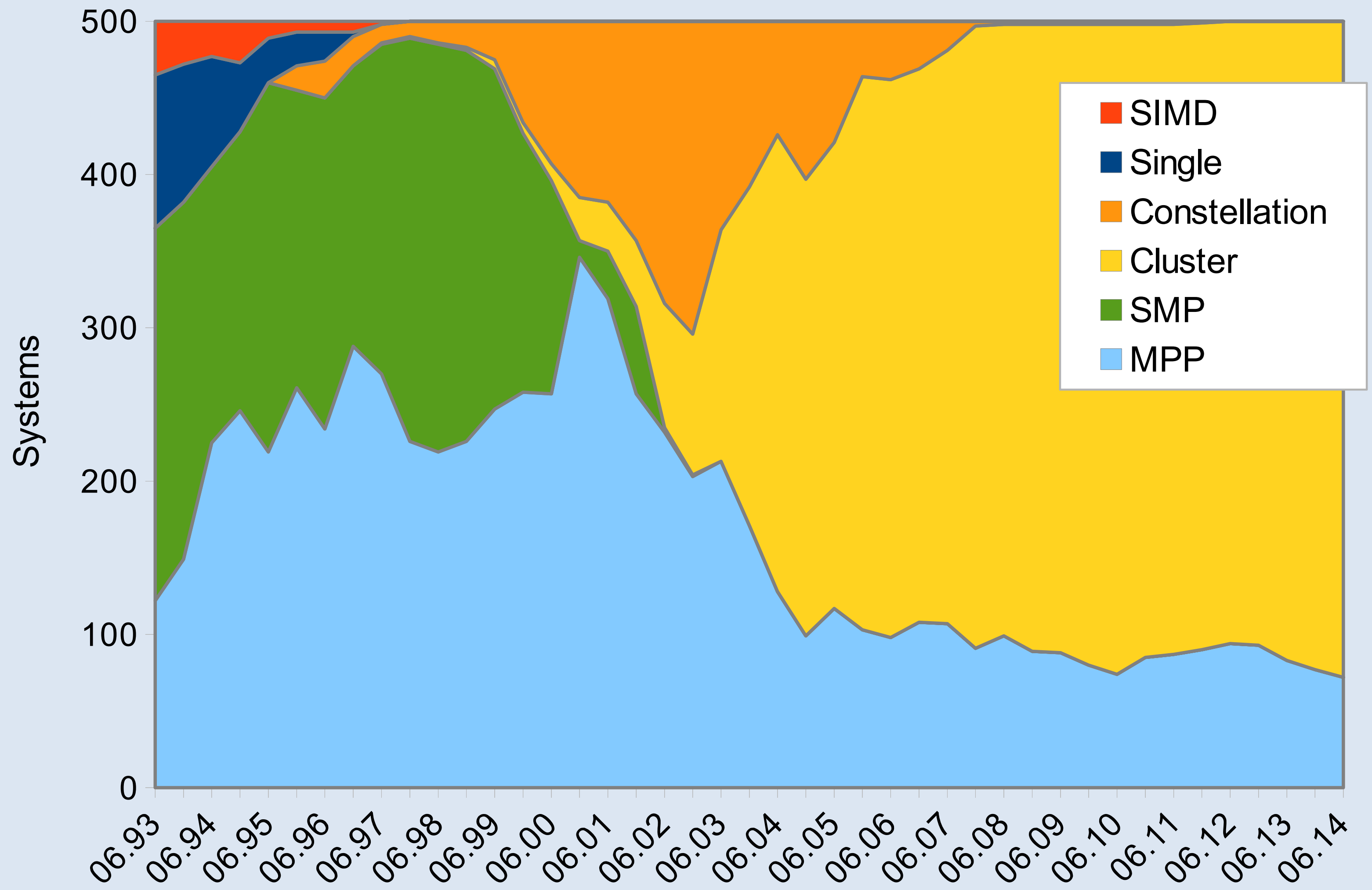
Outline

- Motivation
  - Why heterogeneous systems
  - How to organize heterogeneity
- DEEP
  - General concept
  - Hardware architecture
  - Software stack
  - Programming paradigm
- Summary

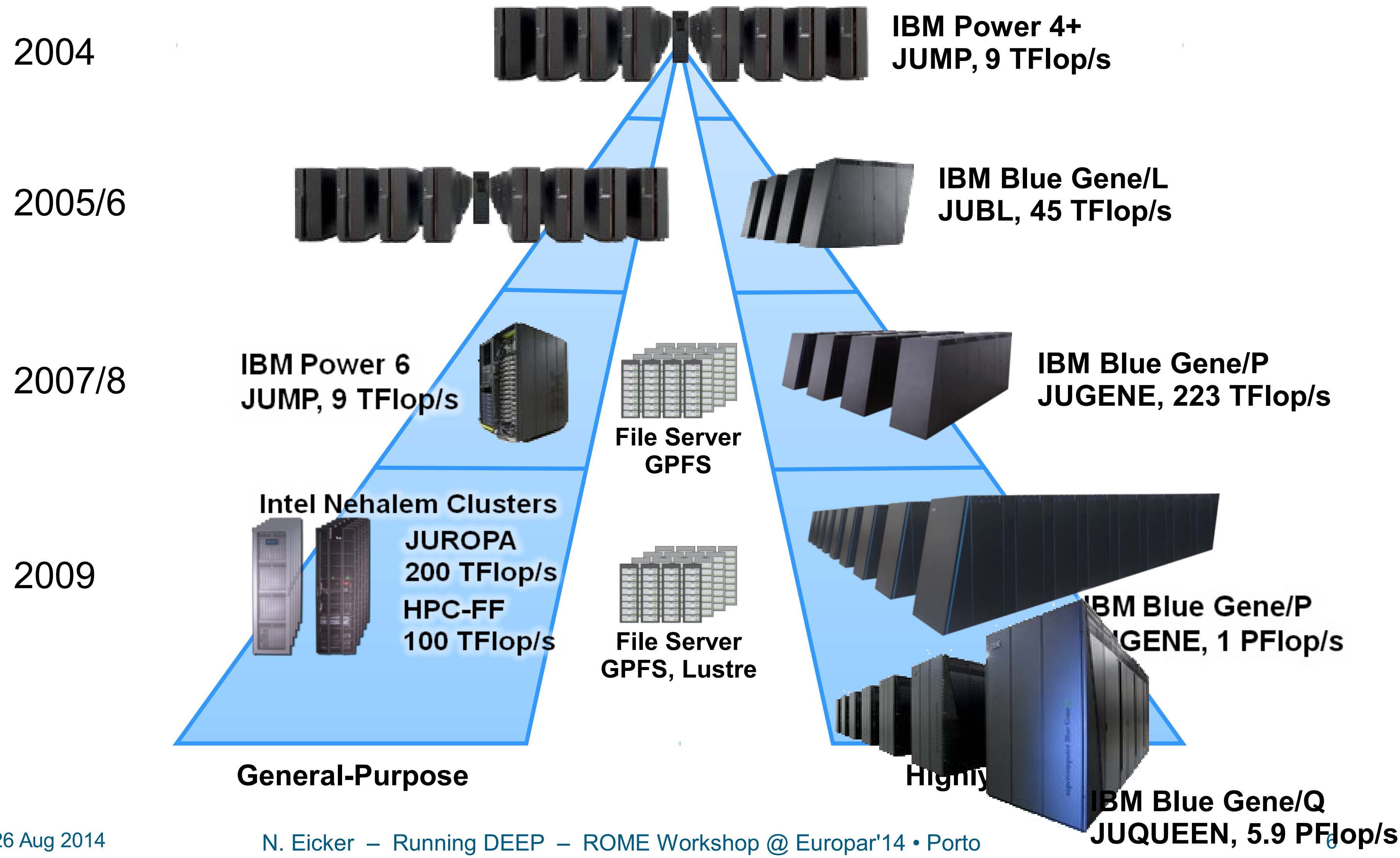
DEEP face-to-face Leuven

HPC today

TOP 500 – Architectures



Supercomputers @ JSC



Application's Scalability

Only few application capable to scale to O(450k) cores

- Sparse matrix-vector codes
- Highly regular communication patterns
- Well suited for BG/P

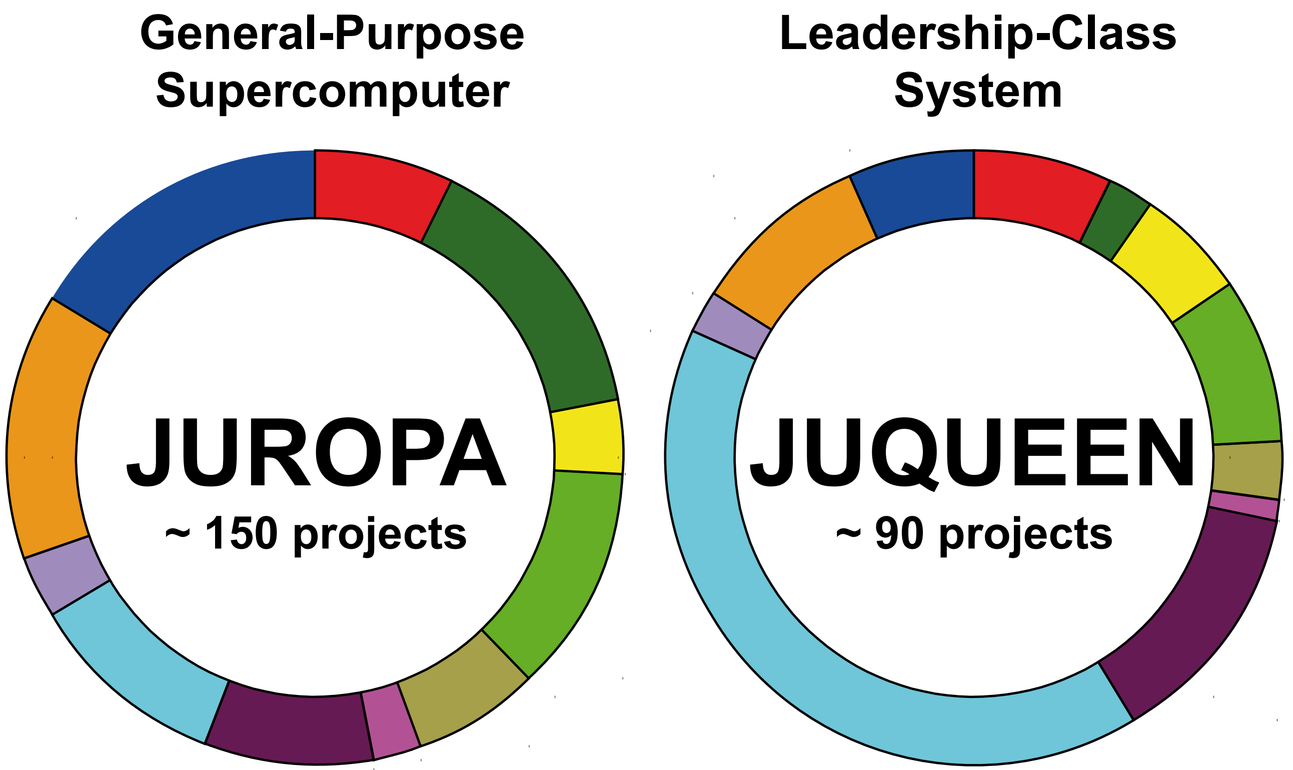
Most applications have more complex kernels

- Less regular control flow and memory access
- Complicated communication patterns
- Less capable to exploit accelerators

In fact:

- Highly scalable apps dominated by highly scalable kernels
- Less scalable apps dominated by less scalable kernels

Research Fields of Current National Projects



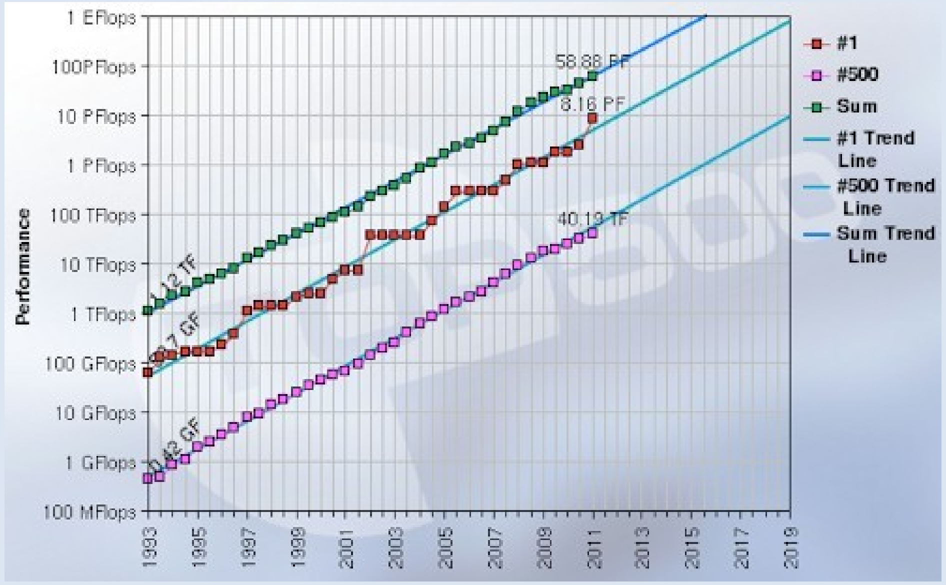
Granting periods
05/2013 – 04/2014
11/2013 – 10/2015

🟥	Astrophysics	🟩	Earth & Environment	🟦	Elementary Particle Physics
🟩	Biophysics	🟫	Plasma Physics	🟪	Computer Science
🟨	Chemistry	🟪	Soft Matter	🟧	Condensed Matter
		🟪	Fluid Dynamics	🟦	Material Science

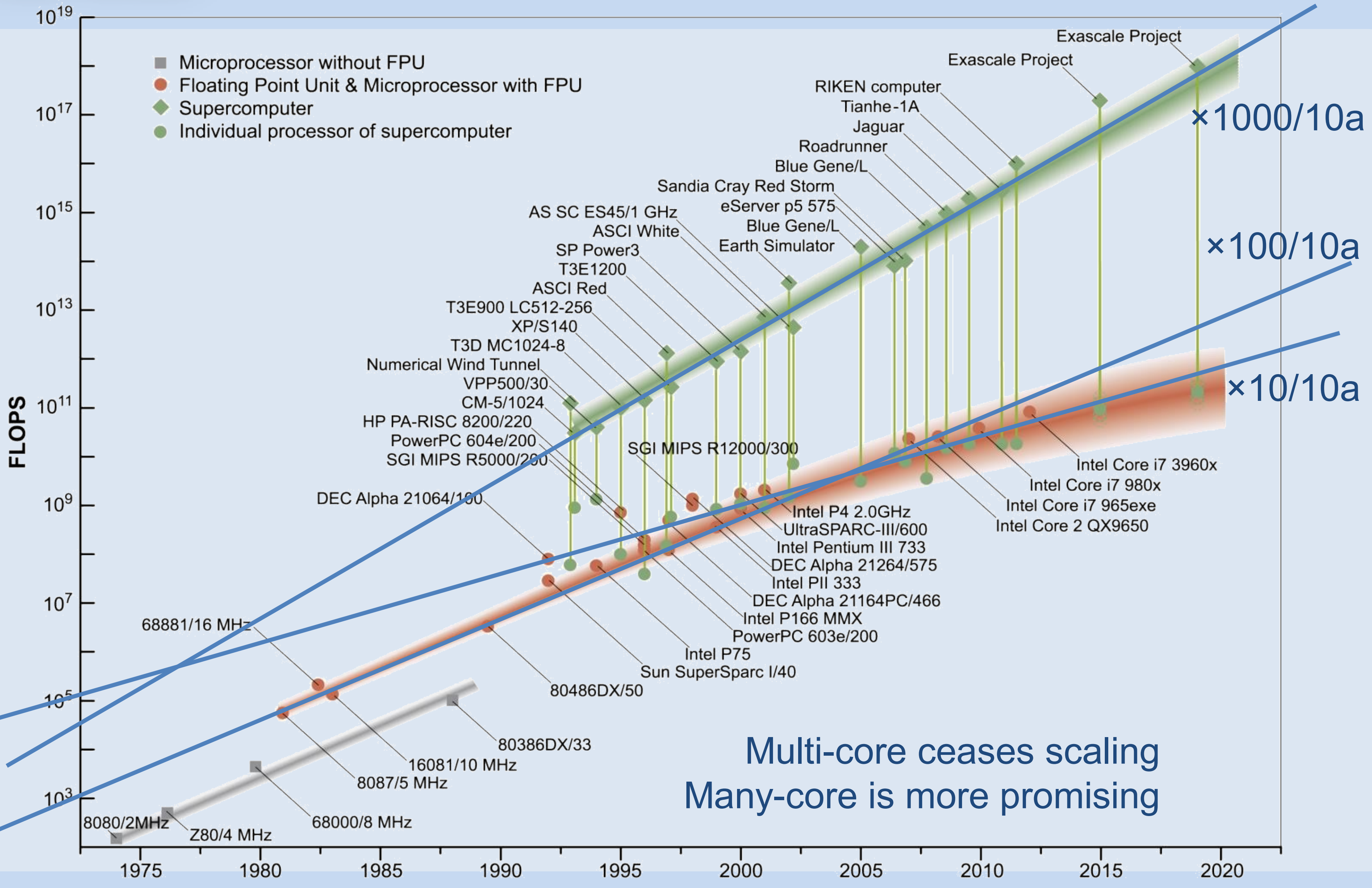
Exascale

ExaScale Systems and Challenges

- (Multi-)PetaFlop (10^{15}) systeme up and running
  - Tianhe-2, Titan, Sequoia, Kei, BlueWaters, etc.
- Meuer's “law” held for 3 decades
  - Each scale (×1000): ~10 y
- Challenges of the next step:
  - ExaFlop (10^{18}) – ca. 2020
  - Processors
  - Energy consumption
  - Reliability / Resiliency
  - Applications
    - Programming models / -paradigms
    - Are today's algorithms still sufficient?



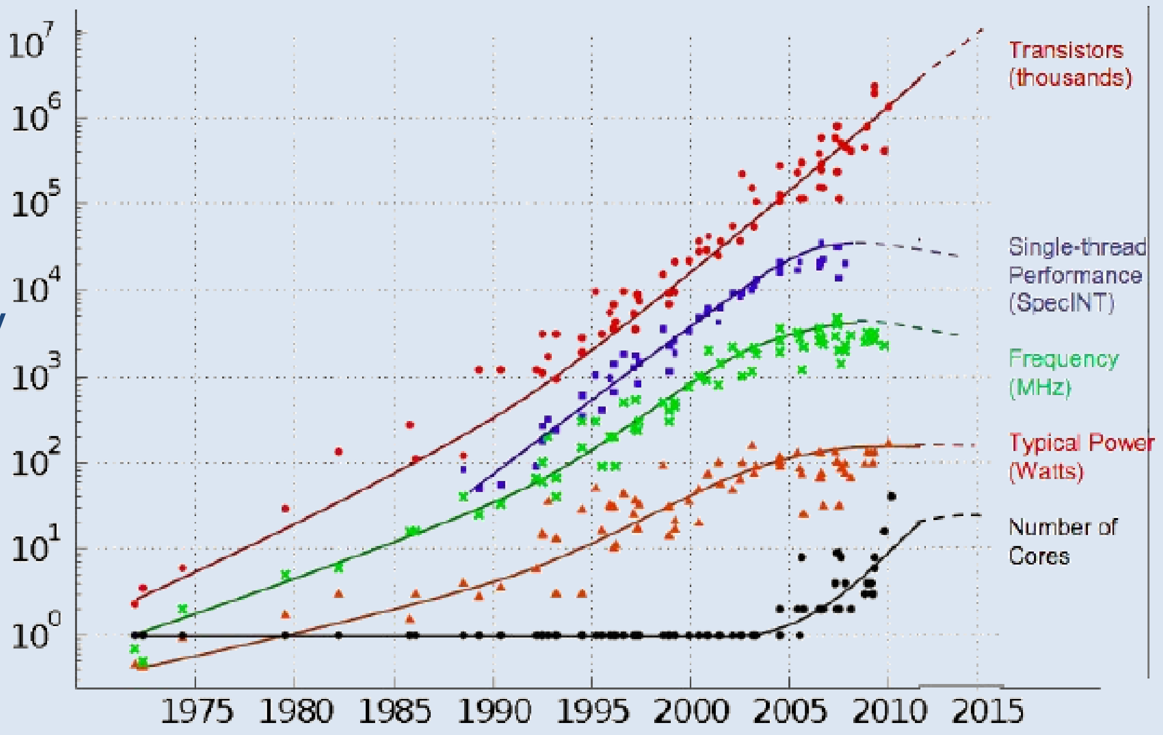
Evolution of Supercomputing



Multi-core ceases scaling
Many-core is more promising

Moore's „law“

- Observation
  - Clock stagnates since 2002
  - Max. clock frequency at about 3 GHz
    - Few exceptions: Power6, Power 7 Gaming
  - # transistors still increases
- Current trends
  - Multi-Core/Many-Core processors
  - Simultaneous Multi Threading (SMT)



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

ExaScale Challenges – energy consumption

- Energy consumption exptected to ever increase

On the CMOS front, the main issue is power consumption, most of which is not strictly related to computation. The paper cites a recent report that projected a 2018-era processor will use 475 picojoules/flop for memory access versus 10 picojoules/flop for the floating point unit. The memory access includes both on-chip communication associated with cache access and off-chip communication to main memory.

Exascale Research: Preparing for the Post–Moore Era

Marc Snir, William Gropp and Peter Kogge

ExaScale Challenges – applications

- Ever increasing levels of parallelism
  - Thousands of node, hundreds of cores, dozens of registers
    - Auto-parallelization or explicit programming
  - Will the size of (cache-) coherency domains continue to grow?
    - From which level on is message-passing a must
  - How many paradigms has to programmer to know?
- MPI + X most probably not sufficient
  - 1 process / core makes orchestration of processes harder
  - GPUs require explicit handling today (CUDA, OpenCL, OpenACC)
- Which paradigm is the future one
  - MPI + X + Y?   PGAS + X (+Y)?
  - PGAS: UPC, Co-Array Fortran, X10, Chapel, Fortress, ….

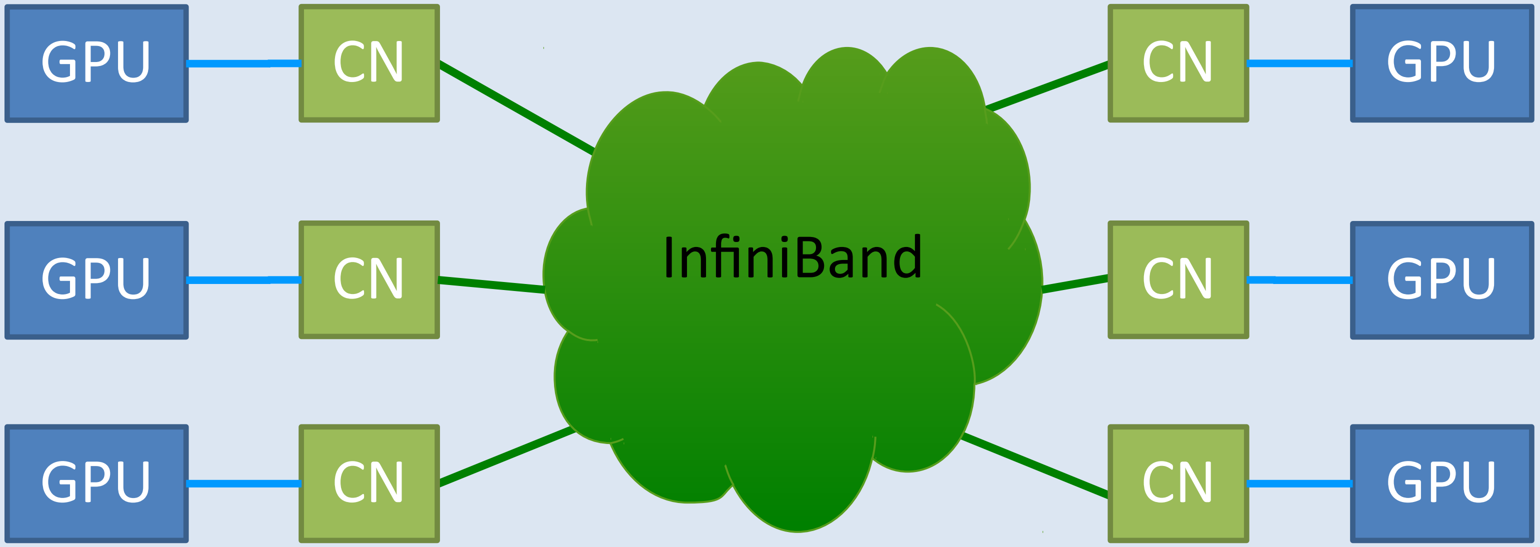
- Do the applications provide a sufficient amount of parallelism?

Application's Scalability

- Only few application capable to scale to O(450k) cores
  - Sparse matrix-vector codes
  - Highly regular communication patterns
  - Well suited for BG/Q
- Most applications are more complex
  - Less regular control flow and memory access
  - Complicated communication patterns
  - Less capable to exploit accelerators
- How to map different requirements to most suited hardware
  - Heterogeneity might be a benefit
  - Do we need better programming models?

Heterogeneity

Heterogeneous Clusters

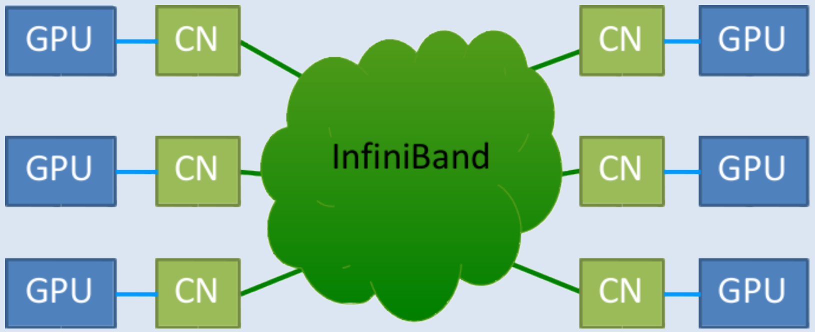


- Flat IB-topology
- Simple management of resources

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

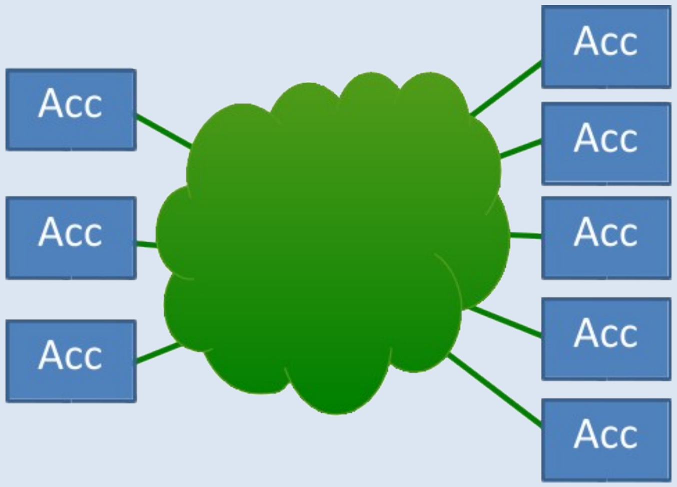
Accelerated Cluster vs. Cluster of Accelerators

- Cluster with Accelerators
  - Accelerator needs a host CPU
  - Static assignment of 1 or more acc.
    - PCIe bus turns out to be a bottleneck
  - Requires explicit GPU programming
    - Cuda, OpenCL, OpenACC, etc.

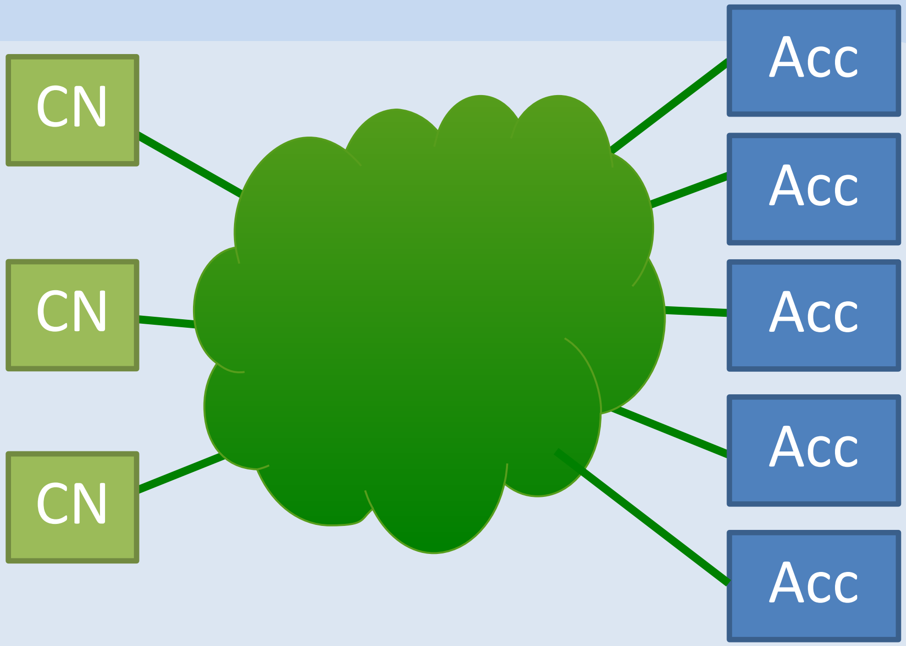


Cluster of Accelerators only

- Node consists of Accelerator only
  - Directly connected to network
  - Today only with Xeon Phi
- Static and dynamical assignment possible
- Concept can be adapted to concurrency levels

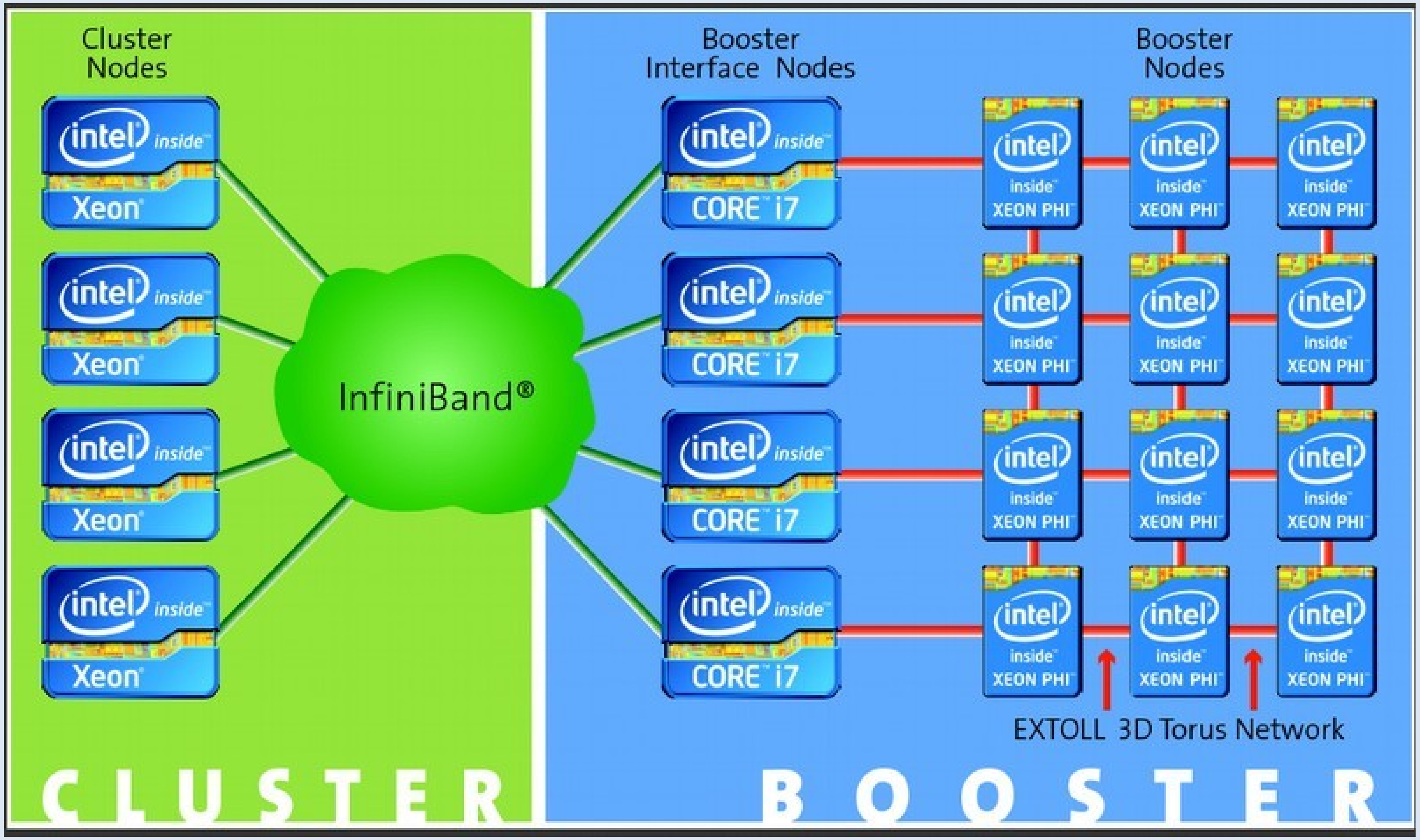


Alternative Integration

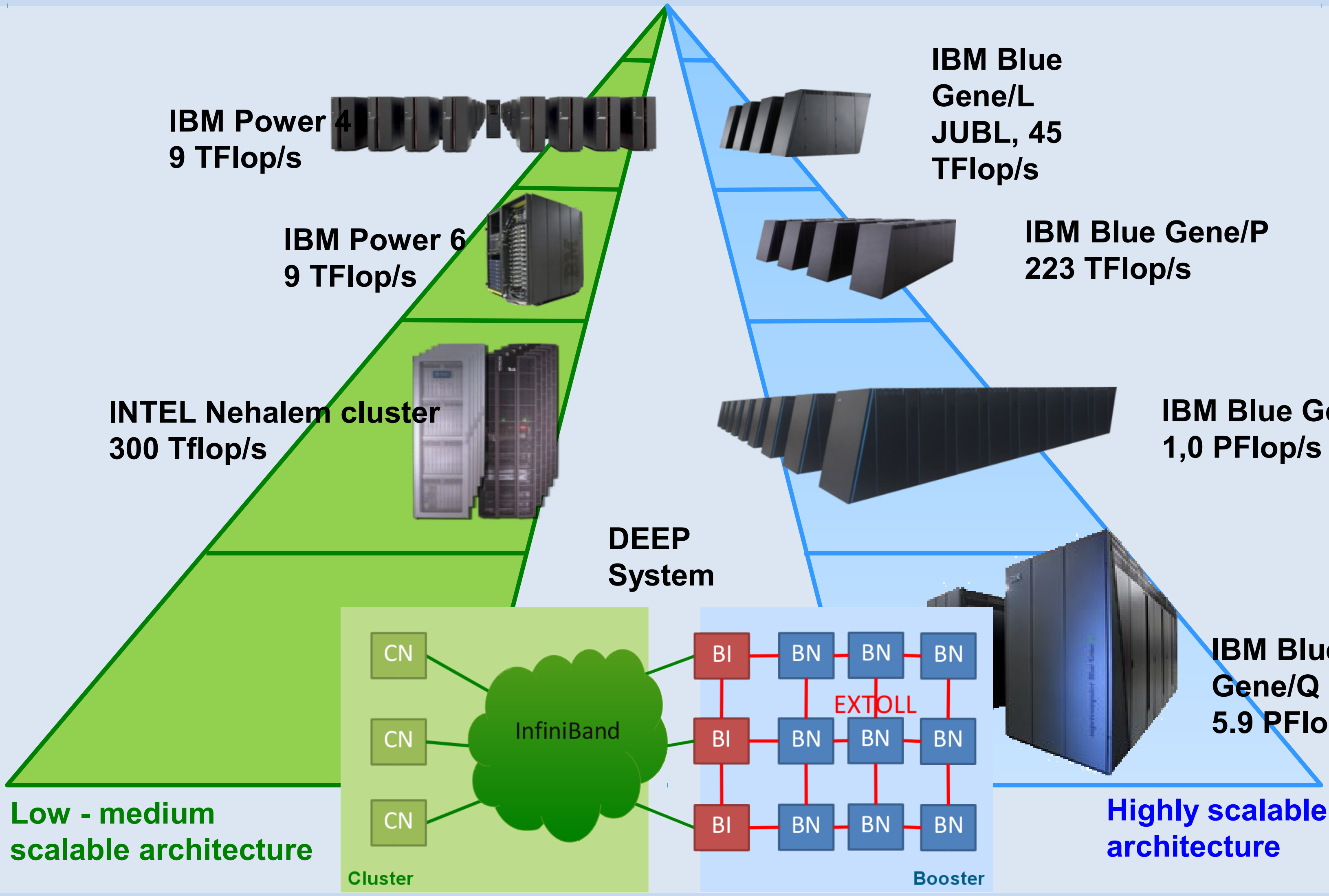


- Go for more capable accelerators (e.g. MIC)
- Attach all nodes to a low-latency fabric
- All nodes might act autonomously
- Dynamical assignment of cluster-nodes and accelerators
  - IB can be assumed as fast as PCIe besides latency
- Ability to off-load more complex (including parallel) kernels
 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

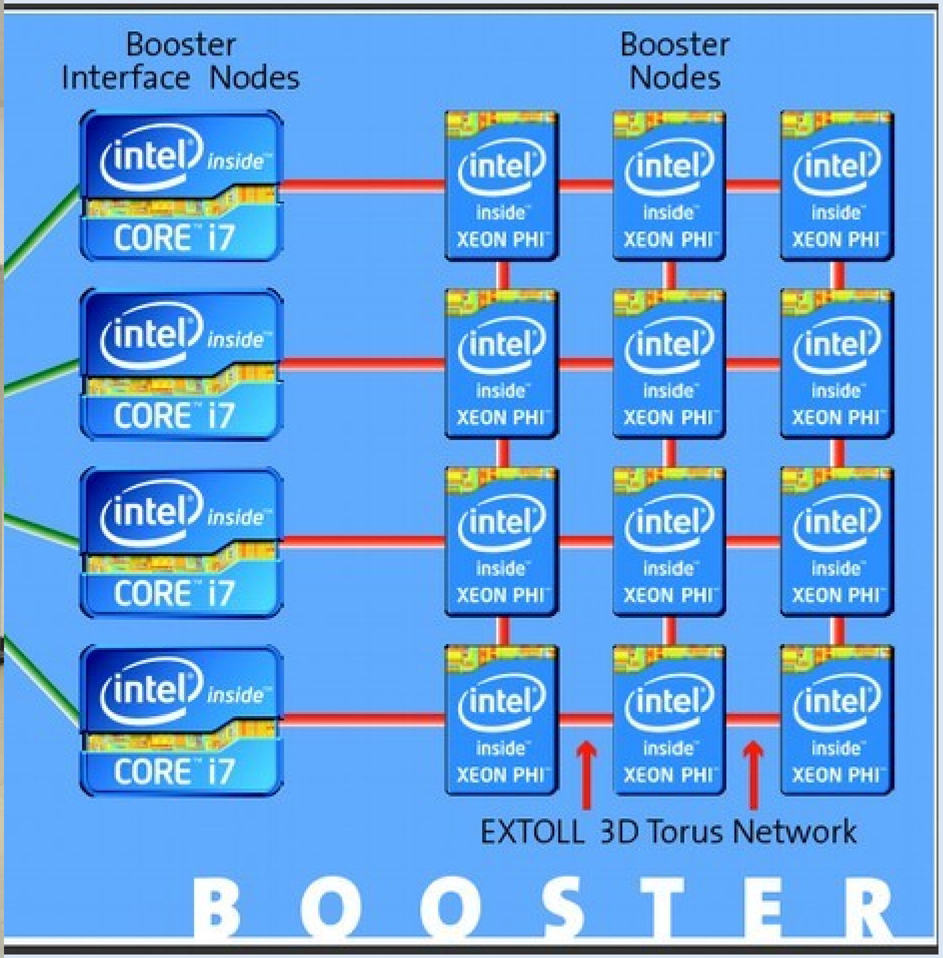
Hardware Architecture



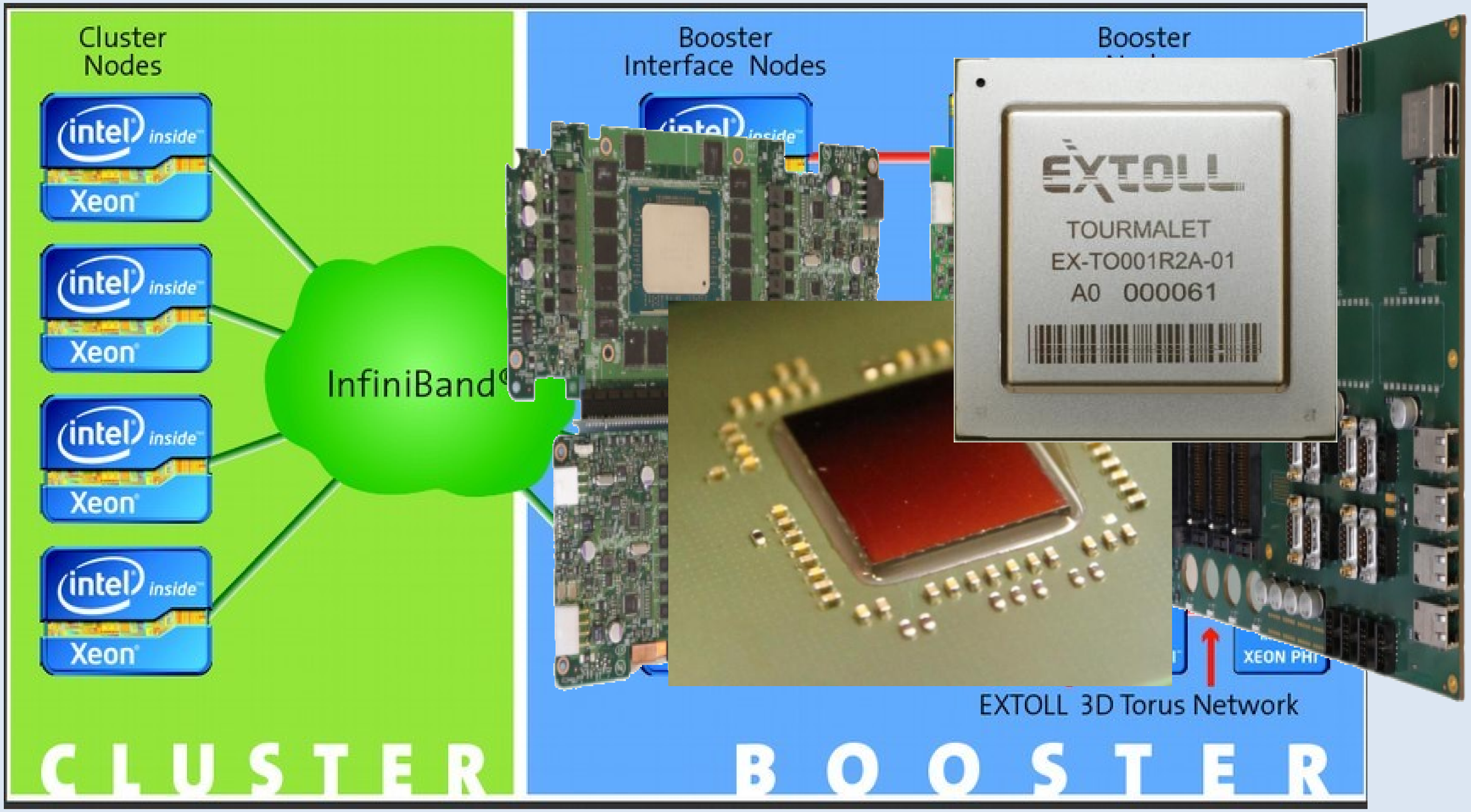
Positioning DEEP



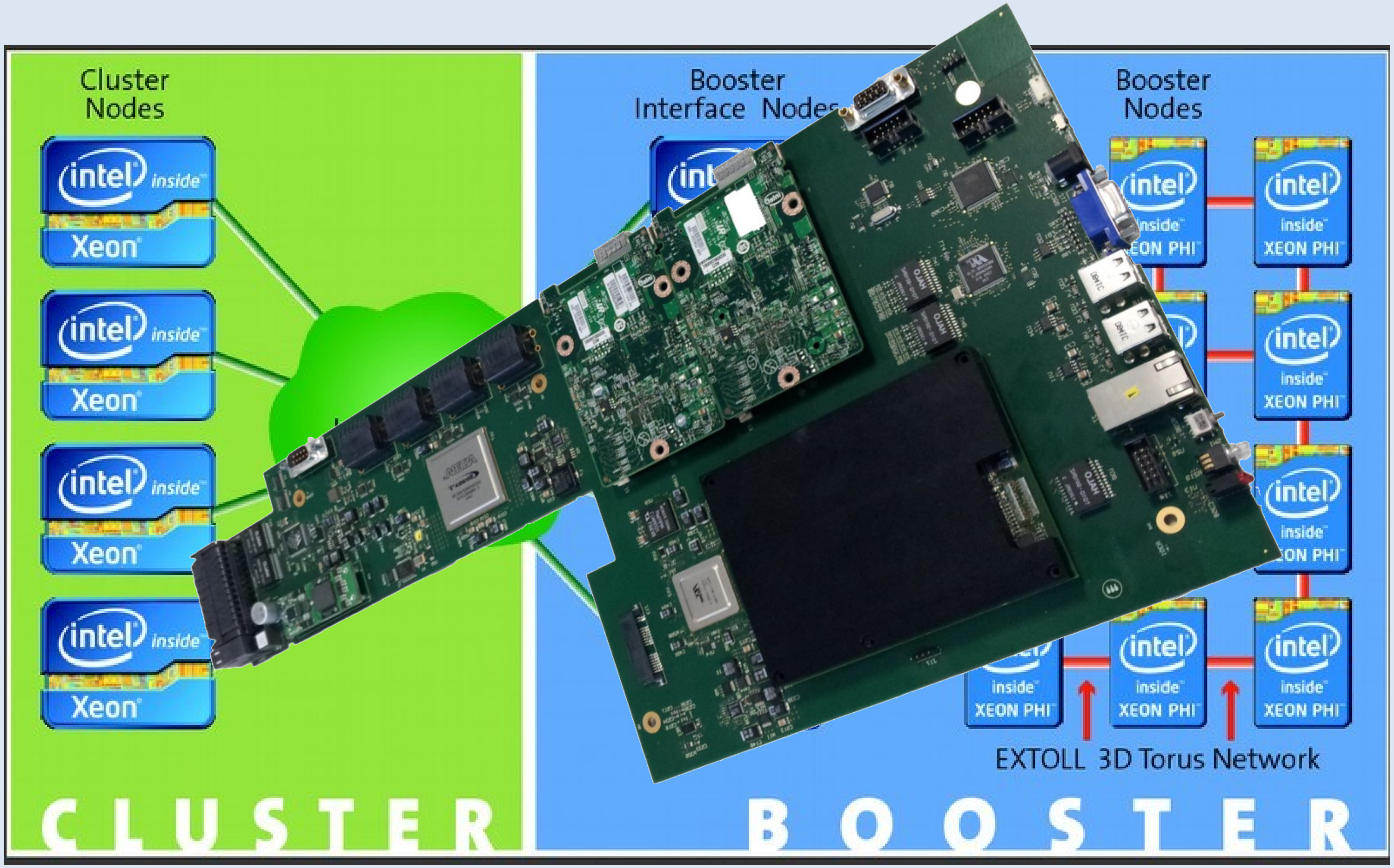
DEEP Hardware



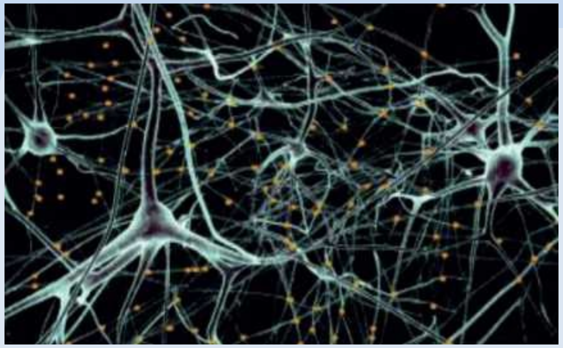
DEEP Hardware



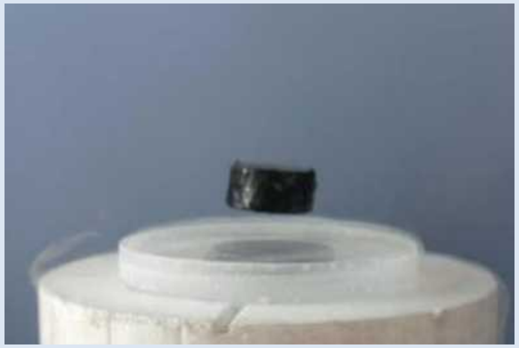
DEEP Hardware



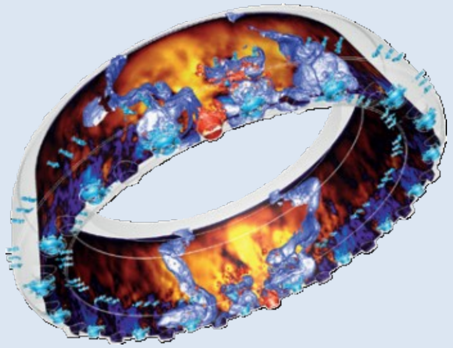
Guiding Applications



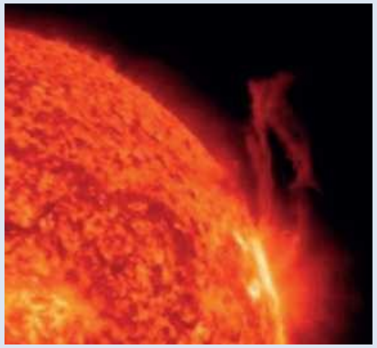
Brain simulation
EPFL



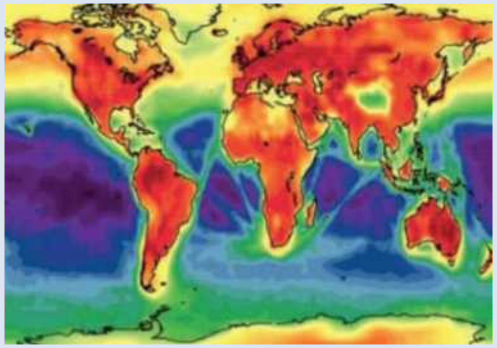
HT_c superconductivity
CINECA



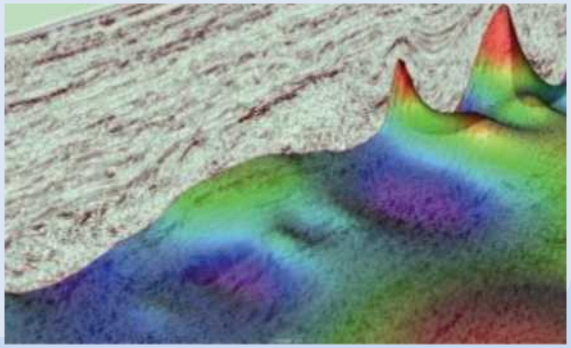
CFE
CERFACS



Space Weather
K.U.Leuven

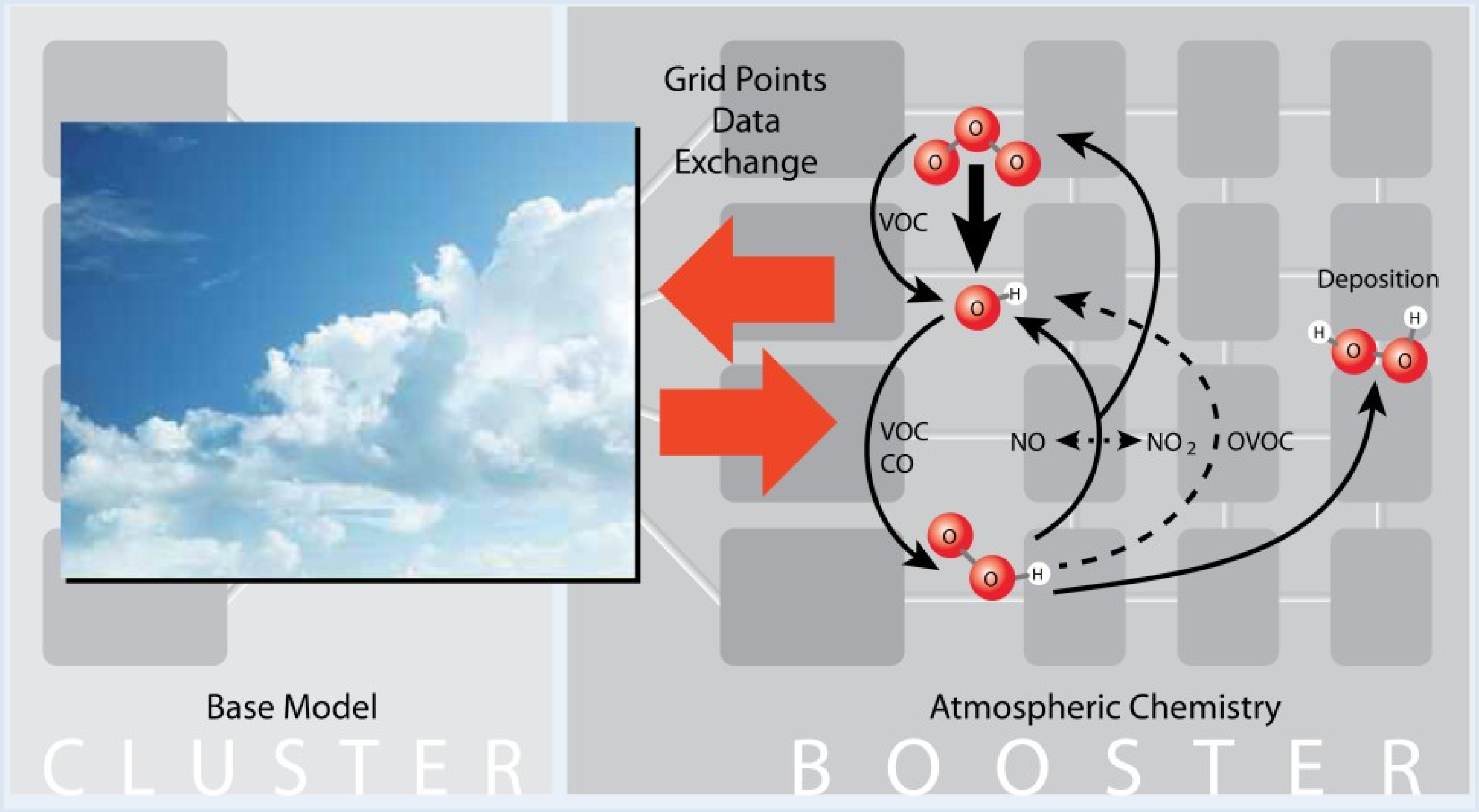


Climate Simulation
Cyprus Institute

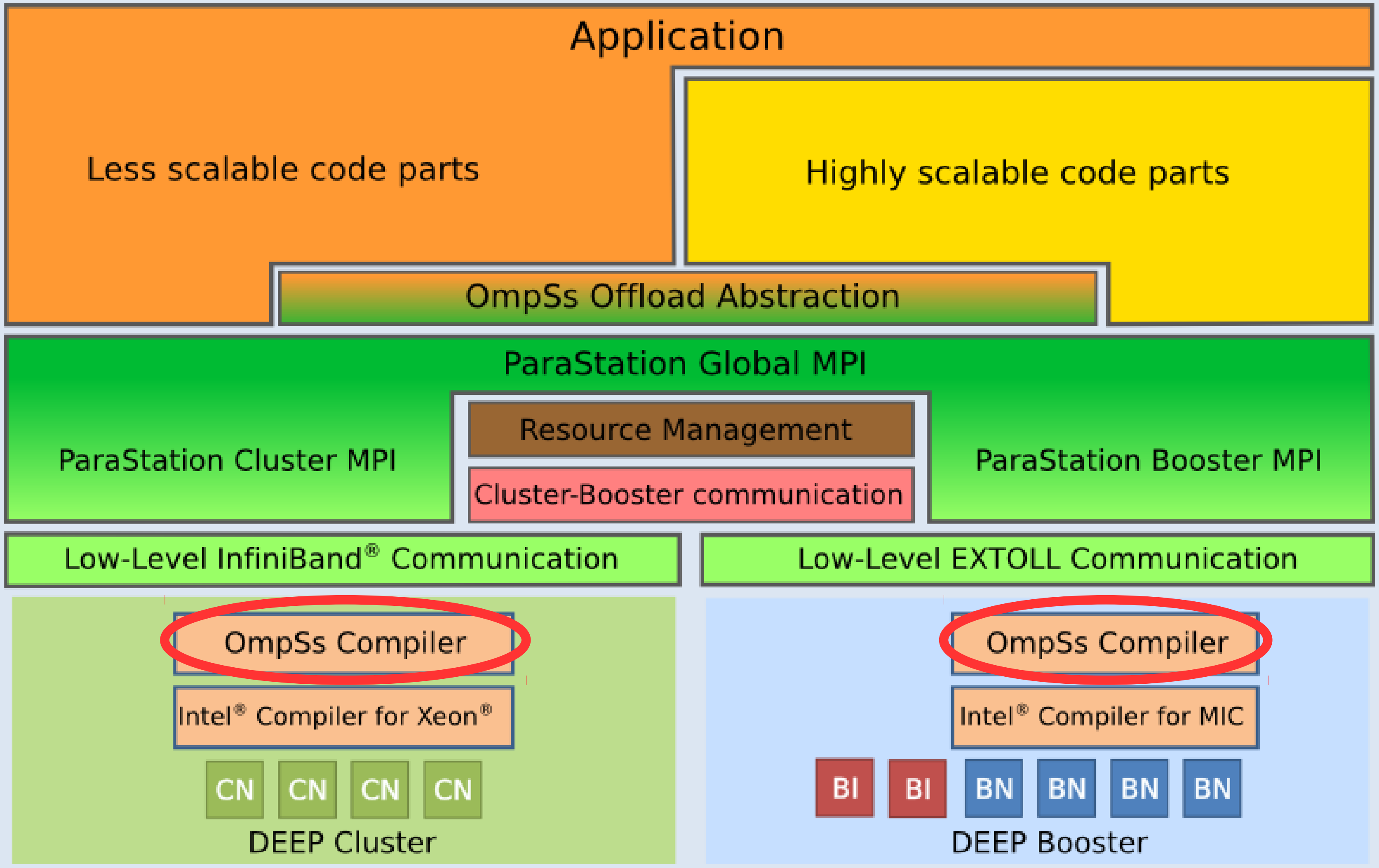


Seismic imaging
CGGVeritas

Cyl – Climate Simulation

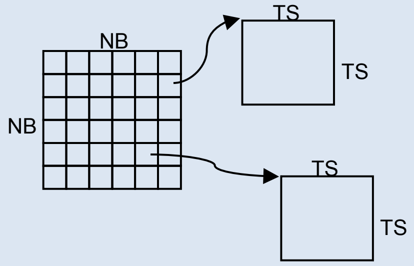


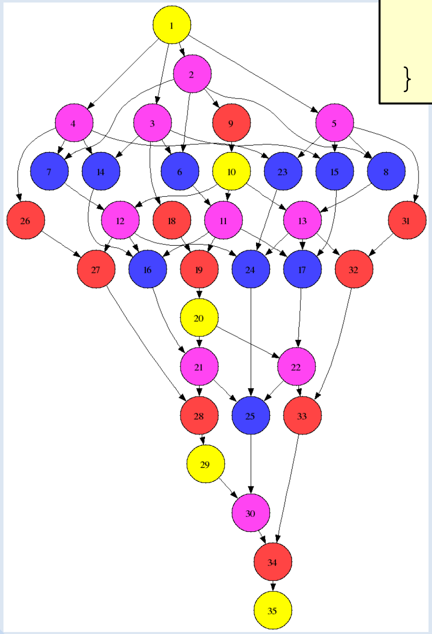
Software Architecture



OmpSs:
tasks, dependencies, heterogeneity

```
void Cholesky( float *A[NT] ) {
int i, j, k;
for (k=0; k<NT; k++) {
   spotrf (A[k][k]) ;
   for (i=k+1; i<NT; i++)
      strsm (A[k][k], A[k][i]);
   for (i=k+1; i<NT; i++) {
      for (j=k+1; j<i; j++)
         sgemm( A[k][i], A[k][j], A[j][i]);
      ssyrk (A[k][i], A[i][i]);
   }
}
```

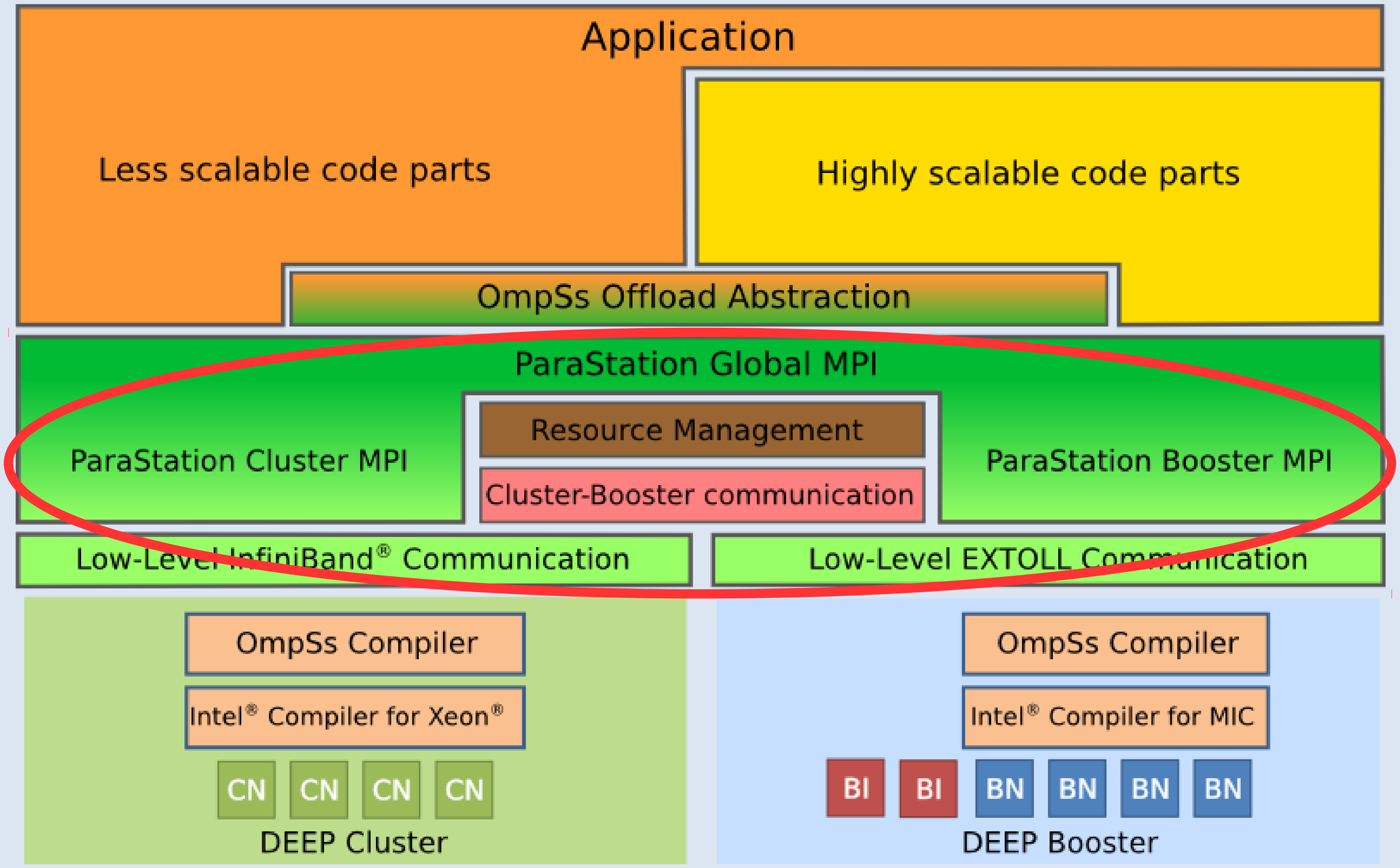




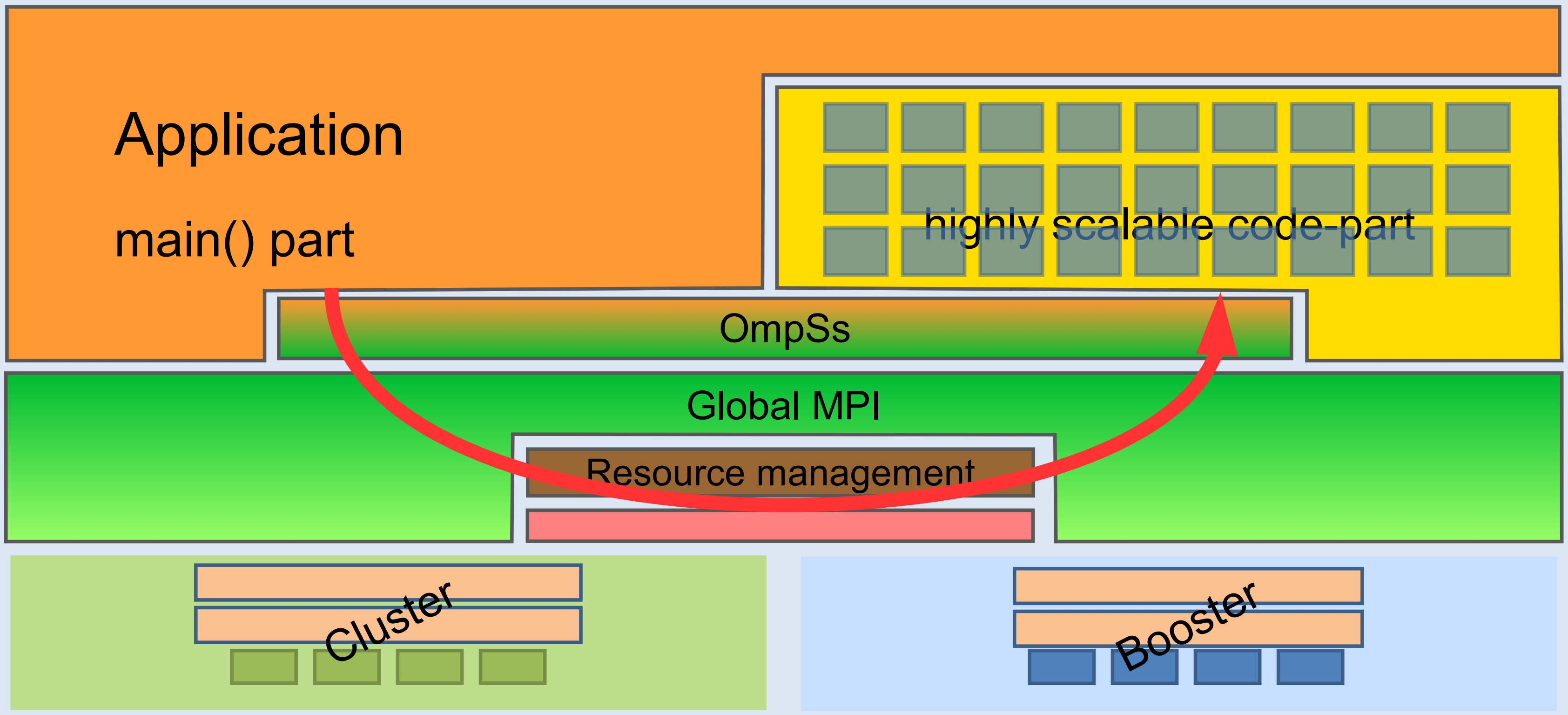
```
#pragma omp task inout ([TS][TS]A)
void spotrf (float *A); ●
#pragma omp task input ([TS][TS]T) inout ([TS][TS]B)
void strsm (float *T, float *B); ●
#pragma omp task input ([TS][TS]A,[TS][TS]B) inout ([TS][TS]C)
void sgemm (float *A, float *B, float *C); ●
#pragma omp task input ([TS][TS]A) inout ([TS][TS]C)
void ssyrk (float *A, float *C); ●
```

Decouple how we write (think sequential) from how it is executed

Software Architecture



Application Startup

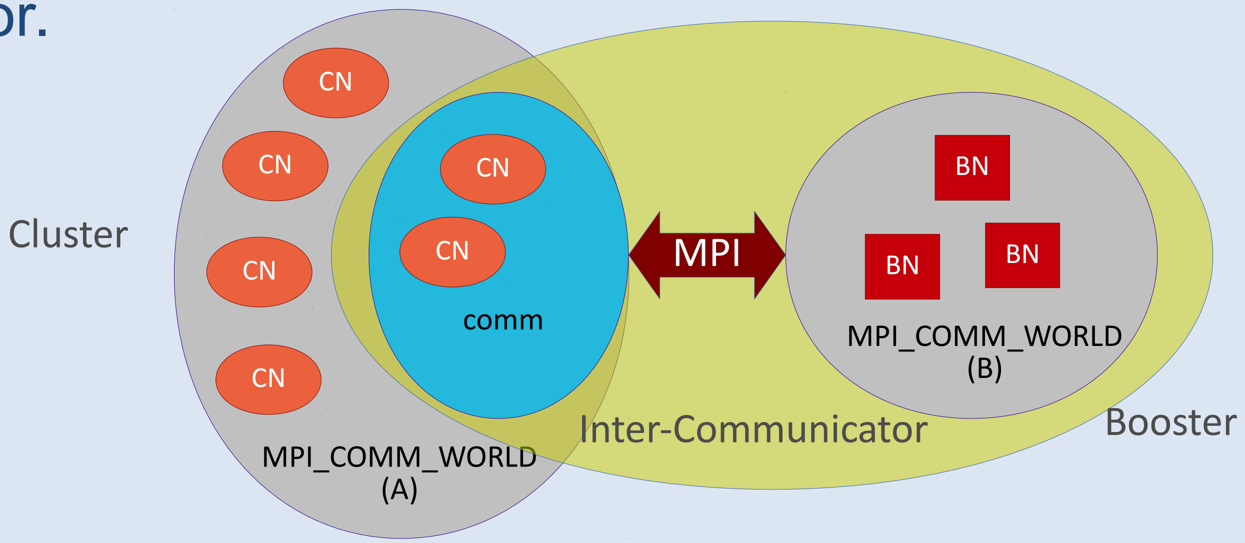


- Application's main()-part runs on Cluster-nodes (CN) only
- Actual spawn done via global MPI
- OmpSs acts as an abstraction layer

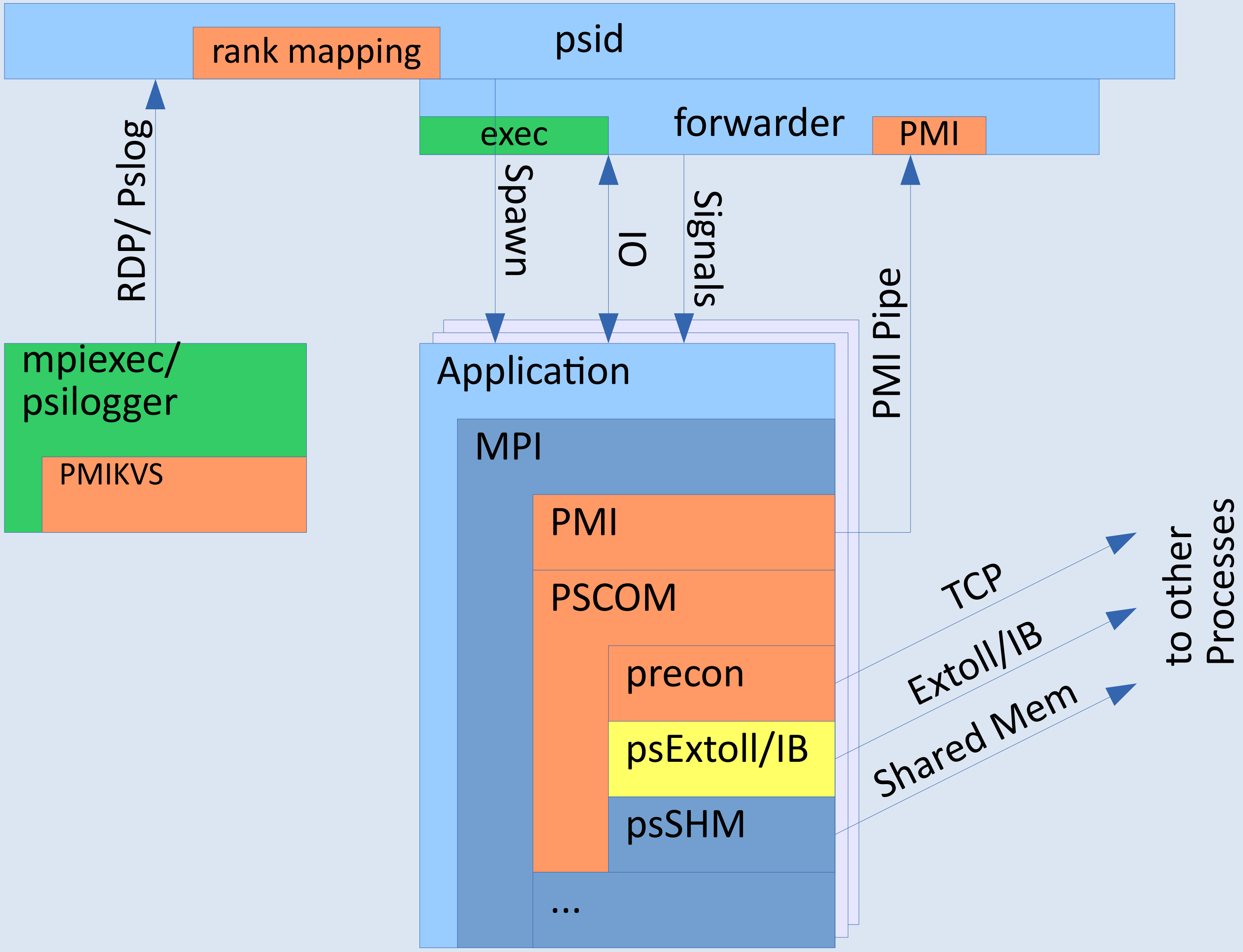
- Spawn is a collective operation of Cluster-processes
- Highly scalable code-parts (HSCP) utilize multiple Booster-nodes (BN)

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.

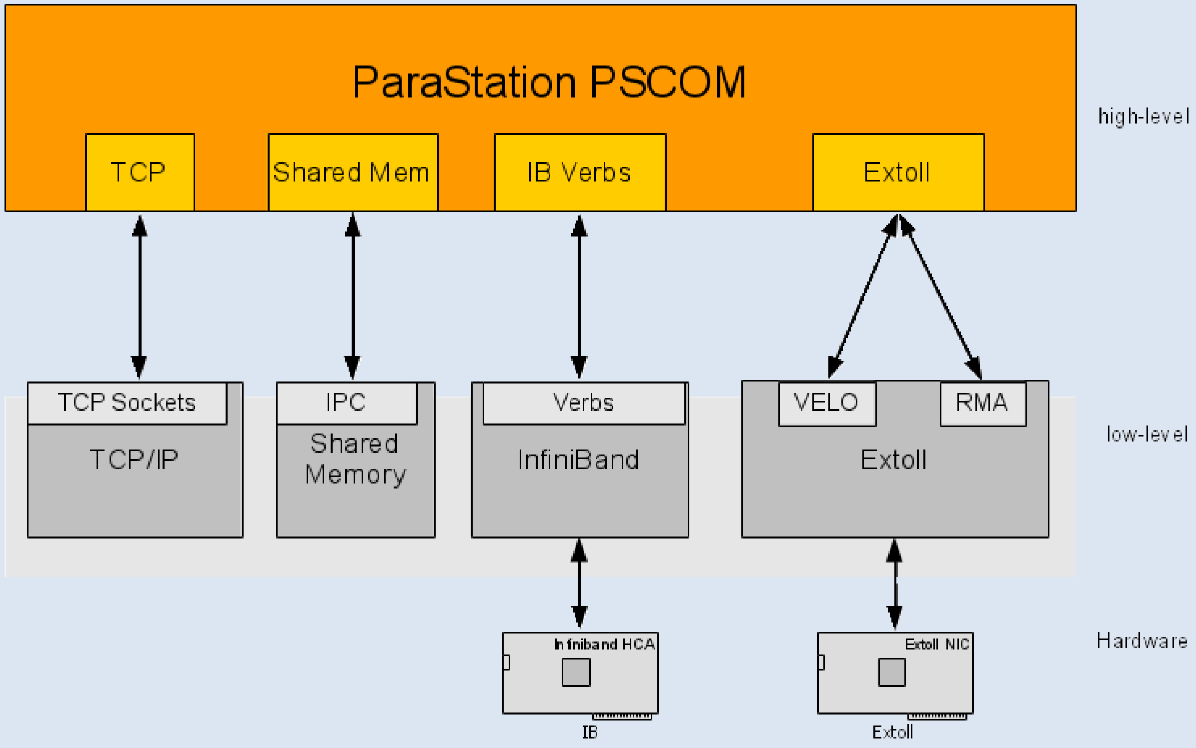


ParaStation MPI

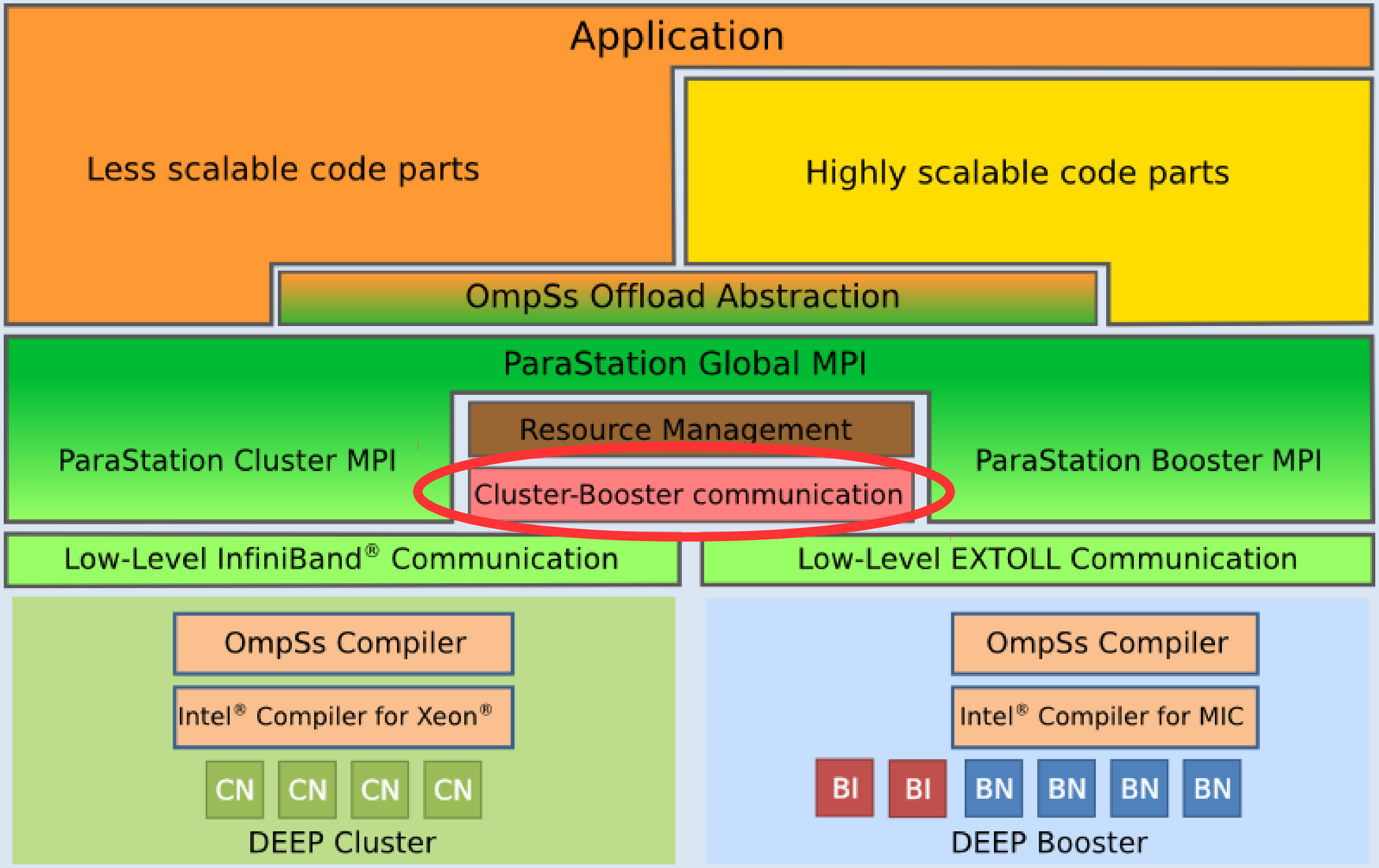


ParaStation pscom

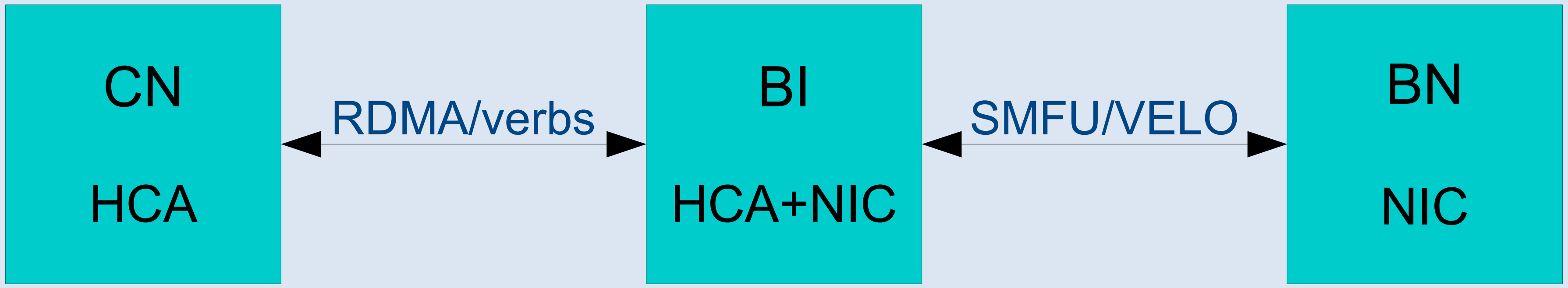
- Unified comm layer
- pscom plugins
  - Modular
  - Flexible to extend
  - Verbs plugin
  - VELO/RMA plugin
  - Easy enabling of Cluster-Booster protocol



Software Architecture

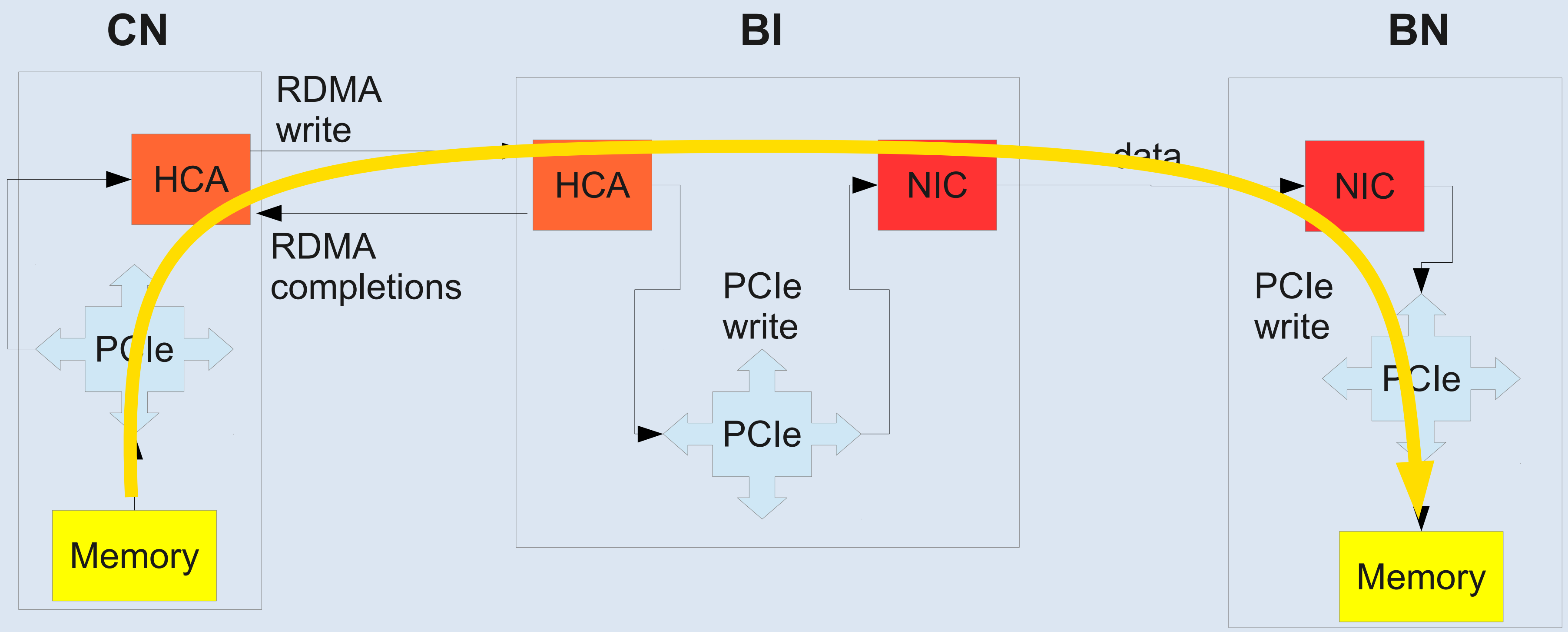


Cluster-Booster Communication



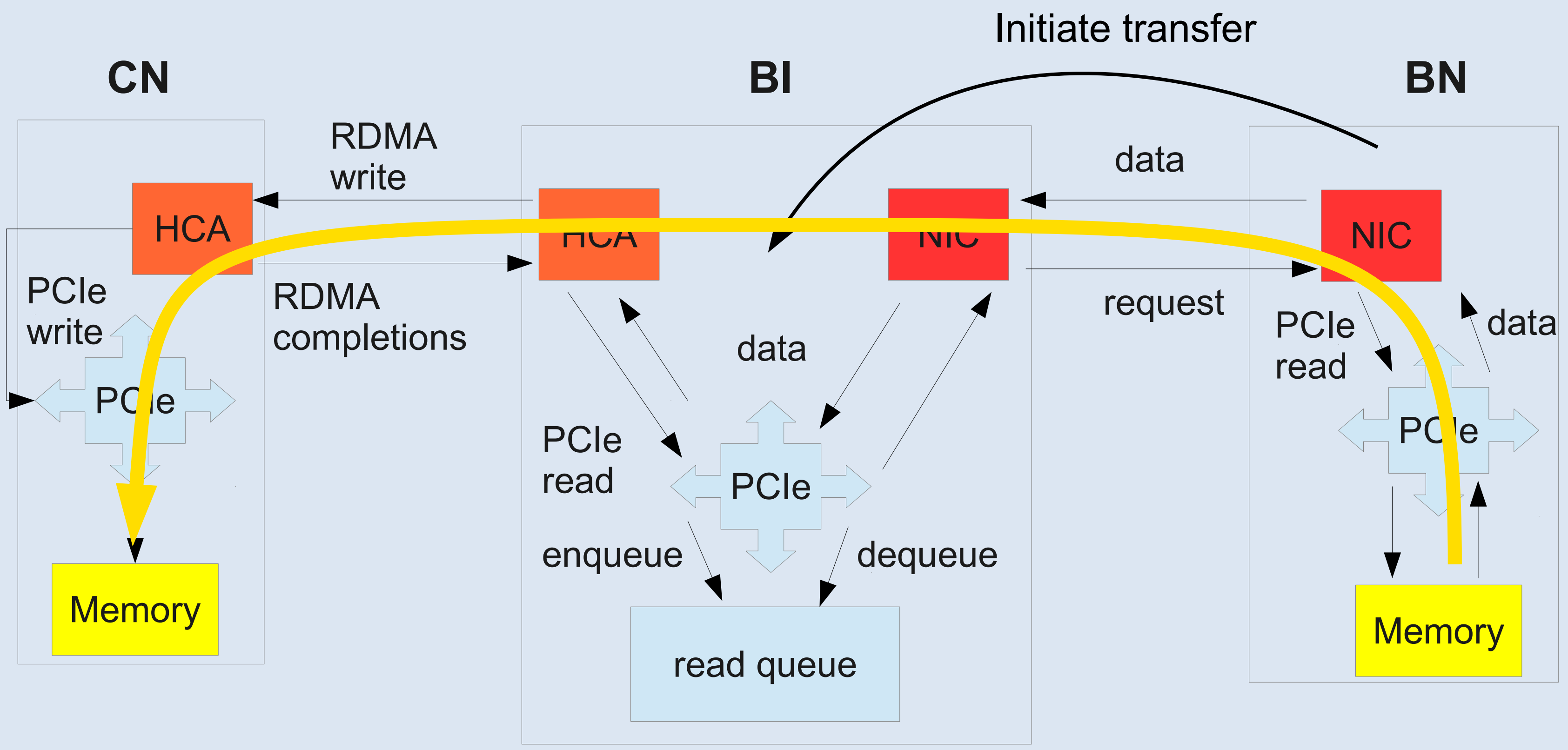
- High-bandwidth, low latency communication between Cluster Nodes and Booster Nodes needed
- Booster Interfaces bridge between InfiniBand and EXTOLL
- A Booster Interface can act in two different modes:
  - Bridge: receive packets, actively forward to end-point
  - Set-up and enable RDMA via SMFU directly between memory locations on CN and BN

CN-to-BN-Write Operation



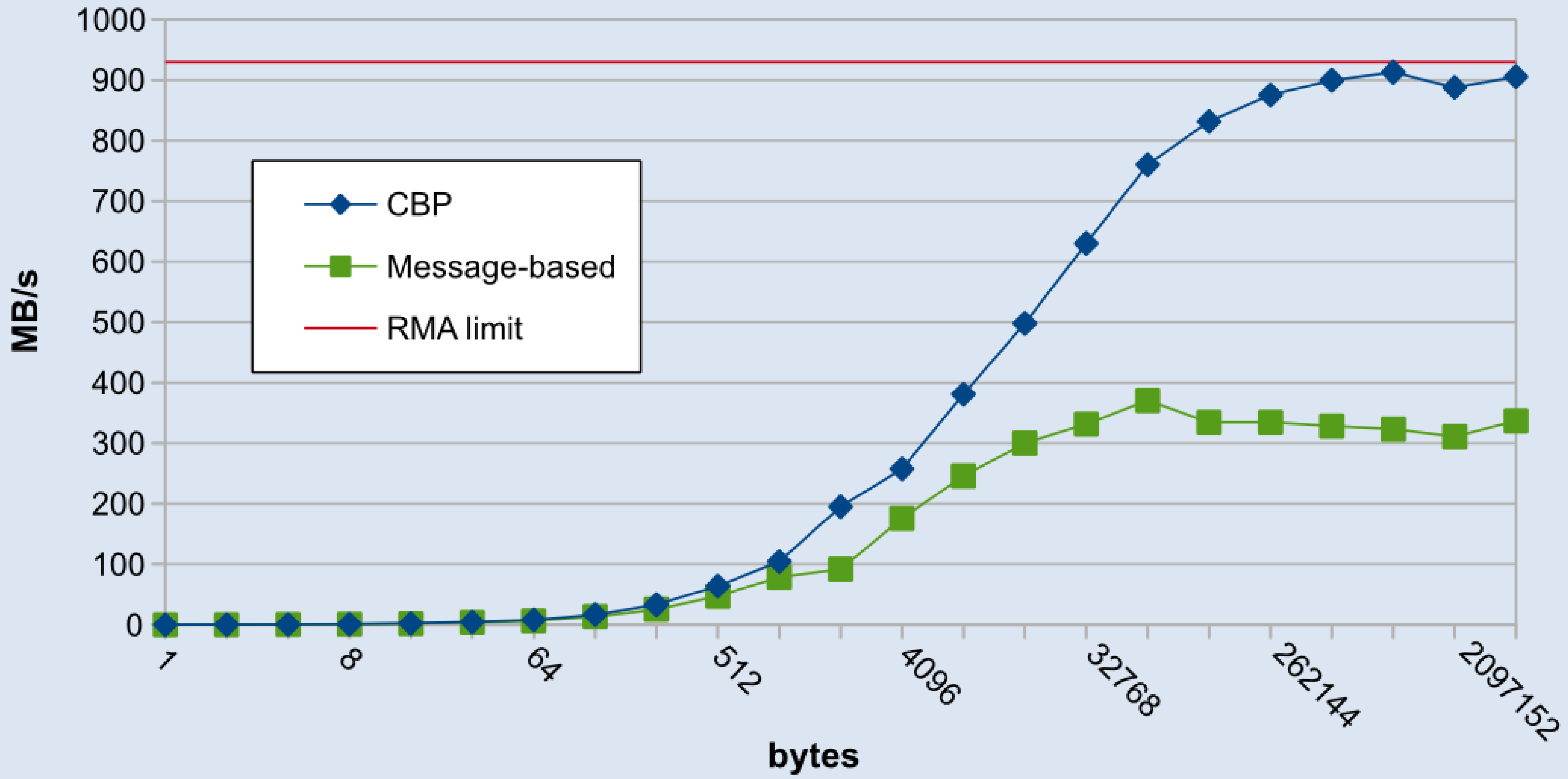
The RDMA write operation *initiated on the Cluster Node* is "forwarded" via PCIe and EXTOLL to the Booster Node

BN-to-CN Write Operation

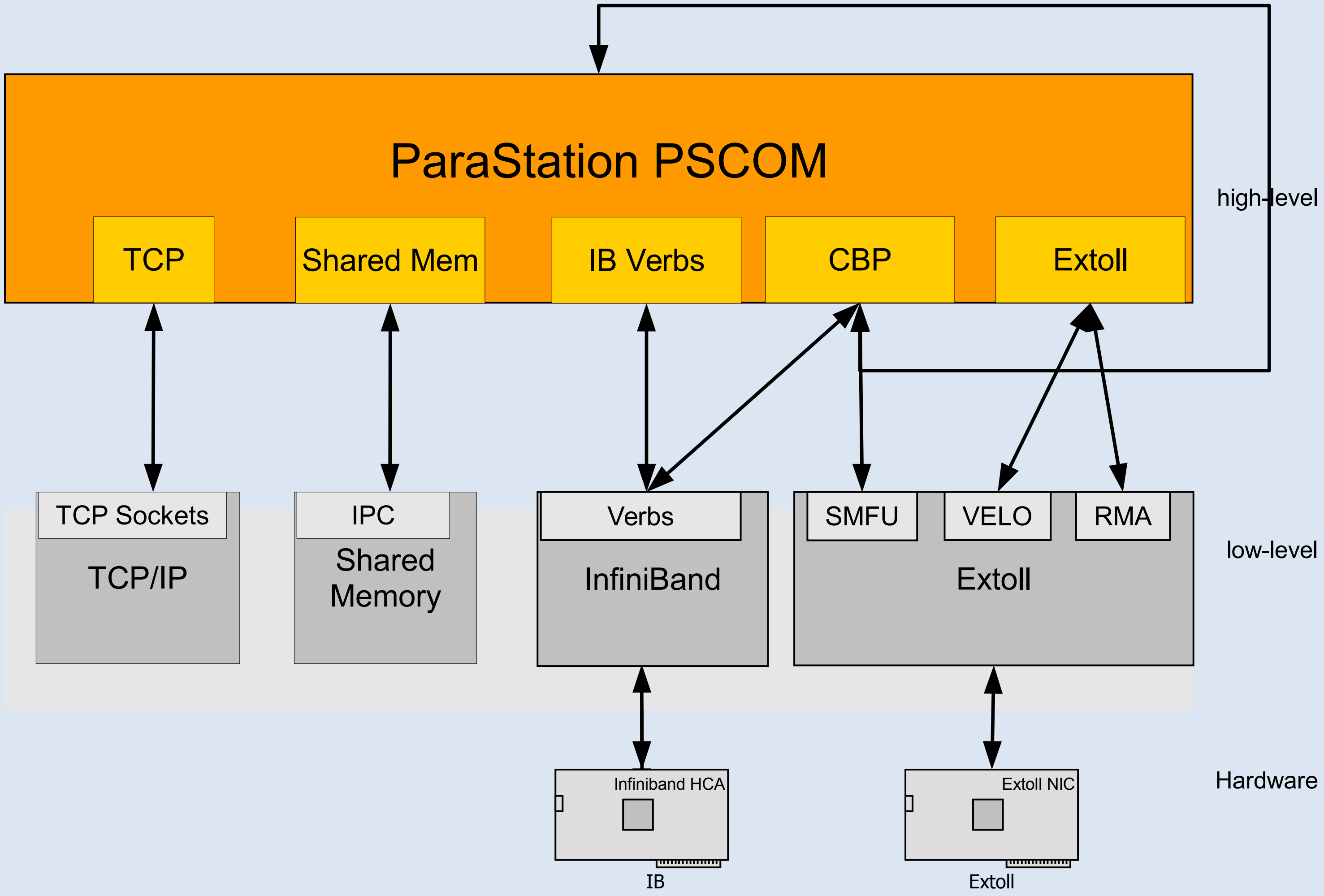


- The *initiating Booster Node* asks the Booster Interface's HCA to do an RDMA write operation to the Cluster Node's Memory
- The resulting PCIe read operation is “forwarded” by Extoll

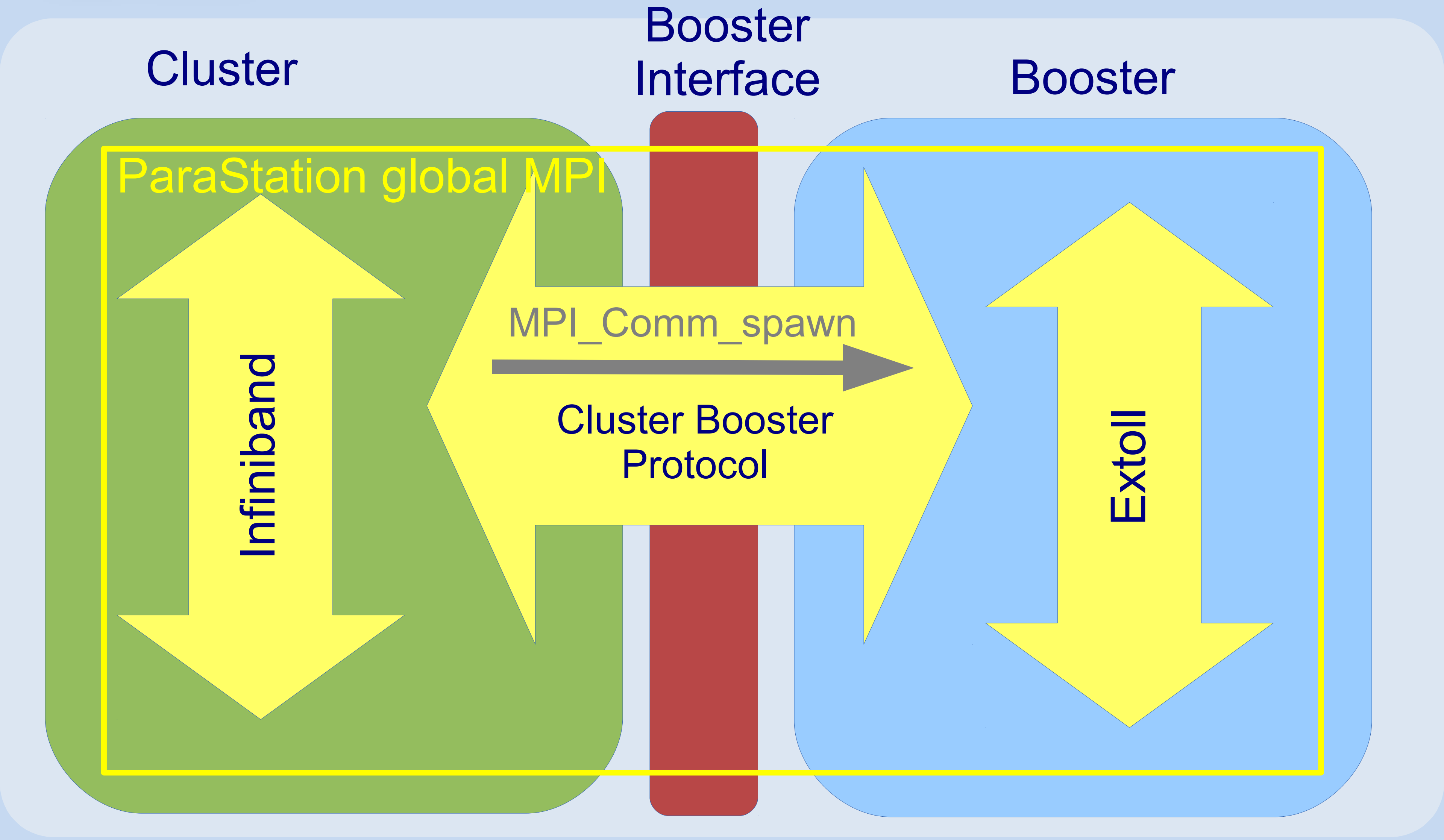
CBP Ping-Pong



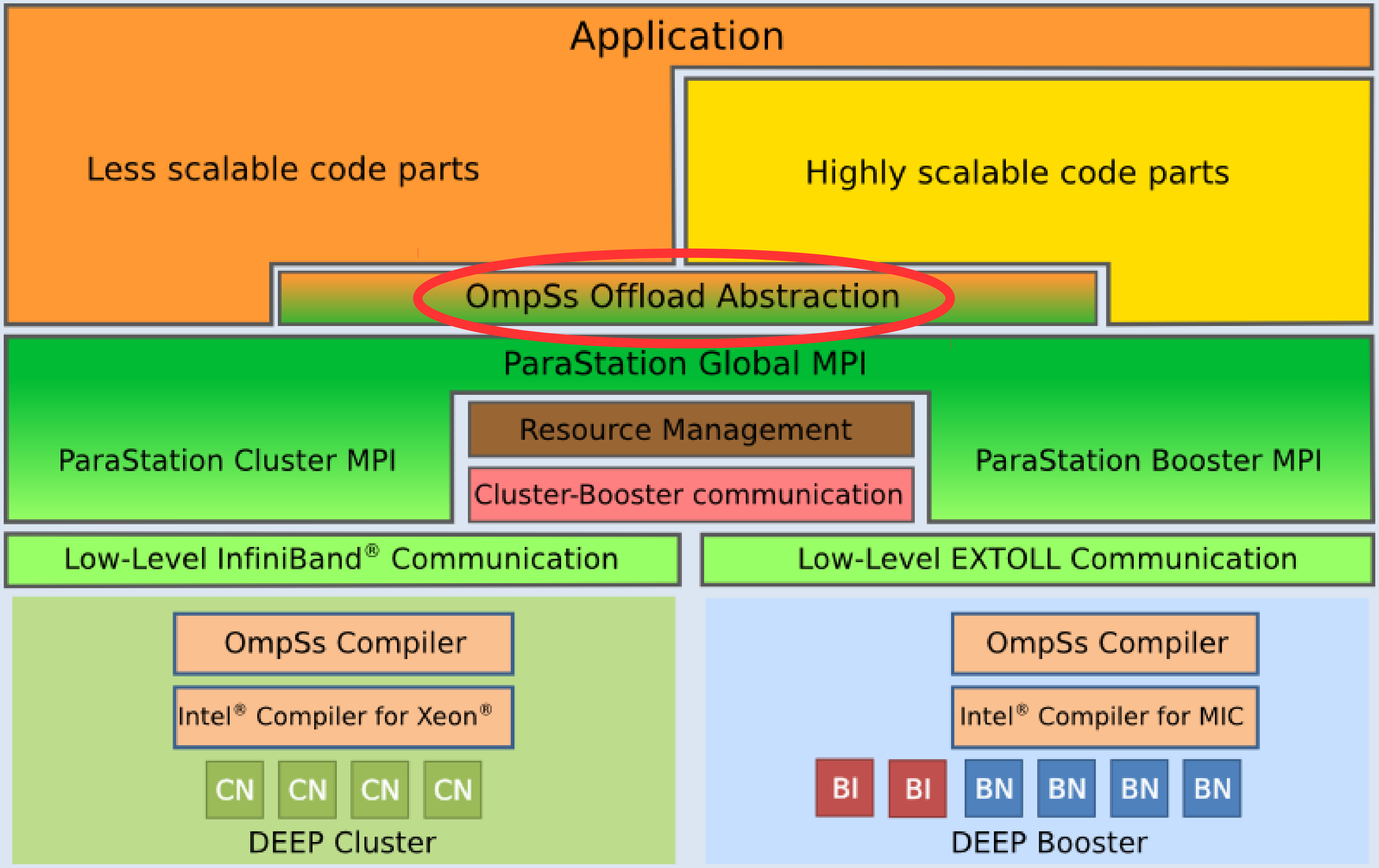
CBP Integration



Programming



Software Architecture



OmpSs Offload Abstraction

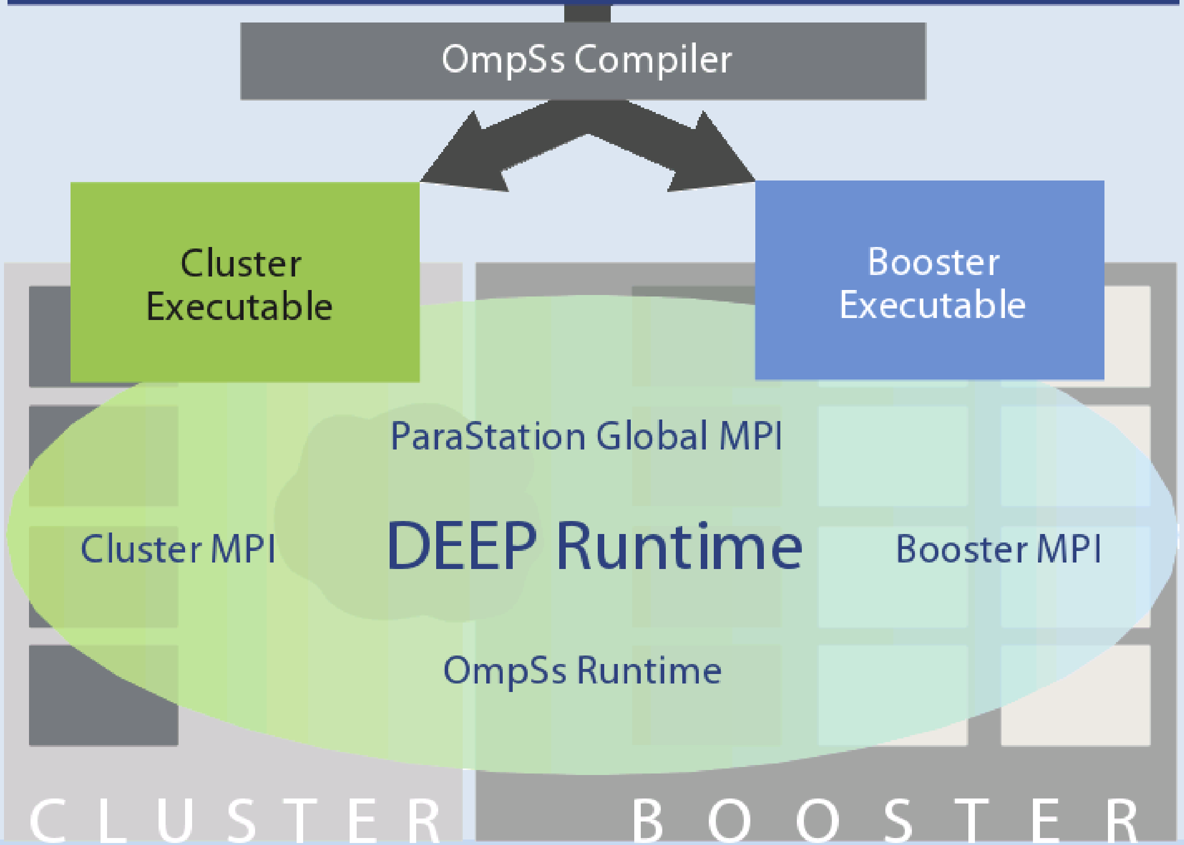
Source Code

```
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, …);}}
```

Compiler

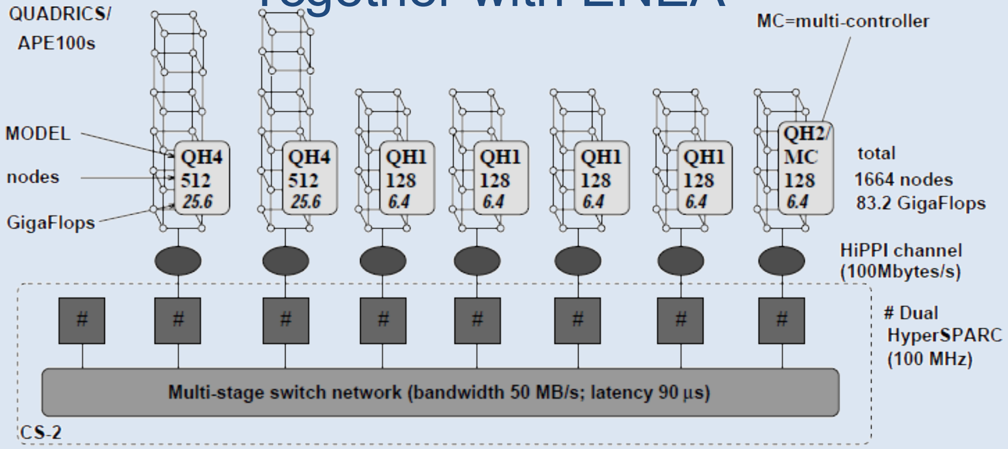
Application Binaries

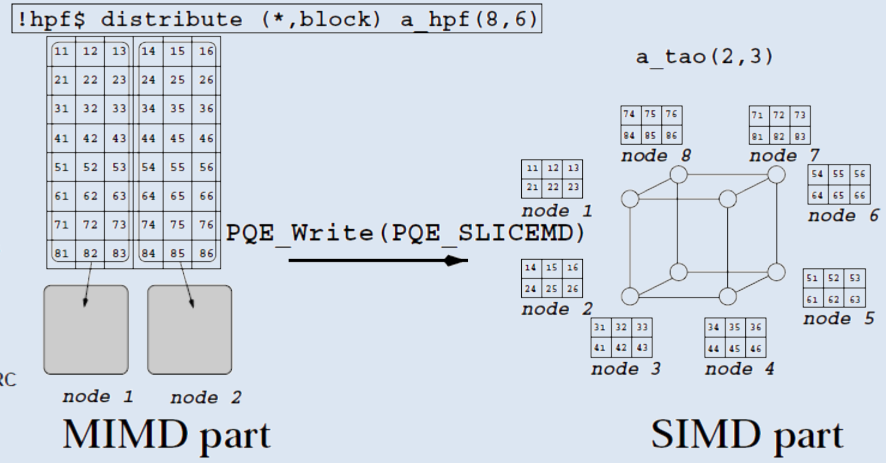
DEEP Runtime



A novel architecture?

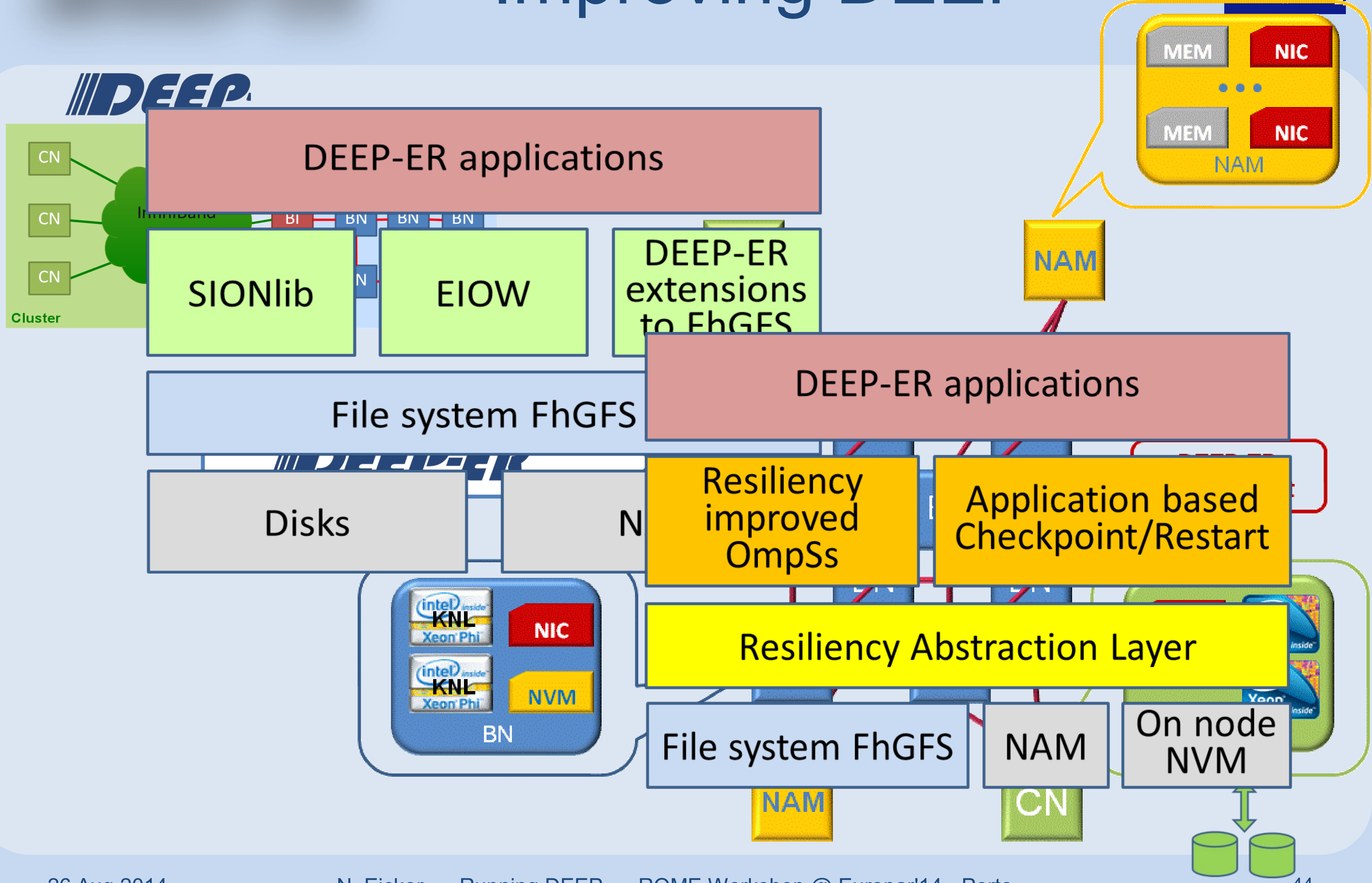
- PQE1 (1996)
  - A Meiko / QSW CS-2
  - Coupled to APE100
  - Quadrics Supercomputing World's first step to commercialize the APE architecture (special purpose machines for lattice QCD)
  - Together with ENEA


QUADRICS/ APE100s
MODEL
nodes
GigaFlops
QH4 512 25.6
QH4 512 25.6
QH1 128 6.4
QH1 128 6.4
QH1 128 6.4
QH1 128 6.4
QH2/ MC 128 6.4
MC=multi-controller
total 1664 nodes 83.2 GigaFlops
HiPPI channel (100Mbytes/s)
# Dual HyperSPARC (100 MHz)
Multi-stage switch network (bandwidth 50 MB/s; latency 90 µs)
CS-2



- Plans for QuadricsPQE2000 never worked out

Improving DEEP



Take aways

- DEEP explores new ways to implement heterogeneity
  - Cluster Booster Architecture
  - Programming Model
- MPI has to support this heterogeneity
  - Use MPI_Comm_spawn() for Booster Offloading
- Try to hide the details via OmpSs abstraction layer
- Powerfull runtime supporting the applications
- Waiting for final hardware to appear

- More info: http://www.deep-project.eu