Taming Heterogeneity by Segregation

The DEEP and DEEP-ER take on Heterogeneous Cluster Architectures

♦

Norbert Eicker

Jülich Supercomputing Centre & University of Wuppertal

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
  - Software stack
- Outlook on DEEP-ER
- Summary

DEEP face-to-face Leuven

DEEP-ER kickoff Jülich

Partners

Fraunhofer ITWM

CGG Passion for Geoscience

CINECA

The Cyprus Institute

EPFL École Polytechnique Fédérale de Lausanne

ASTRON

CERFACS

intel

BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación

EUROTECH Imagine. Build. Succeed.

German Research School for Simulation Sciences

Inria Inventors for the digital world

ParTec Cluster Competence Center

KU Leuven

Mellanox Technologies

Universität Regensburg

Seagate

Jülich Forschungszentrum

lrz

Ruprecht-Karls-Universität Heidelberg

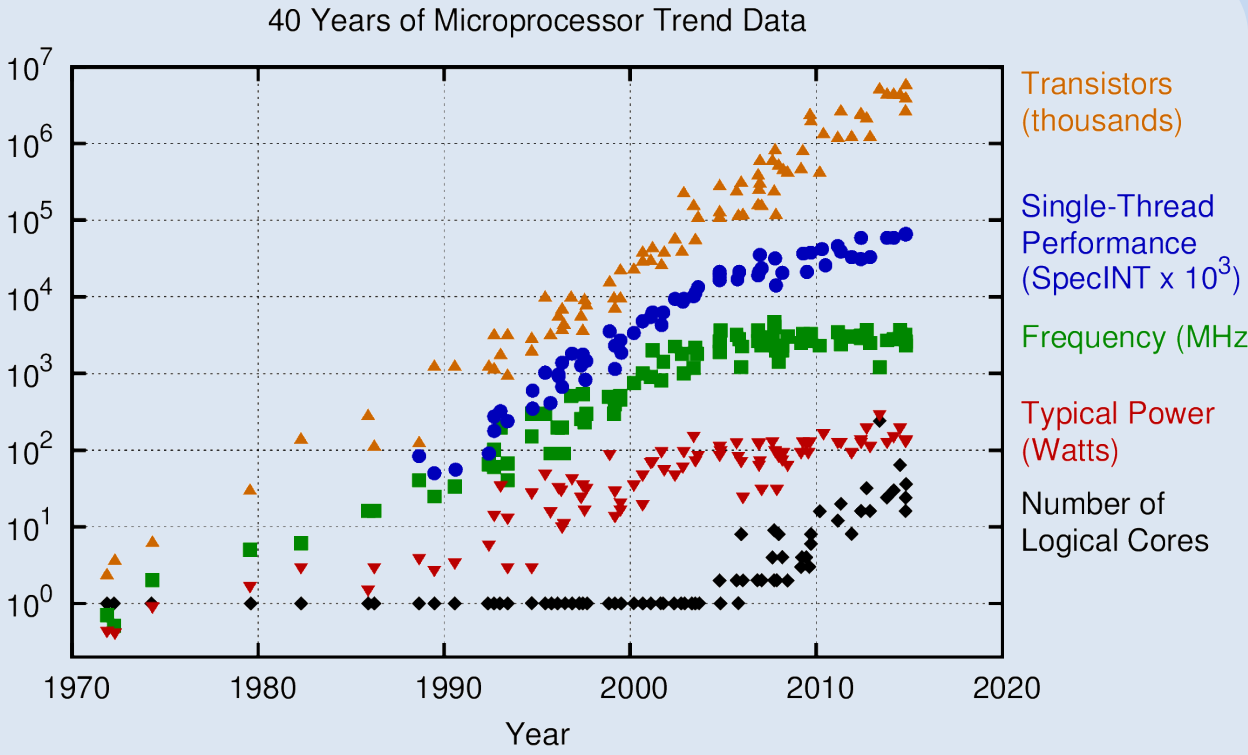
DEEP Project

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

Heterogeneity

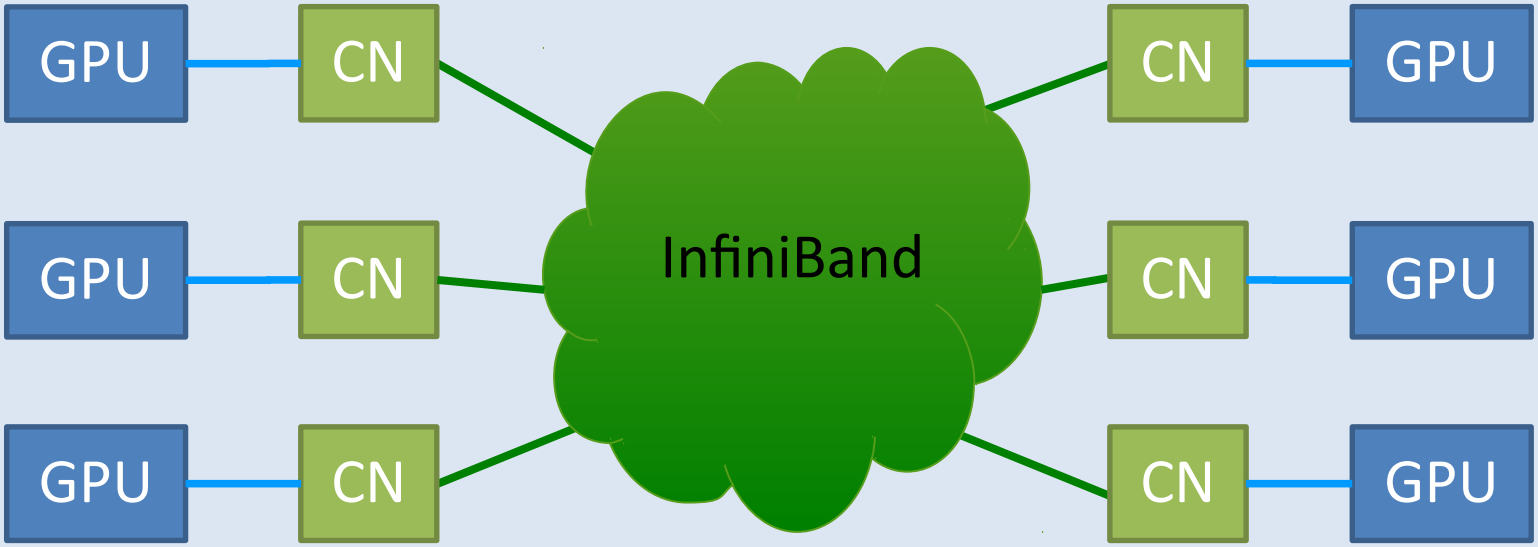
Moore's law

- Observation
  - Clock stagnates since 2002
  - Max. clock freq. 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 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

Heterogeneous Clusters



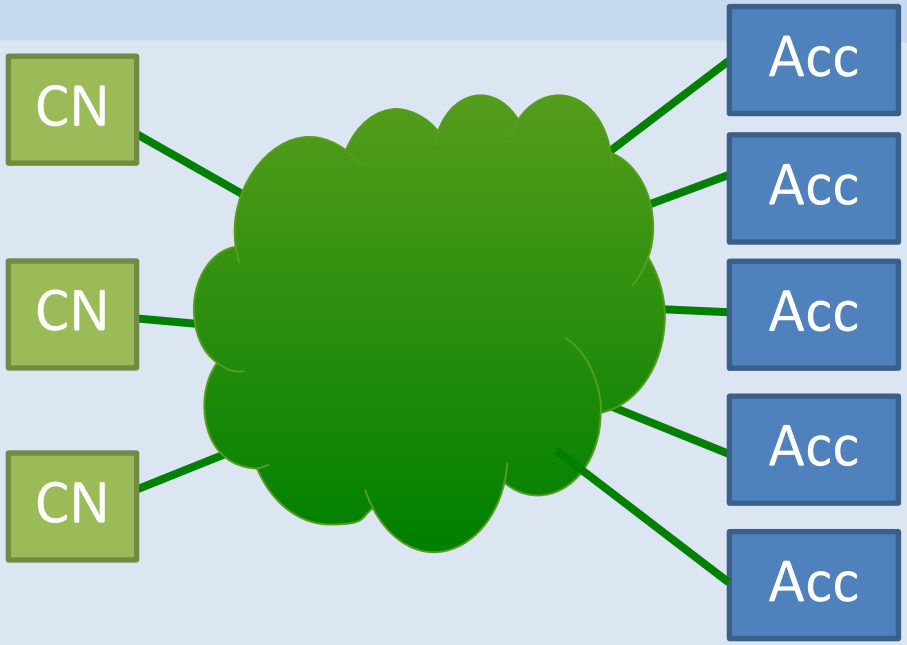
Flat IB-topology

Simple management of resources

Static assignment of CPUs to GPUs

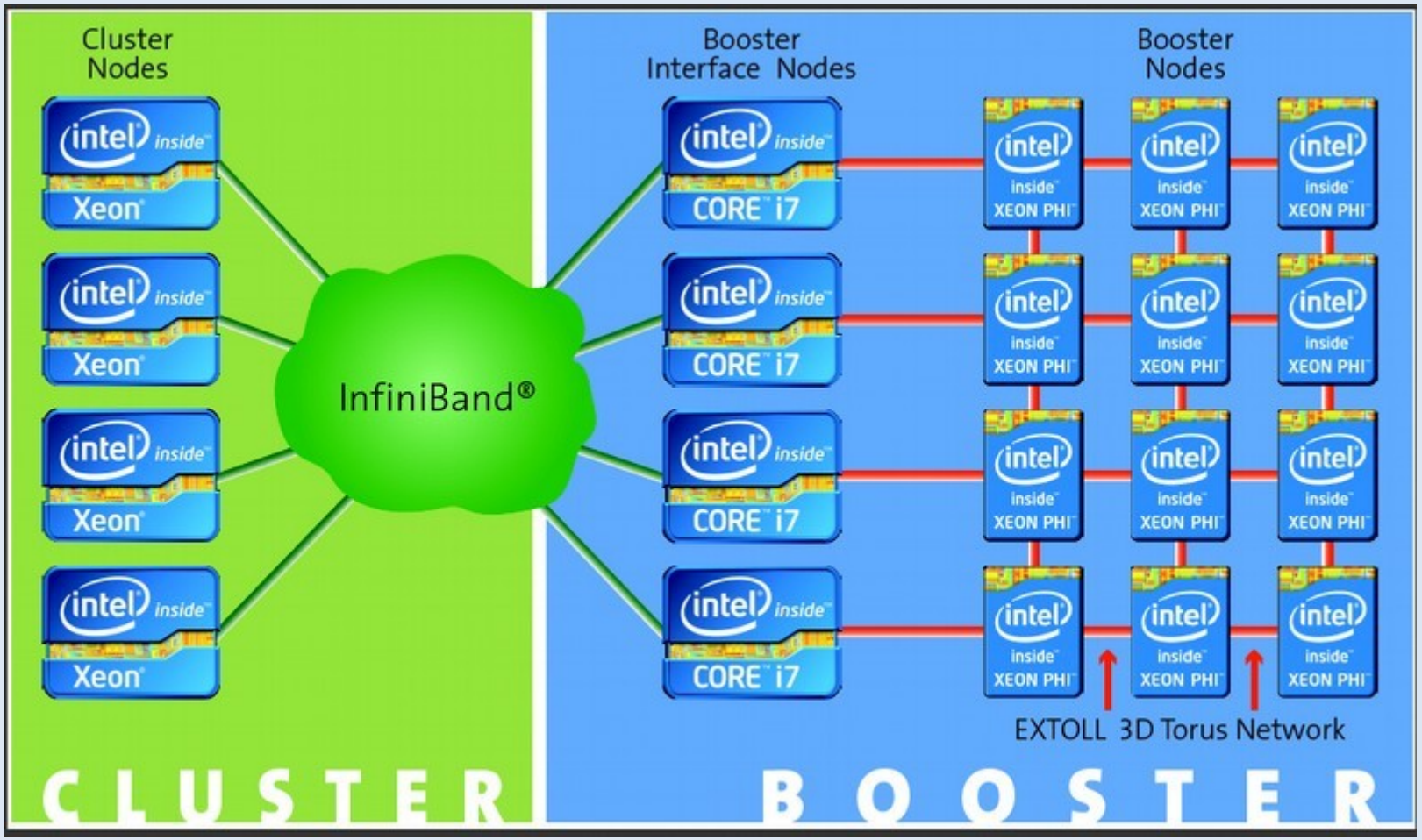
Accelerators not capable to act autonomously

Alternative Integration

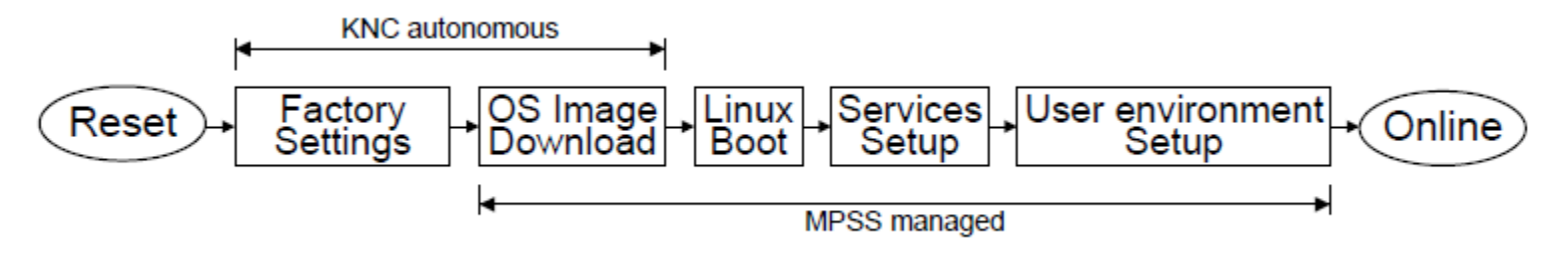


- Go for more capable accelerators (e.g. MIC)
- Attach all nodes to a low-latency fabric
- All nodes might act autonomously
- Dynamical assignment of cluster-nodes and accelerators
  - IB can be assumed as fast as PCIe besides latency
- Ability to off-load more complex (including parallel) kernels
 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

Hardware Architecture

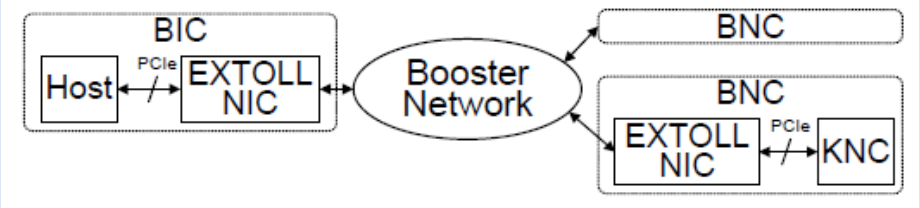


Booting KNC



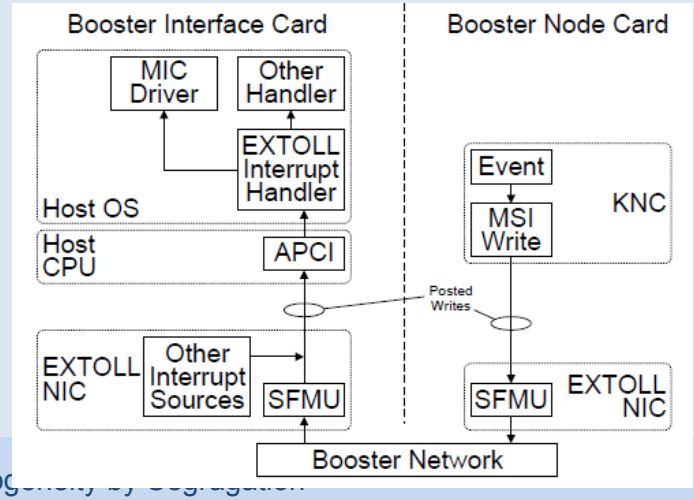
KNC is not self-booting

- Host processor required

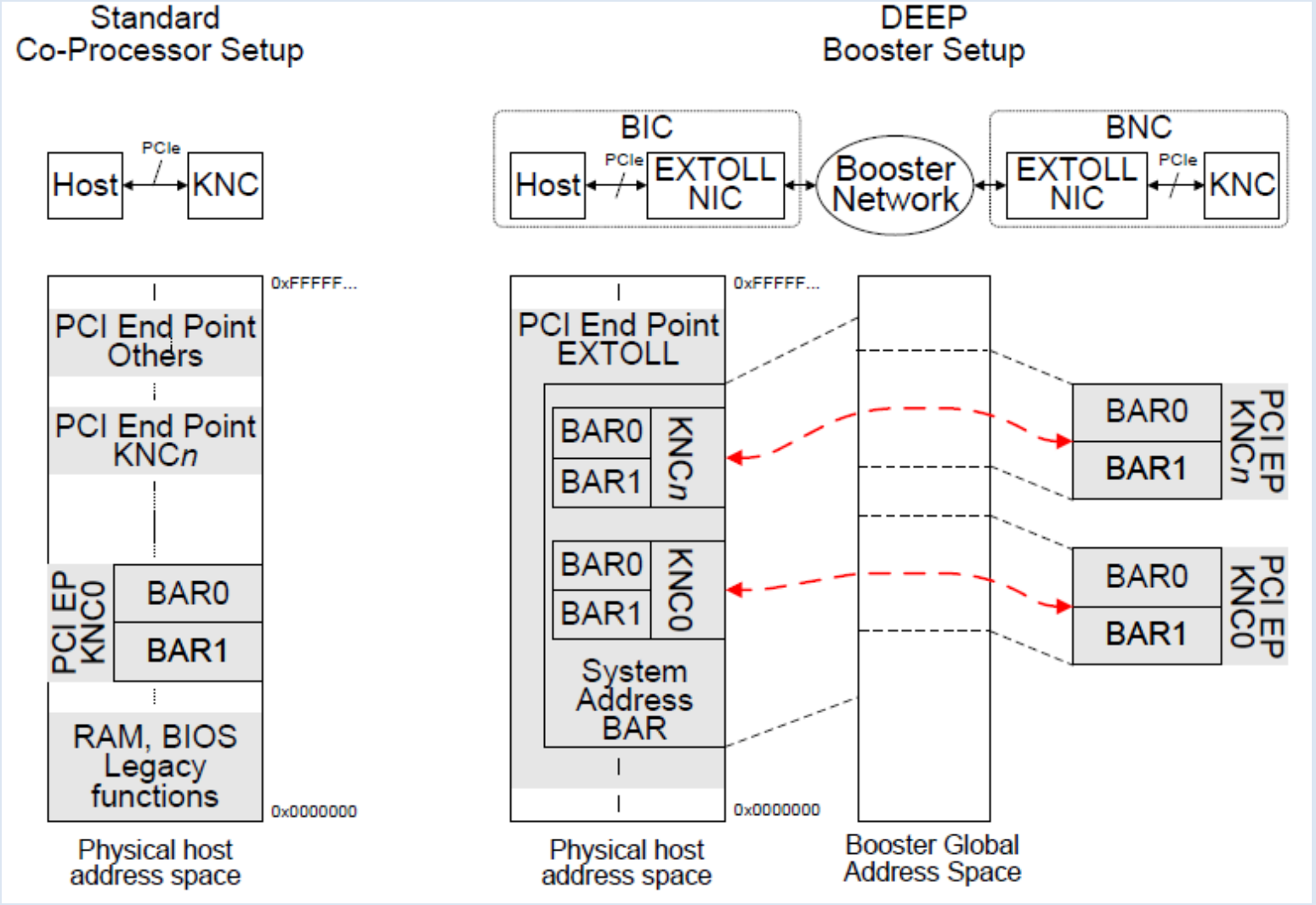


DEEP to extends BIC's PCIe via EXTOLL

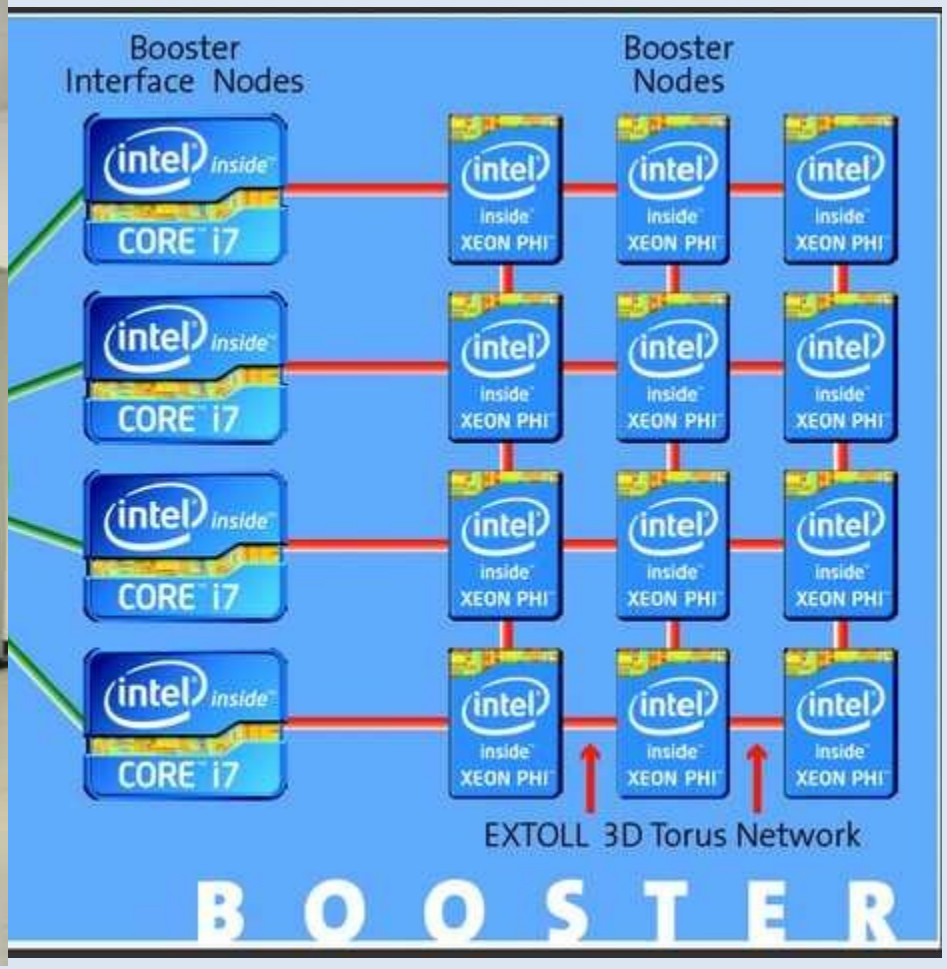
- BIC serves as host for 16 KNCs
- MPSS transparently forwarded
- All tools available



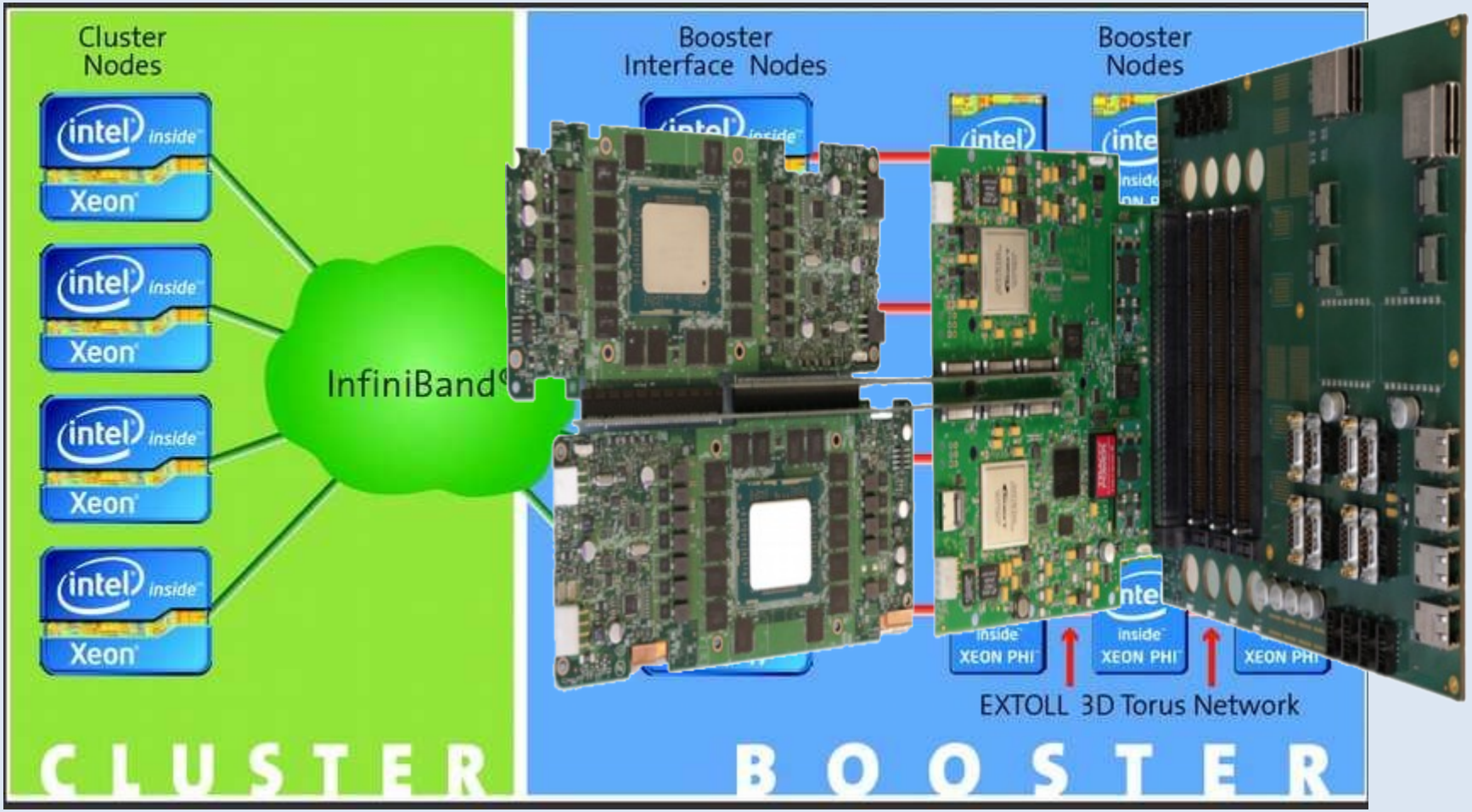
BIC address mapping



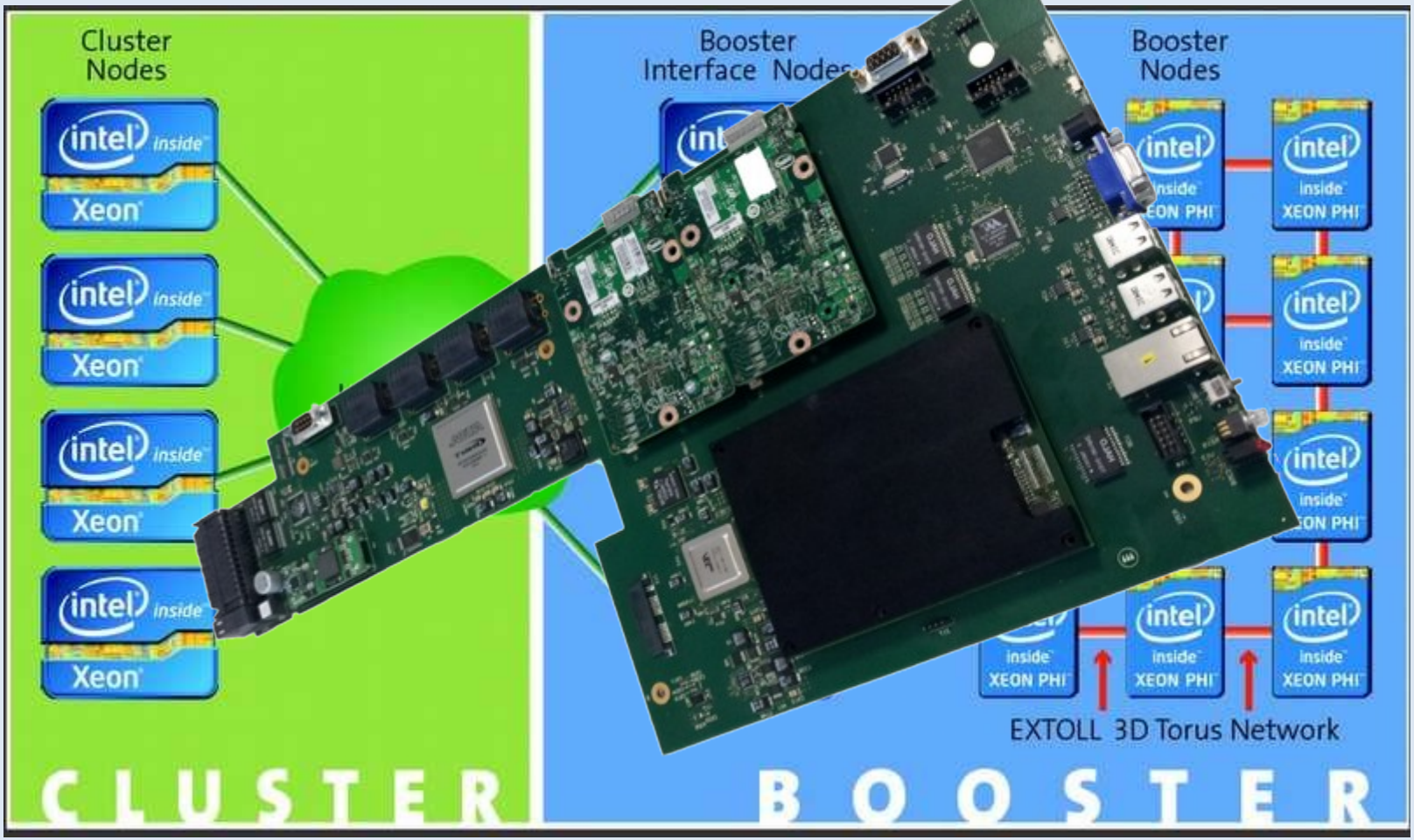
DEEP Hardware



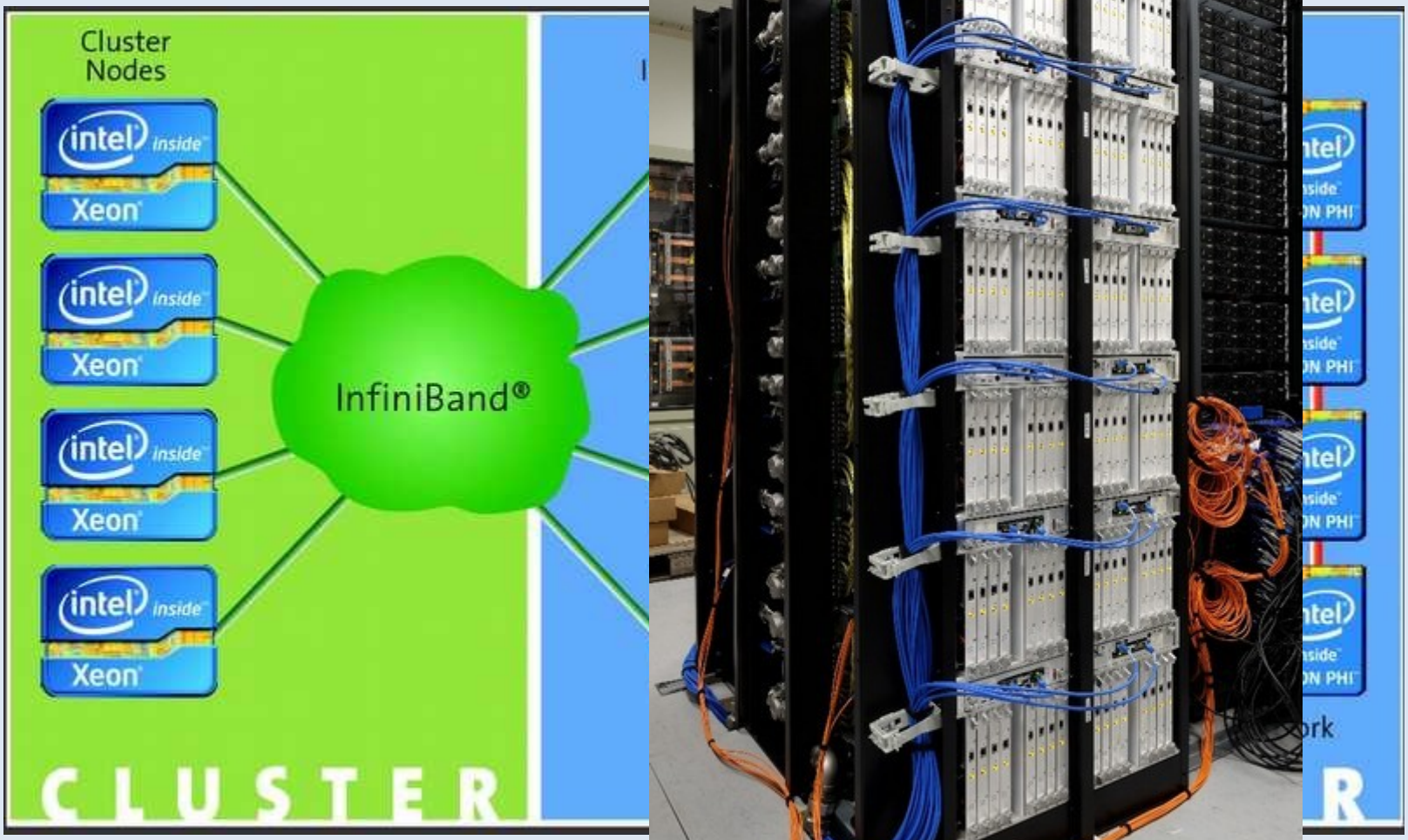
DEEP Hardware



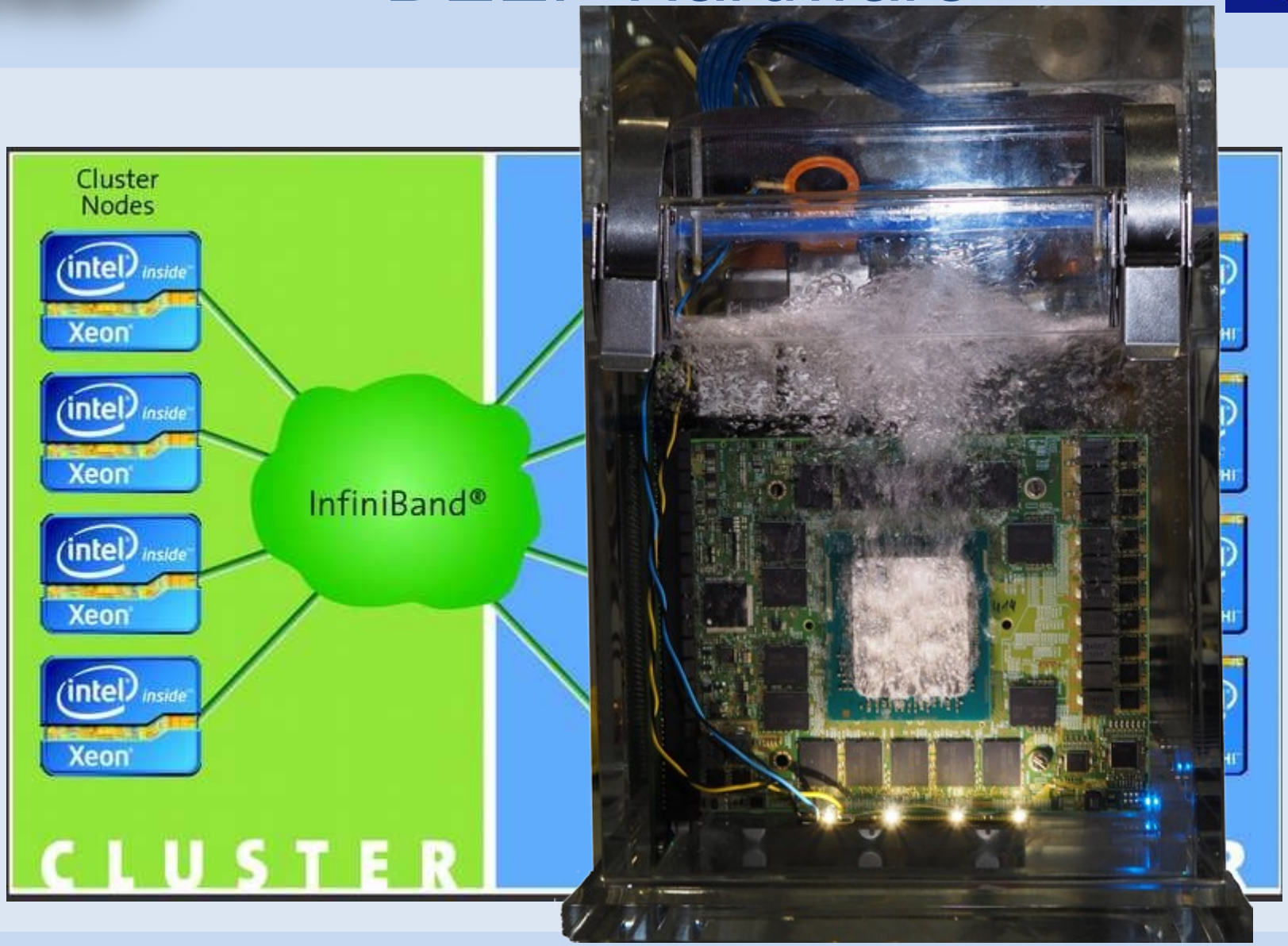
DEEP Hardware



DEEP


Cluster
Nodes
intel inside
Xeon
intel inside
Xeon
intel inside
Xeon
intel inside
Xeon
InfiniBand®
CLUSTER

DEEP Hardware

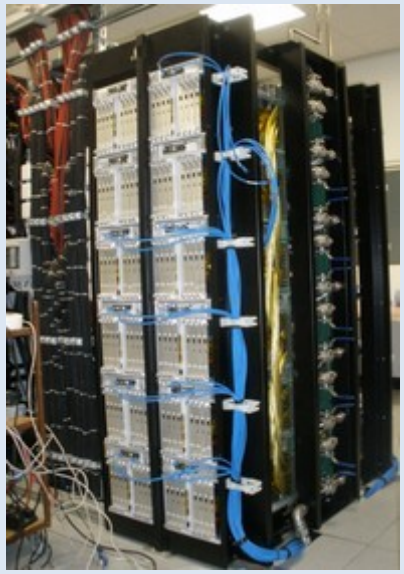


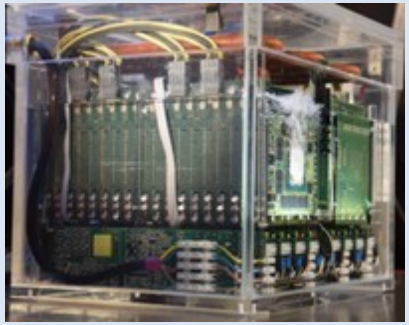
Hardware achievements

Design, construction and bring-up of 3 systems:

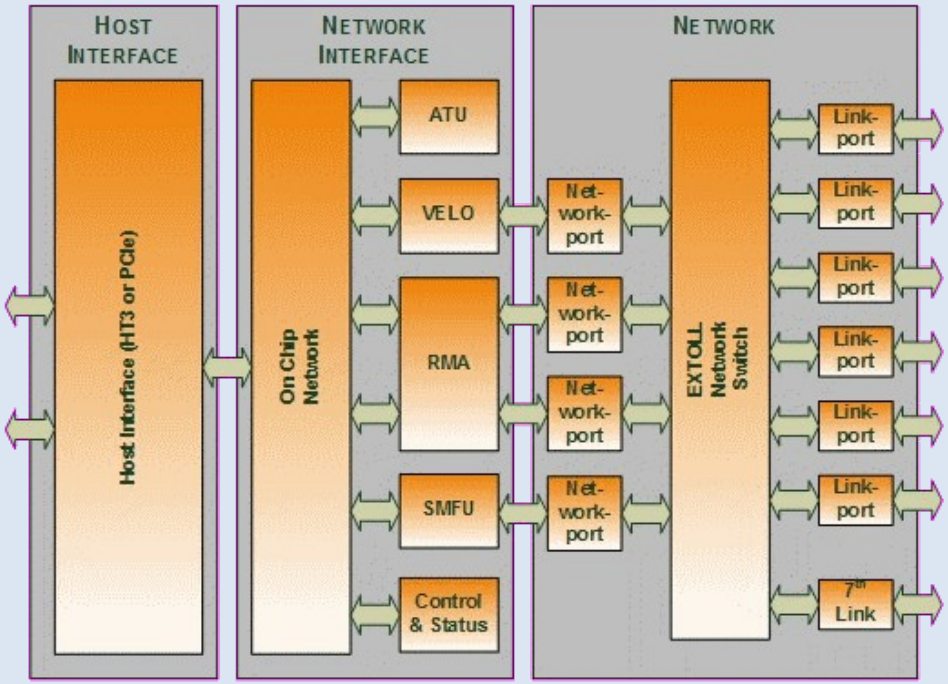
- **DEEP Booster** (Eurotech):
  - Largest Intel Xeon Phi system in Europe
  - Unique cluster of autonomous accelerators worldwide
  - Largest system with EXTOLL network
  - Scalable and energy efficient
  - Advanced monitoring capabilities
- **Energy Efficiency Evaluator** (Eurotech):
  - Test bed for energy efficiency experiments
- **ASIC Evaluator** (EXTOLL, GreenICE):
  - First EXTOLL TOURMALET ASIC platform
  - Test of innovative immersion cooling
  - Extreme component density

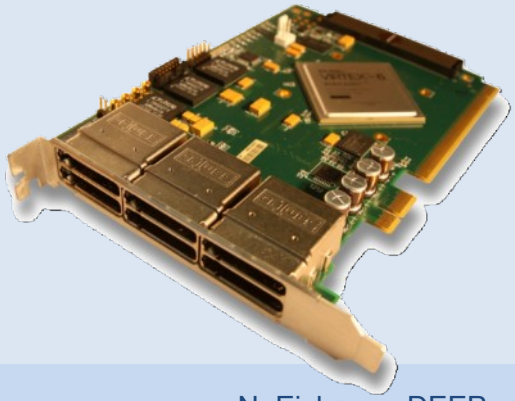
DEEP Booster





GreenICE Booster



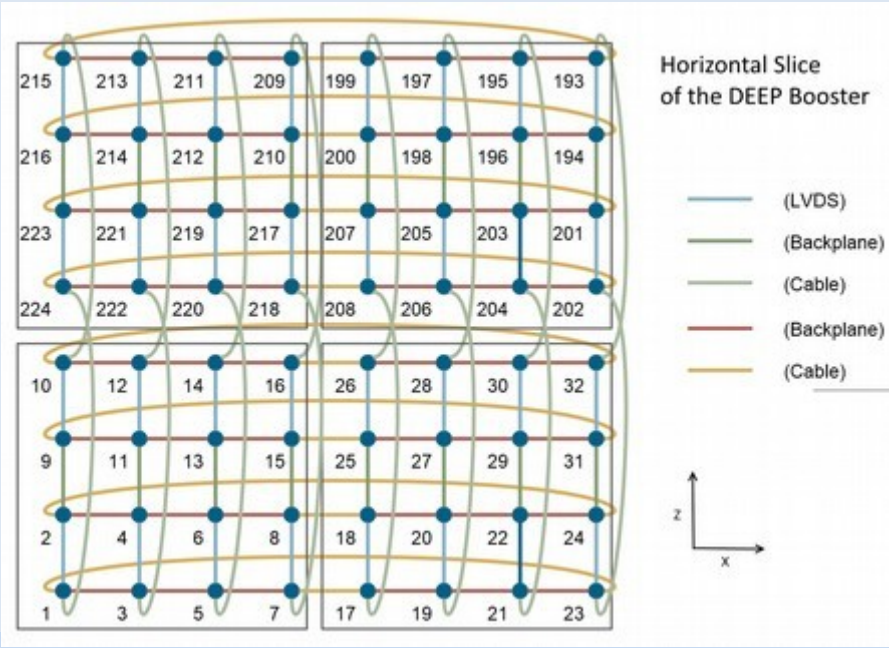


Relevant features for DEEP

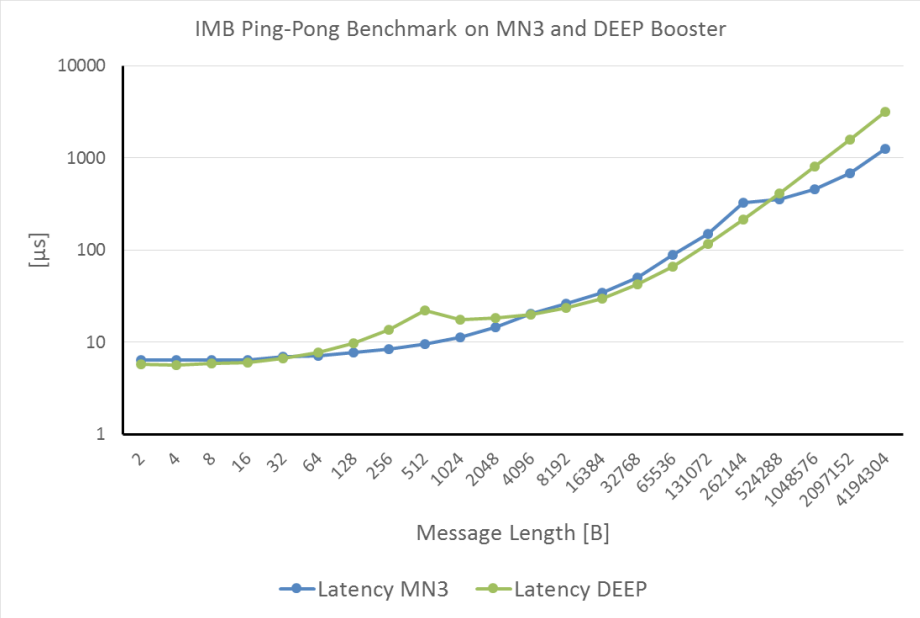
- Low latency, high bandwidth
- RMA engine for remote memory access, bulk data transfer
- VELO communication engine (zero-copy MPI)
- SMFU engine for bridging to InfiniBand
- 6 links for 3D torus topology
- 7th link for general devices
- Built-in PCIe root-port
- RAS features: CRC/ECC protection, link level retransmission
- Many status & control registers
- Access from host, via I2C bus or over EXTOLL

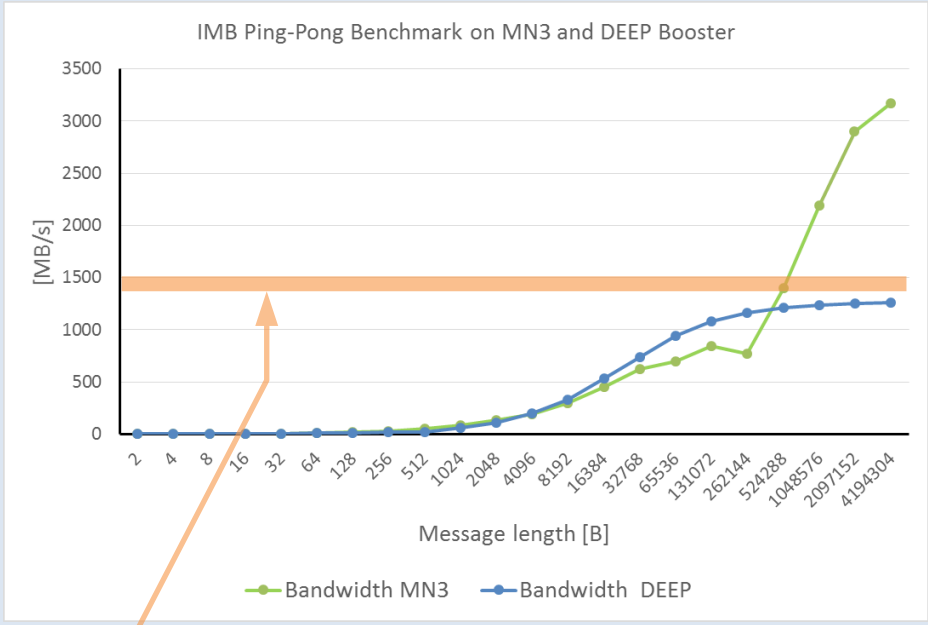
Topology and Cabling

- System configured as a 8×6×8 torus
  - X direction: within chassis
  - Y direction: between horizontal “slices”
  - Z direction: within BNC & chassis and between back & front chassis
- All non-LVDS links first go to backplane
- Many links travel via cable
  - 25 % of X ands Z links
  - 100% of Y links
  - Cable lengths do vary



Booster – MPI performance



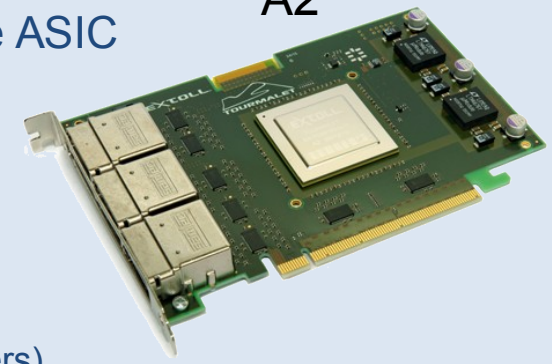


FPGA Bandwidth Limit around 1.4 GB/s

EXTOLL TOURMALET

- EXTOLL Tourmalet ASIC A2 stepping completed and tested
  - Fully validated up to 5 Gbit/s per lane (60 Gbit/s per link)
  - Functional up to 8 Gbit/s per lane (100 Gbit/s per link)
    → 80% of design performance
  - Full validation at that speed work in progress
- EXTOLL TOURMALET NIC
  - First version working for 5 Gbit/s per lane and used in the ASIC Evaluator
  - New board designed with advanced PCB technology
  - Board under validation for 8 Gbit/s per lane
    - PCI Express generation 3 speed achieved
    - EXTOLL link speed work in progress (tuning of SERDES parameters)
  - Plan: have the NIC with full speed (100 Gbit/s per link) available in Q1/2016

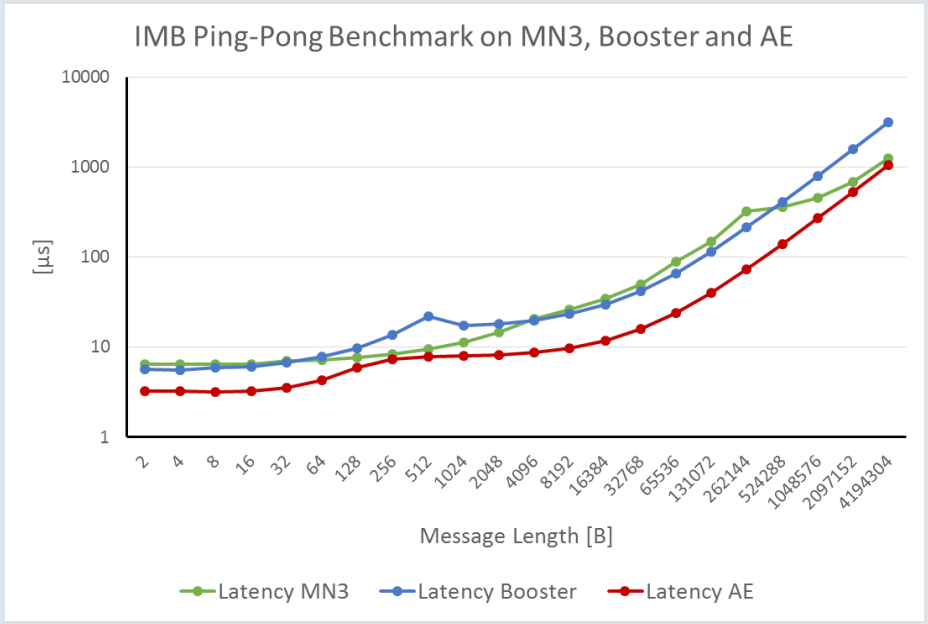
A2

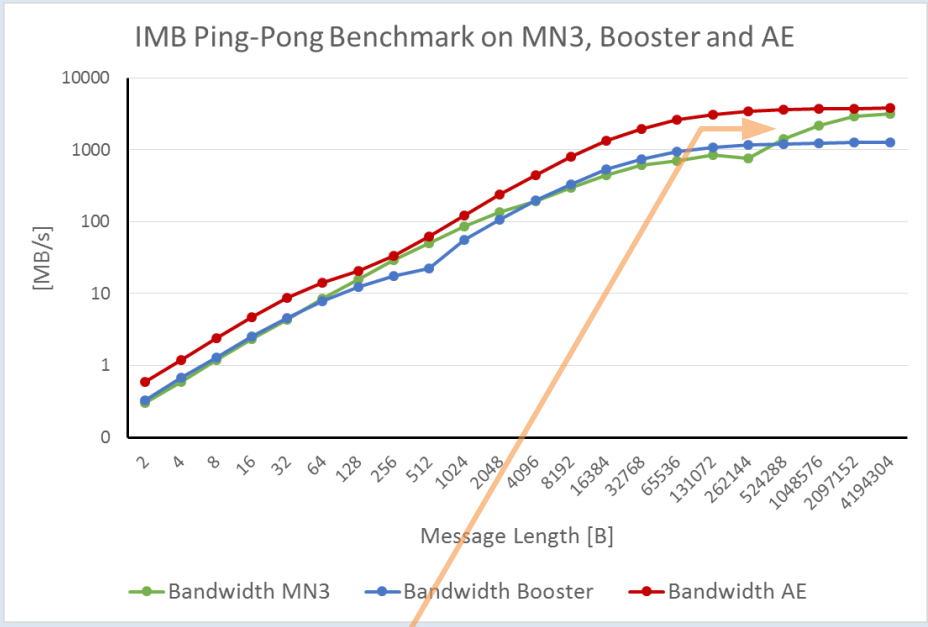


ASIC Evaluator Status

- System installed at Juelich
  - 16 Booster nodes (KNC 7120D)
  - EXTOLL TOURMALET with 5 Gbit/s per lane speed
- System status
  - Nodes operational
  - Low-level interconnect and driver problems addressed
  - Ready for system SW integration

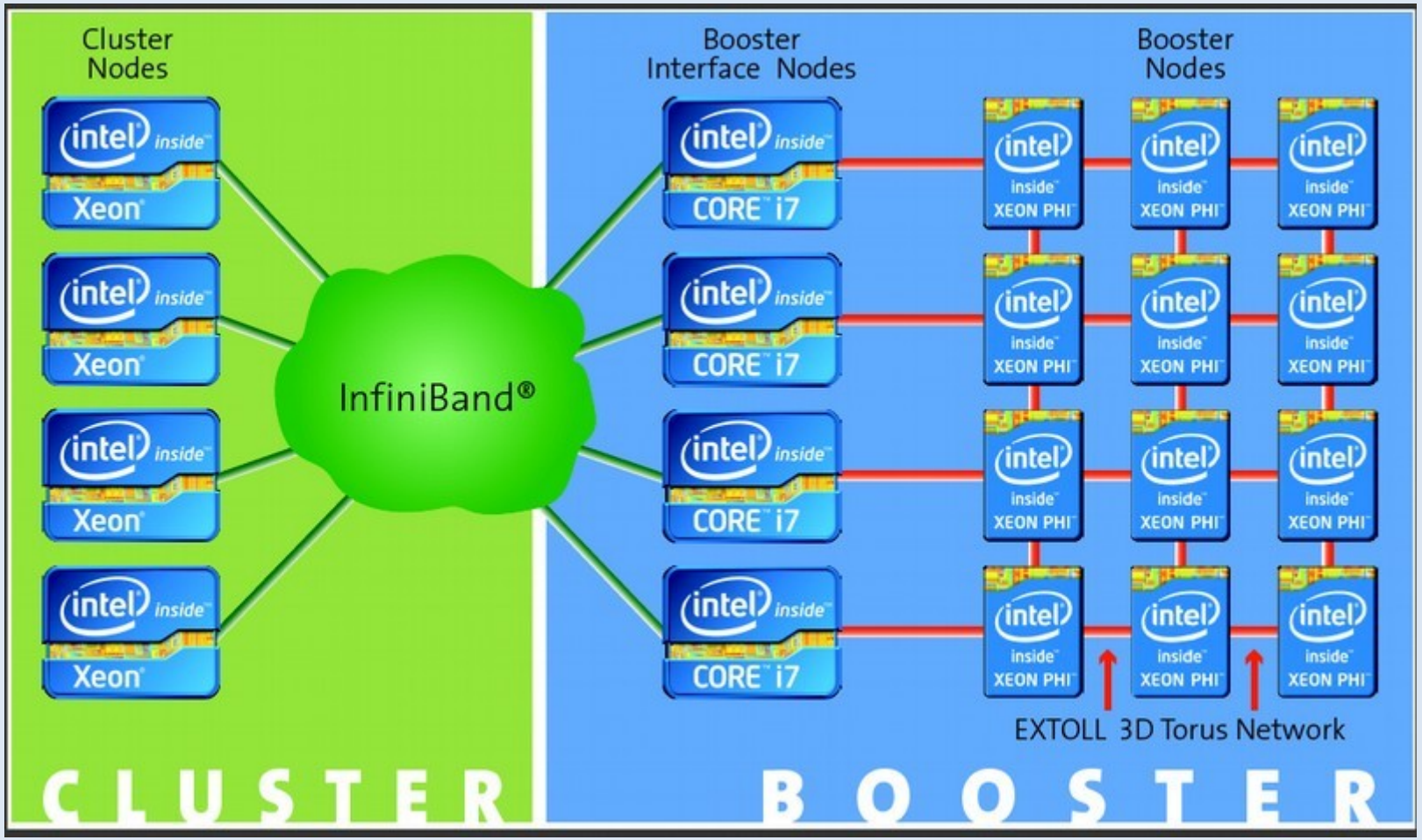
AE – MPI performance



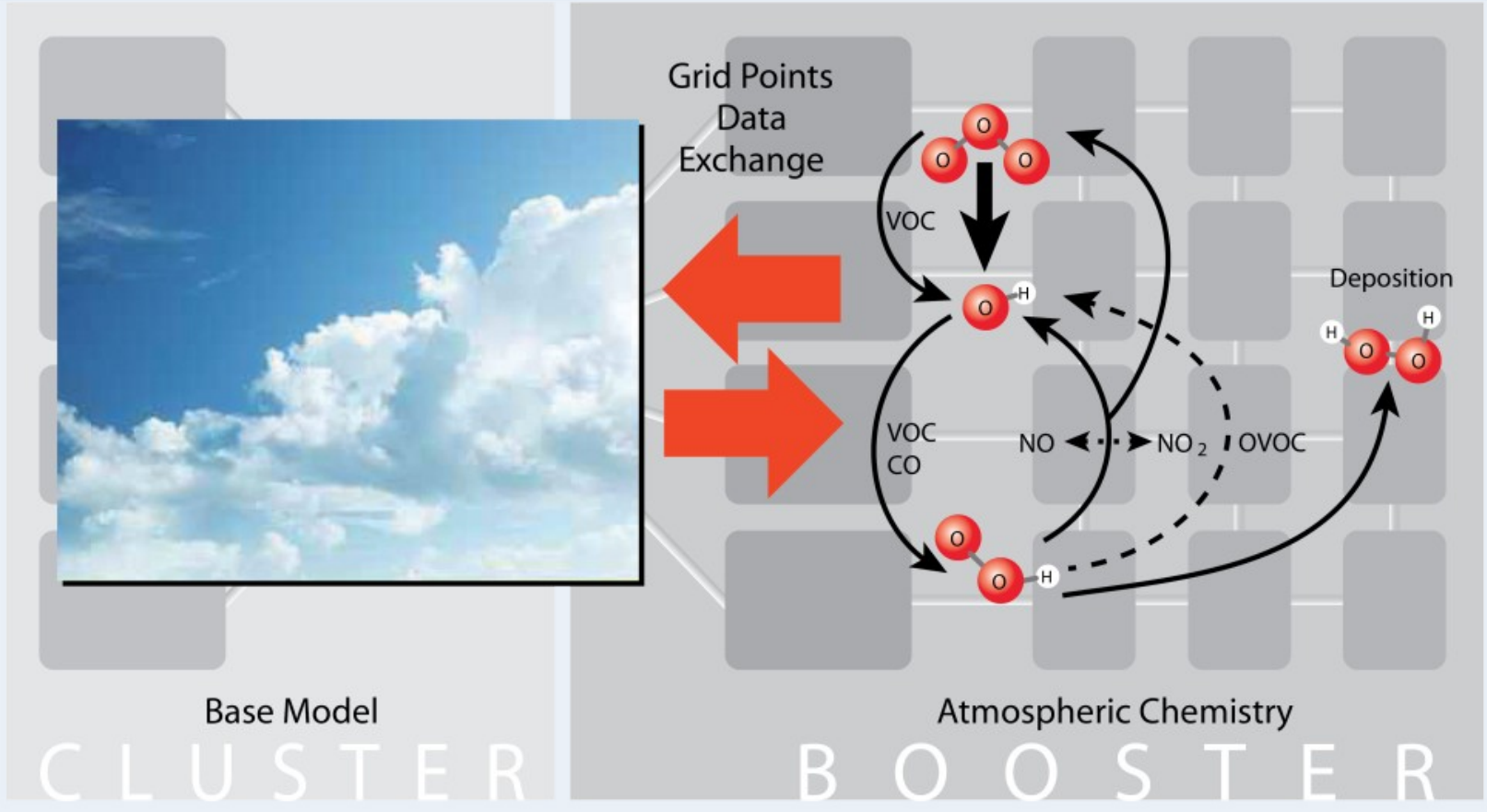


Factor of 3× achieved by TOURMALET(5Gbit/s version)

Hardware Architecture

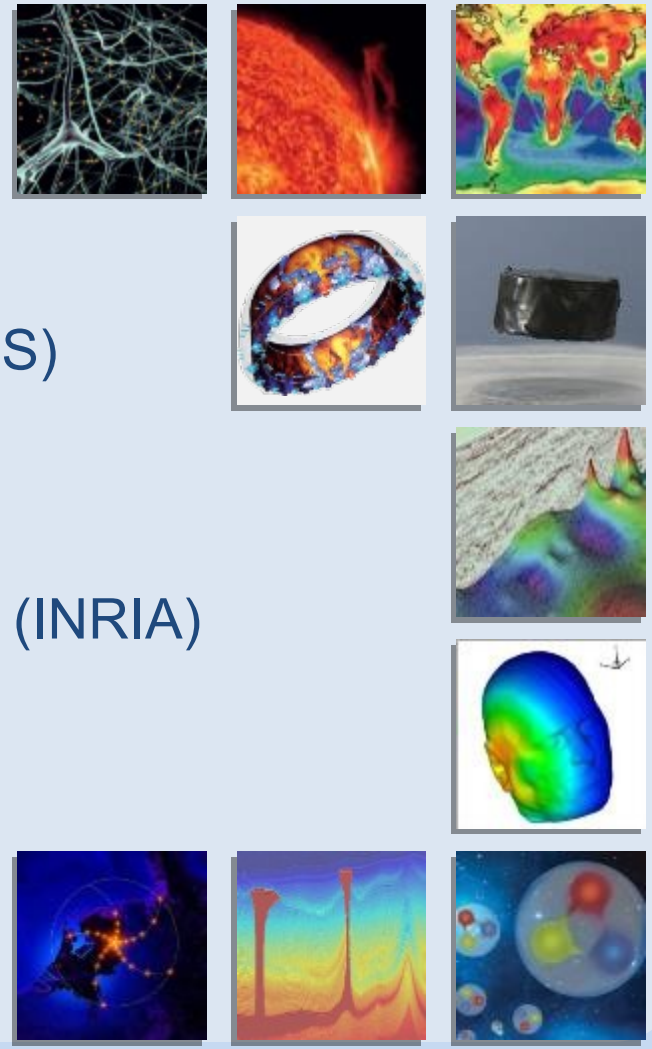


CyI – Climate Simulation



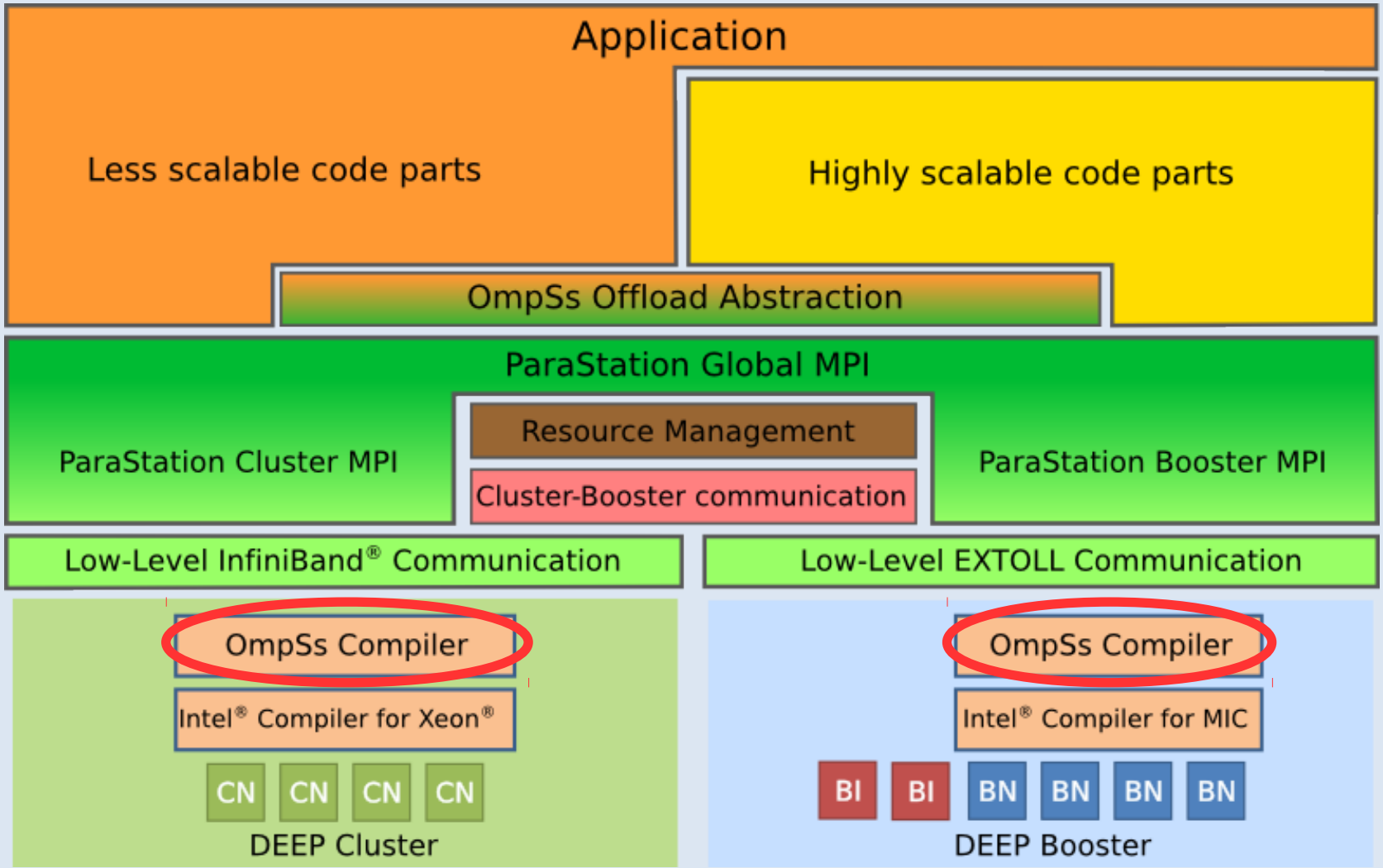
Guiding Applications

- 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)



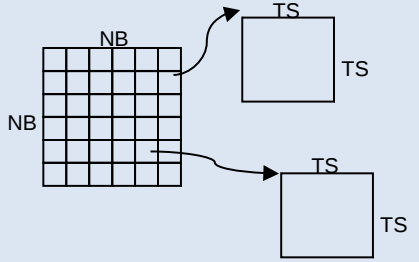
Programming Paradigm

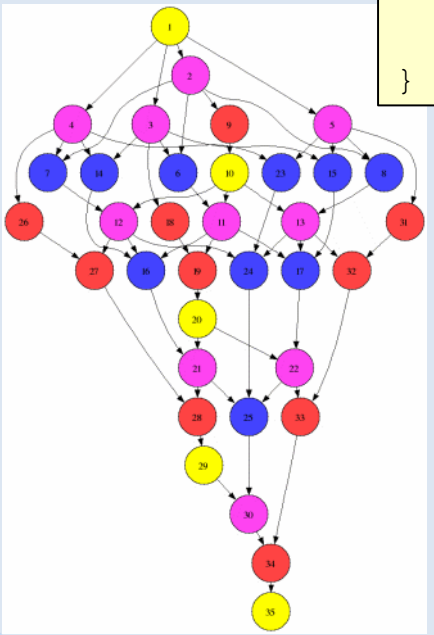
Software Architecture



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]);
   }
}
```

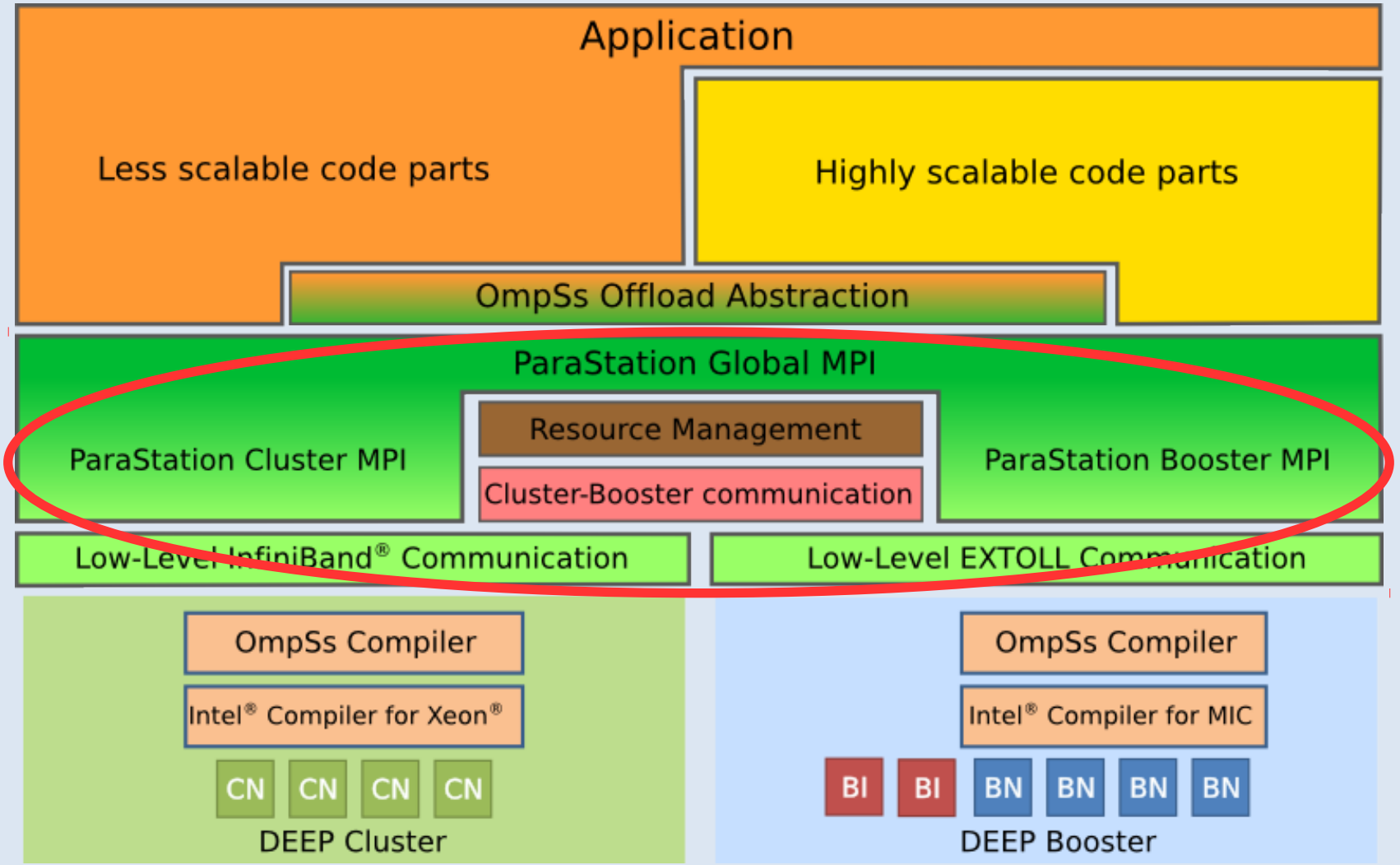




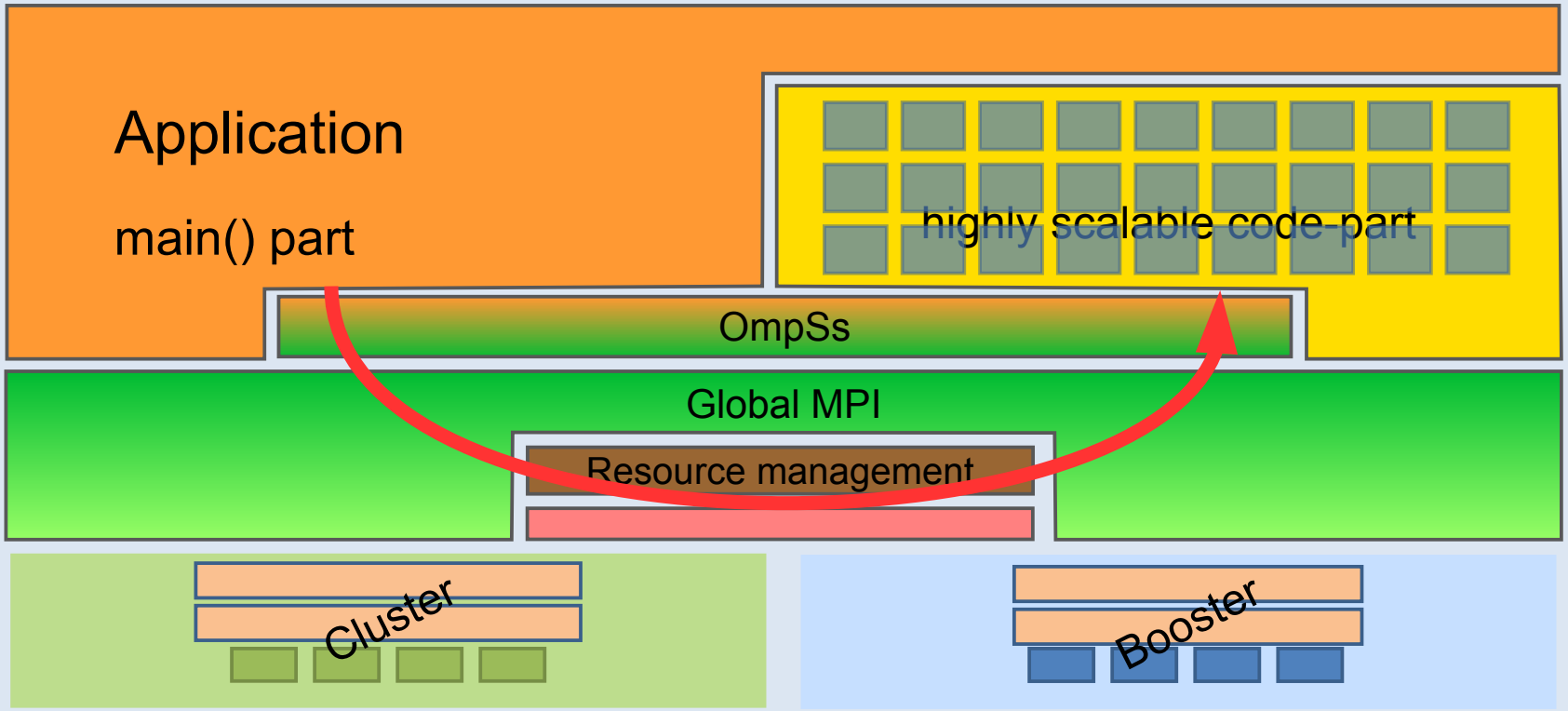
```
#pragma omp task inout ([TS][TS]A)
void spotrf (float *A);
#pragma omp task input ([TS][TS]T) inout ([TS][TS]B)
void strsm (float *T, float *B);
#pragma omp task input ([TS][TS]A,[TS][TS]B) inout ([TS][TS]C)
void sgemm (float *A, float *B, float *C);
#pragma omp task input ([TS][TS]A) inout ([TS][TS]C)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

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

Software Architecture



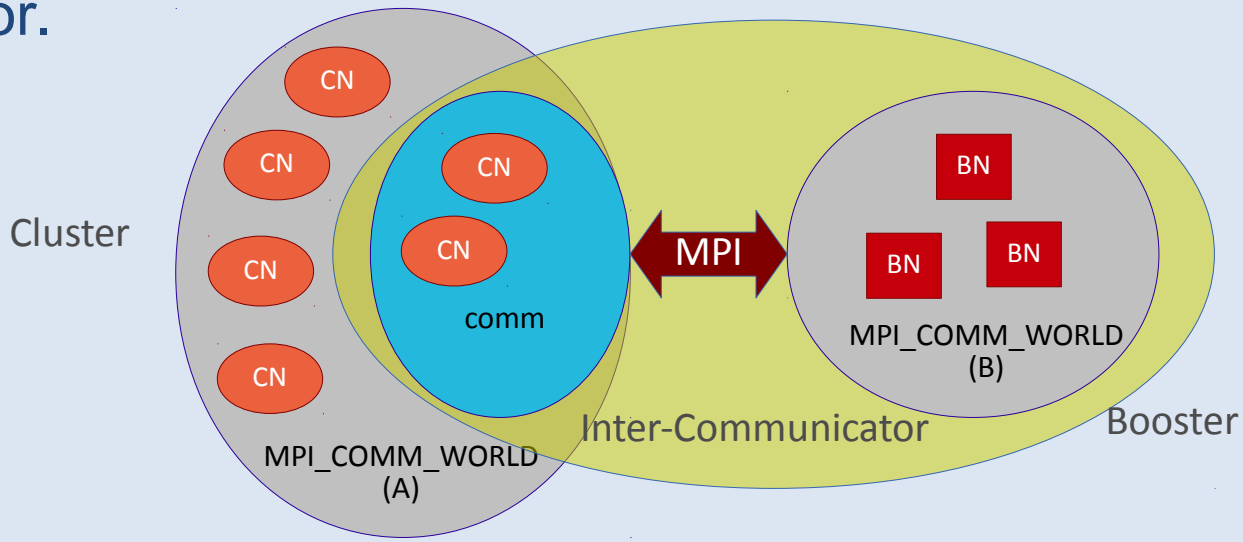
Application Startup



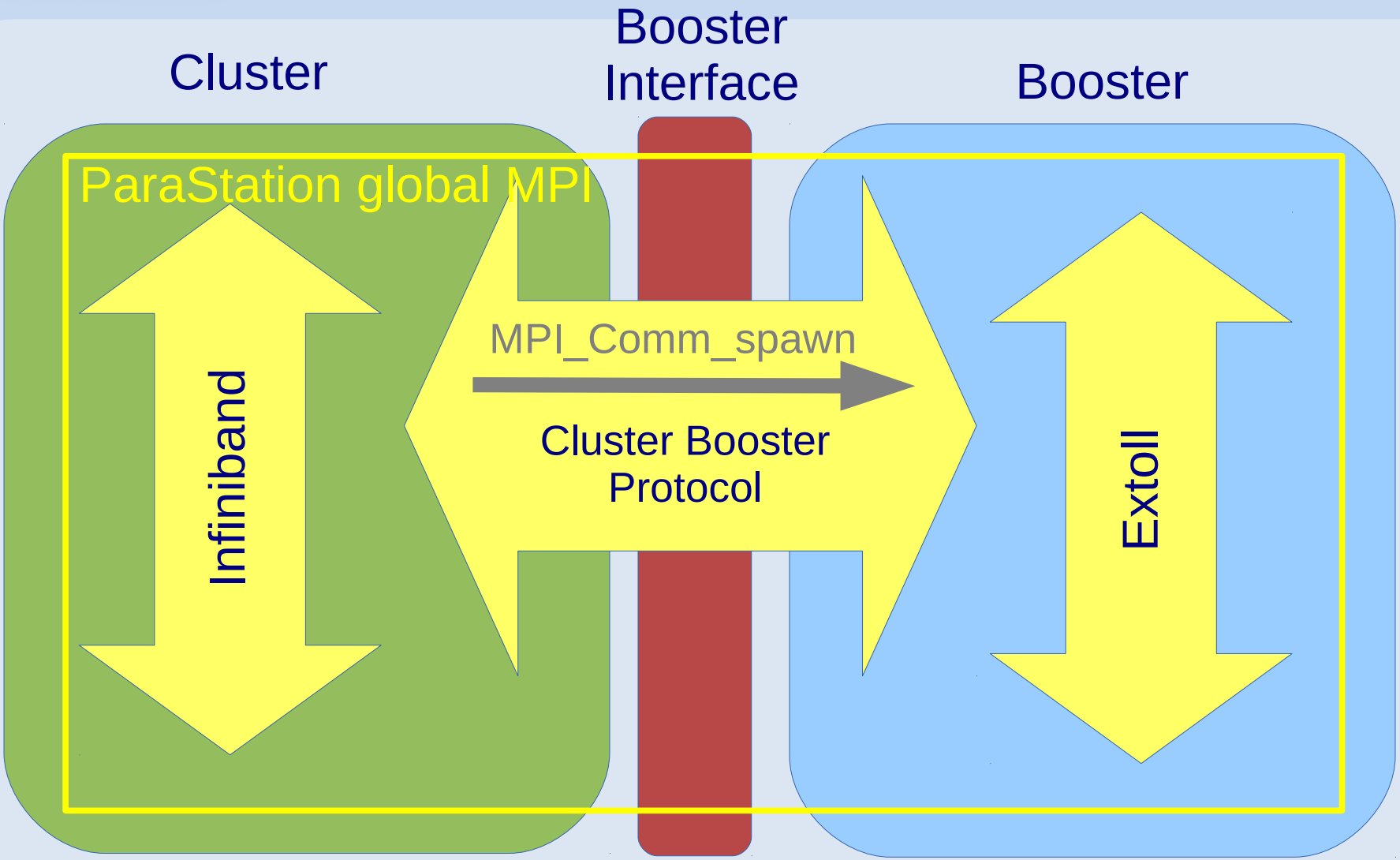
- Application's main()-part runs on Cluster-nodes (CN) only
- Actual spawn done via global MPI
- OmpSs acts as an abstraction layer
- Spawn is a collective operation of Cluster-processes
- Highly scalable code-parts (HSCP) utilize multiple Booster-nodes (BN)

MPI Process Creation

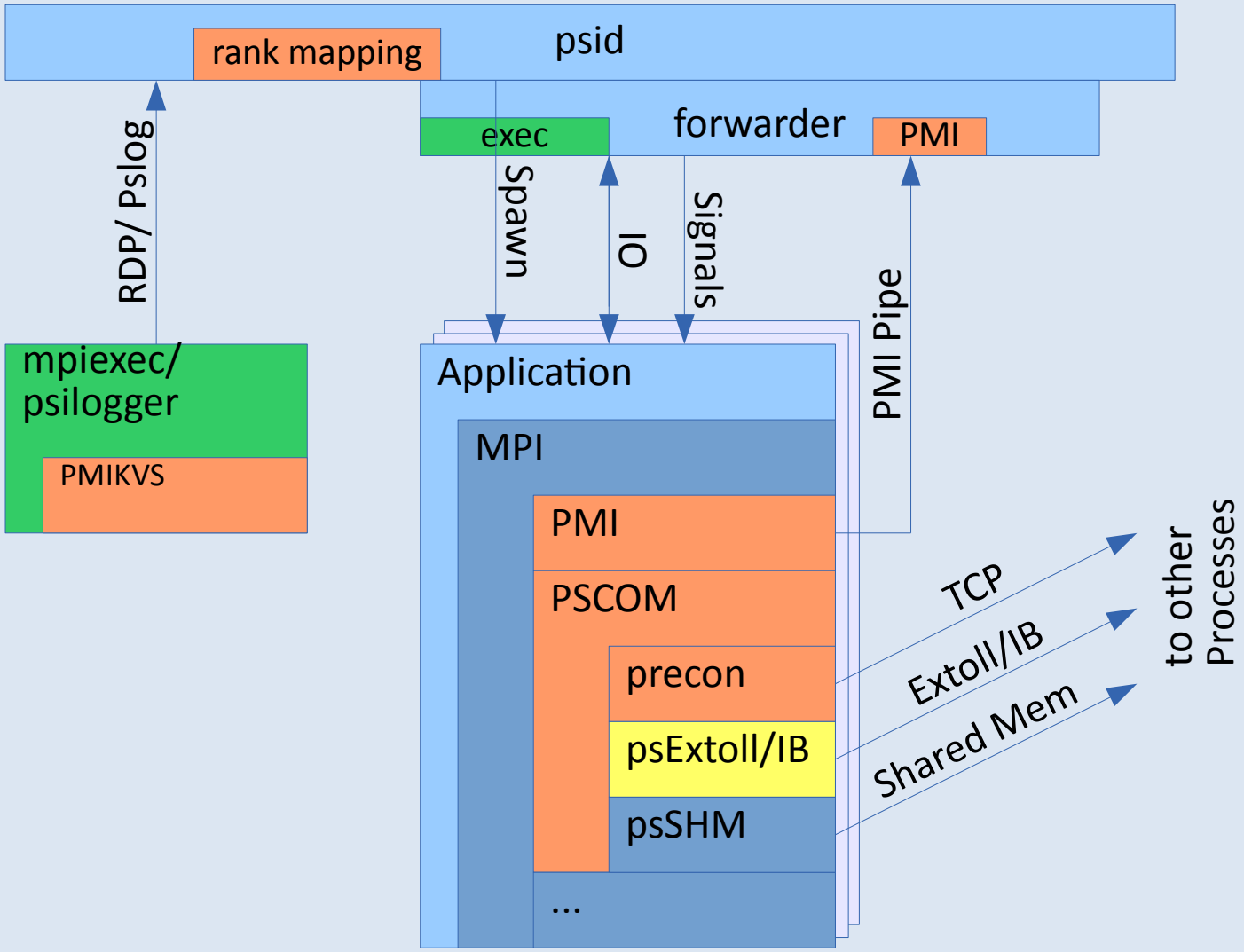
- The inter-communicator contains all parents on the one side and all children on the other side.
  - Returned by MPI_Comm_spawn for the parents
  - Returned by MPI_Get_parent by the children
- Rank numbers are the same as in the the corresponding intra-communicator.



Programming

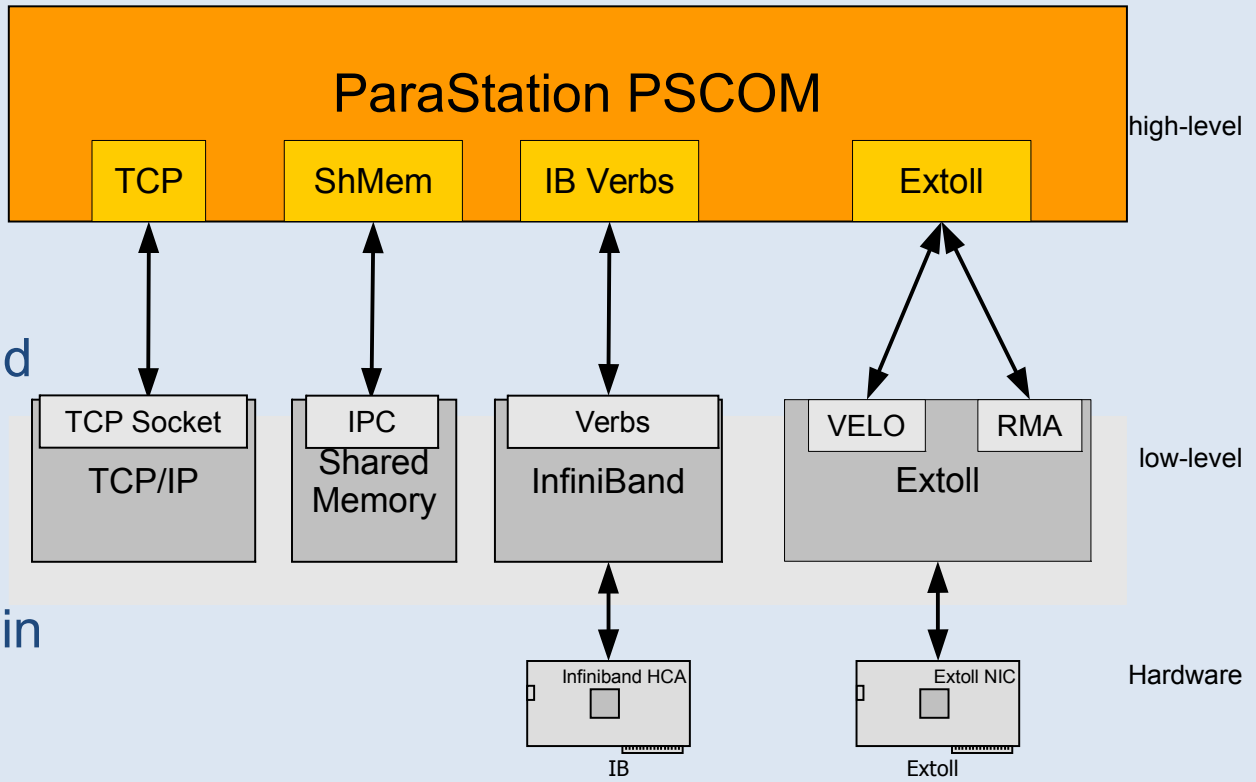


ParaStation MPI

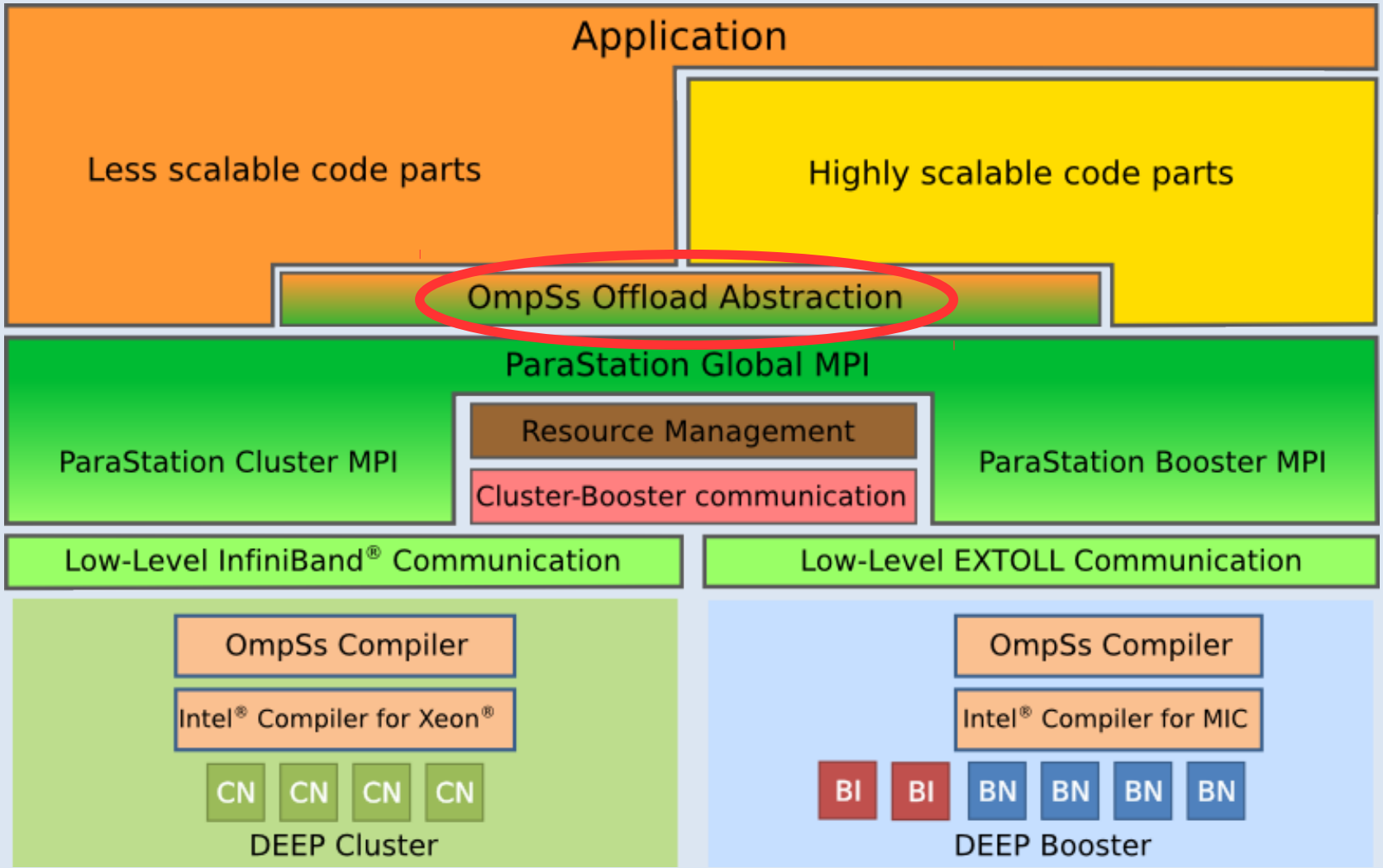


ParaStation pscom

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



Software Architecture



OmpSs Offload Abstraction

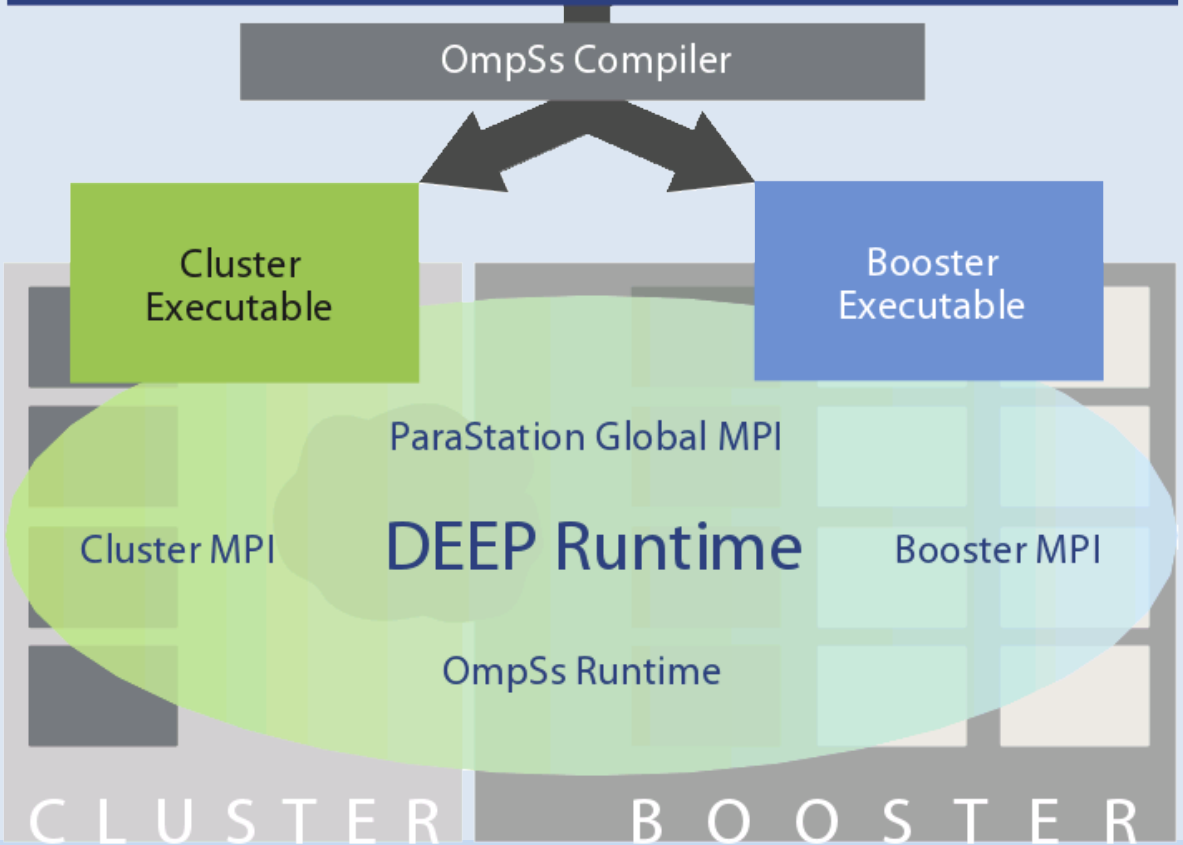
Source Code

Compiler

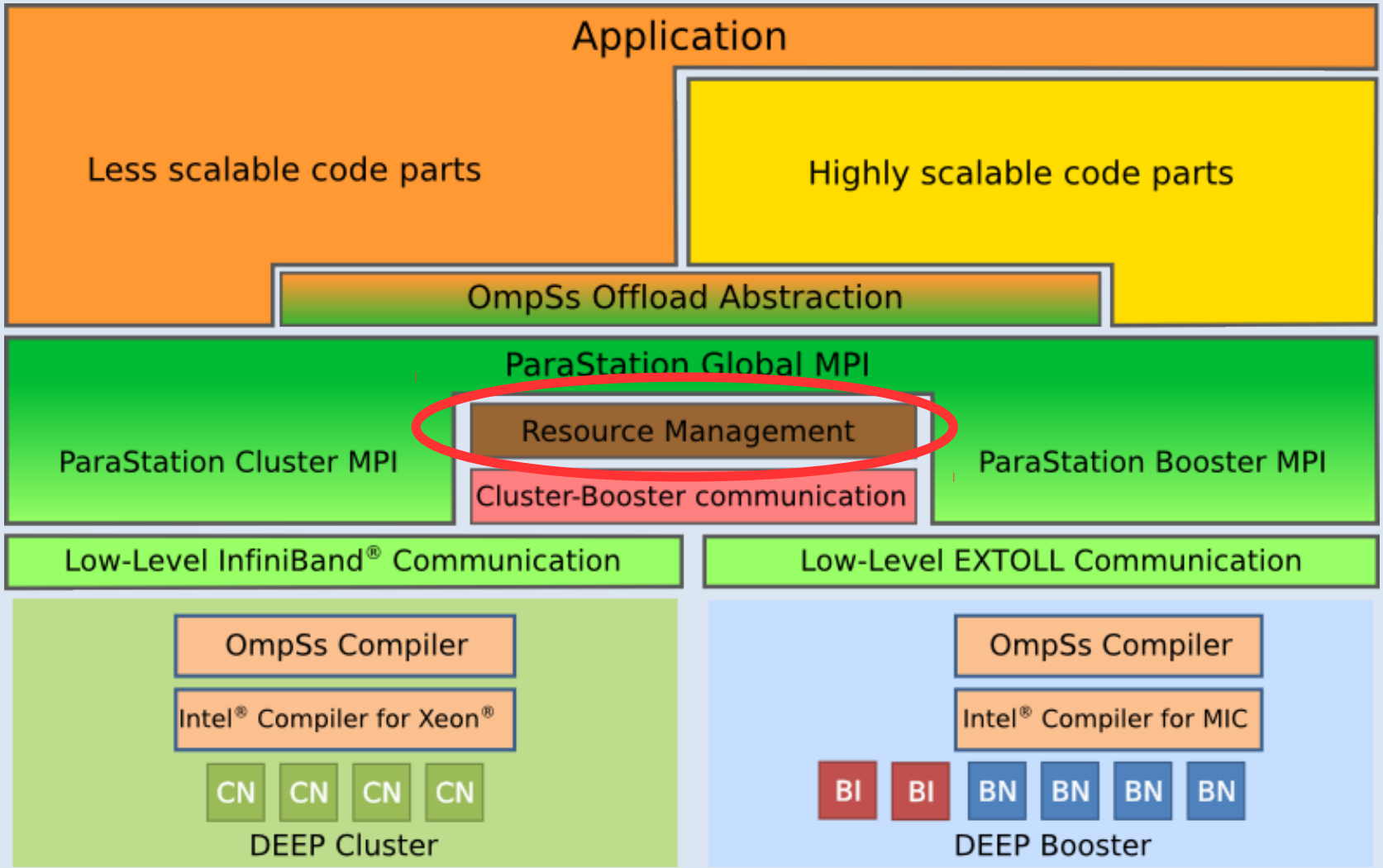
Application Binaries

DEEP Runtime

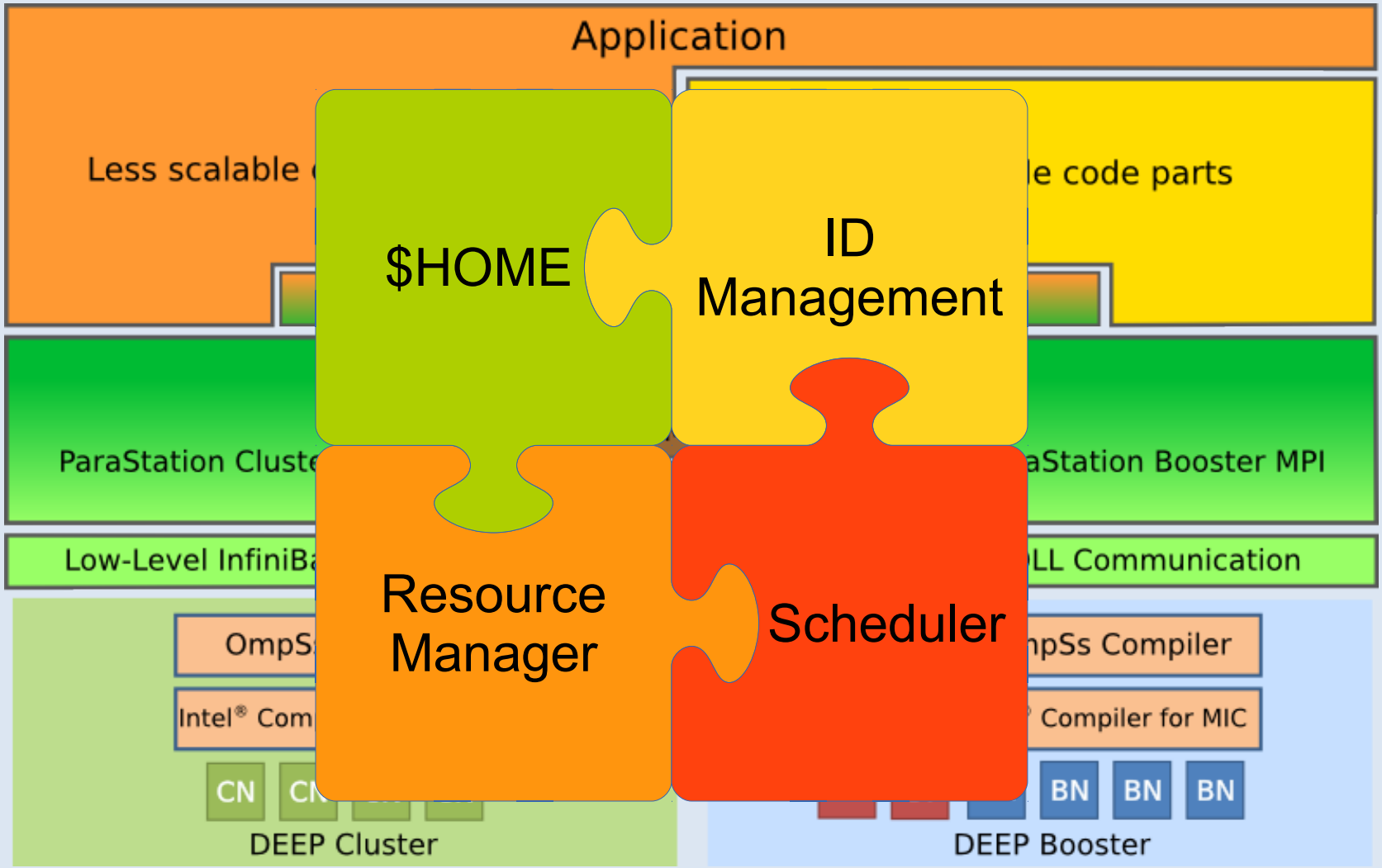
```
int main(int argc, char *argv[]){
    /*...*/
    for(int i=0; i<3; i++){
        #pragma target device (comm:size*rank+i) copy_deps
        #pragma omp task input(…) output(…)
        foo_mpi(i, …);}}
```



Software Architecture



Software Architecture



User Environment

- LDAP for ID management
  → MPSS 3.4.3 to support this
- Some file-system for $HOME
  → Brought to BNs via NFS
- Resource Manager
  → psmom (part of the ParaStation psmgmt)
- The actual batch system
  → Torque/Maui with modifications

User are provided with an environment known from JSC's production systems

Results

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

DEEP Advantages

1. Dynamic ratio of processors/coprocessors
   - Each process in the Cluster has its own set of coprocessors in the Booster (determined at runtime)
   - Example: Seismic imaging
   - On DEEP, applications that rely mostly on accelerators do not have the host processors idling a significant amount of time

Node:Coprocessor ratio in most systems	
1:1	Common
1:2	Common
1:4	Rare
1:8	Very rare (and expensive)

Node:Coprocessor ratio in DEEP	
1:0	Assigned dynamically depending on the user needs
1:1	
…..	
1:384	

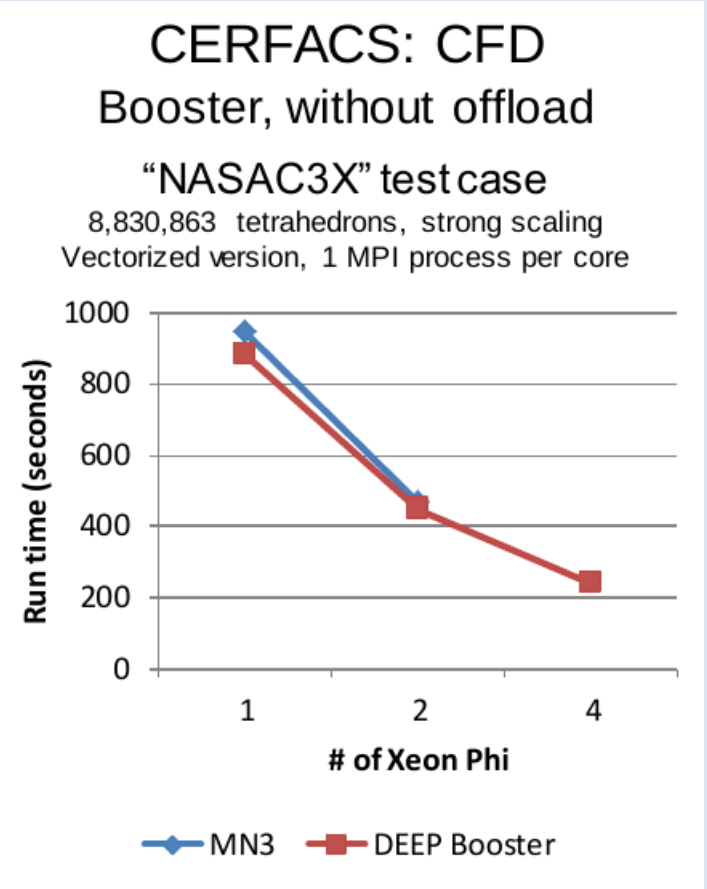
DEEP Advantages

2. Use Booster as pool of accelerators (globally shared)
   - All the processes in the Cluster have access to all the coprocessors in the Booster
   - Example: Climate Simulation
   - Helps to alleviate load imbalance caused by the local chemistry
     - Processes with more load can offload to whatever Booster node is free
   - Estimations based on measured data (offload rate to Xeon Phis and “offloadable” time) show good predictions
     - Better with larger resolutions

DEEP Advantages

3. Discrete use of the Booster

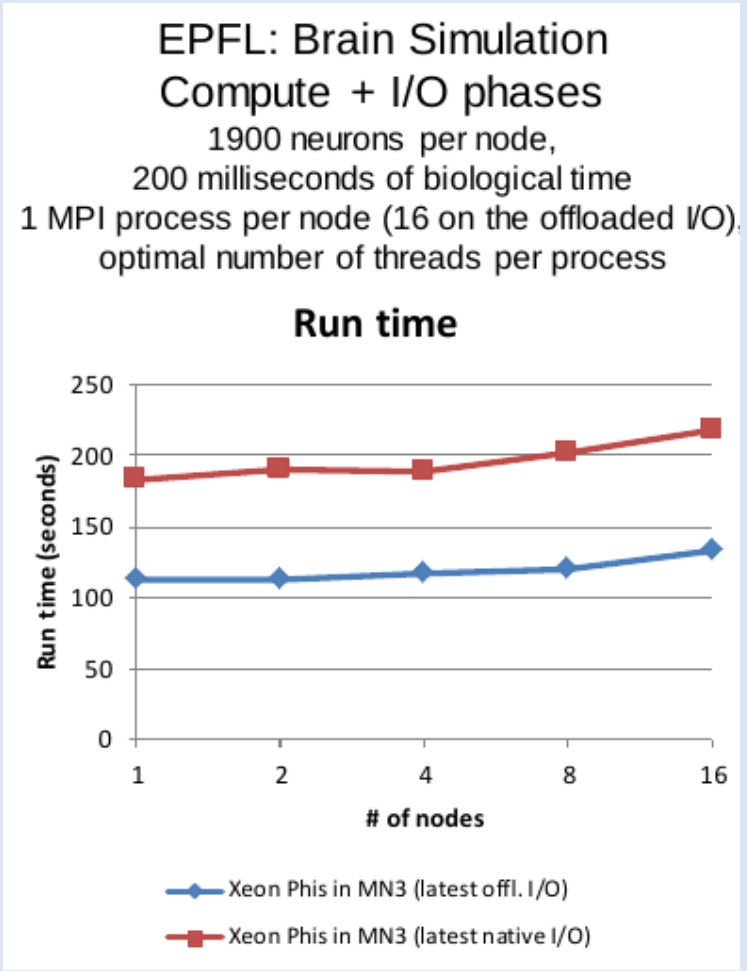
- Autonomous use of the Xeon Phis
- DEEP is currently the closest alternative to future systems like Cray's Cori (KNL) and Aurora (KNH)
  - These systems will also have "Cluster nodes"
- Some applications do not need anything else
  - MC and QCD applications are an example



DEEP Advantages

4. Discrete use + I/O offload

- Autonomous use using the Cluster as I/O proxy
- Example: Brain Simulation
- Example: Computational Fluid Dynamics
- Good results on Brain Simulation
  - Will get more meaningful in the future with “interactive supercomputing”
- Likely to show good results on CFDs
  - Diverse technical issues and lack of developers impacted the benchmarking

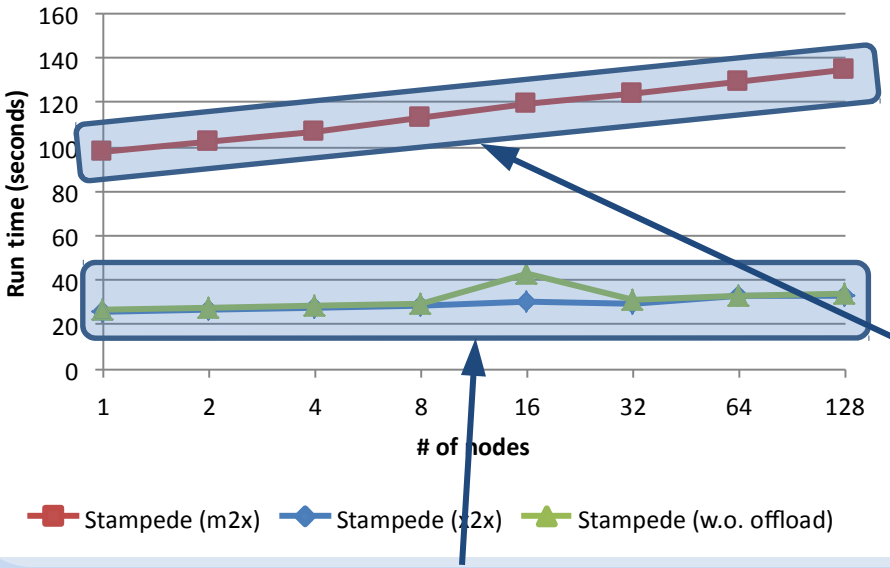


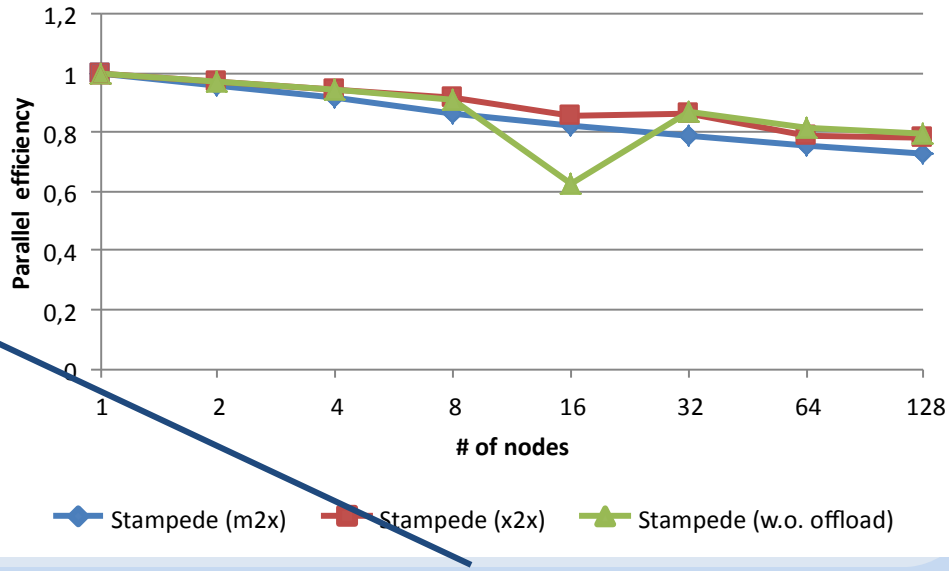
DEEP Advantages

5. Specialized symmetric mode

- Mix of Cluster and Booster nodes without “hierarchical” relationship, but each part of the system runs a specific part of the application
- Example: Space Weather
- The code division enables new possibilities (reduced noise by using large number of particles)

256x256 cells per node, 5x5x5x2 particles per cell, weak scaling
16 MPI processes per node, 16 MPI processes per Xeon Phi, 15 threads per MPI process on Xeon Phi
Xeon Phi to Xeon offload (m2x), and Xeon to Xeon offload (x2x)





The code division does not add overhead

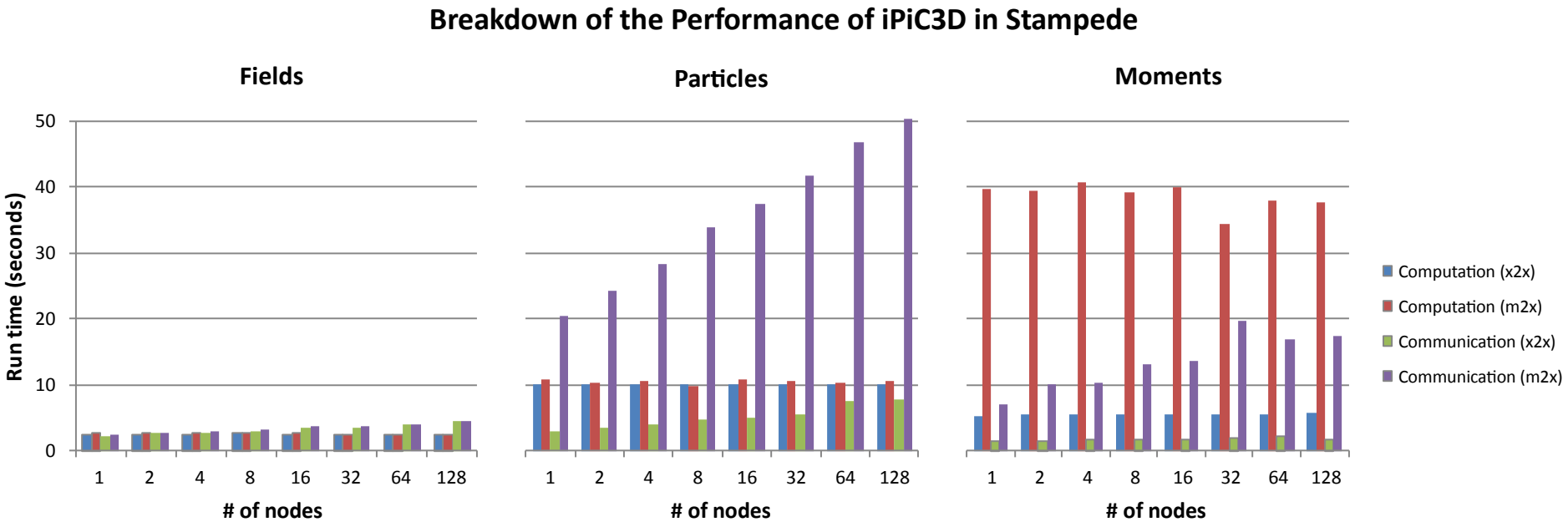
Analyzed in next slide

DEEP Advantages

5. Specialized symmetric mode

- Mix of Cluster and Booster nodes without “hierarchical” relationship, but each part of the system runs a specific part of the application
- Example: Space Weather
- The code division enables new possibilities (reduced noise by using large number of particles)

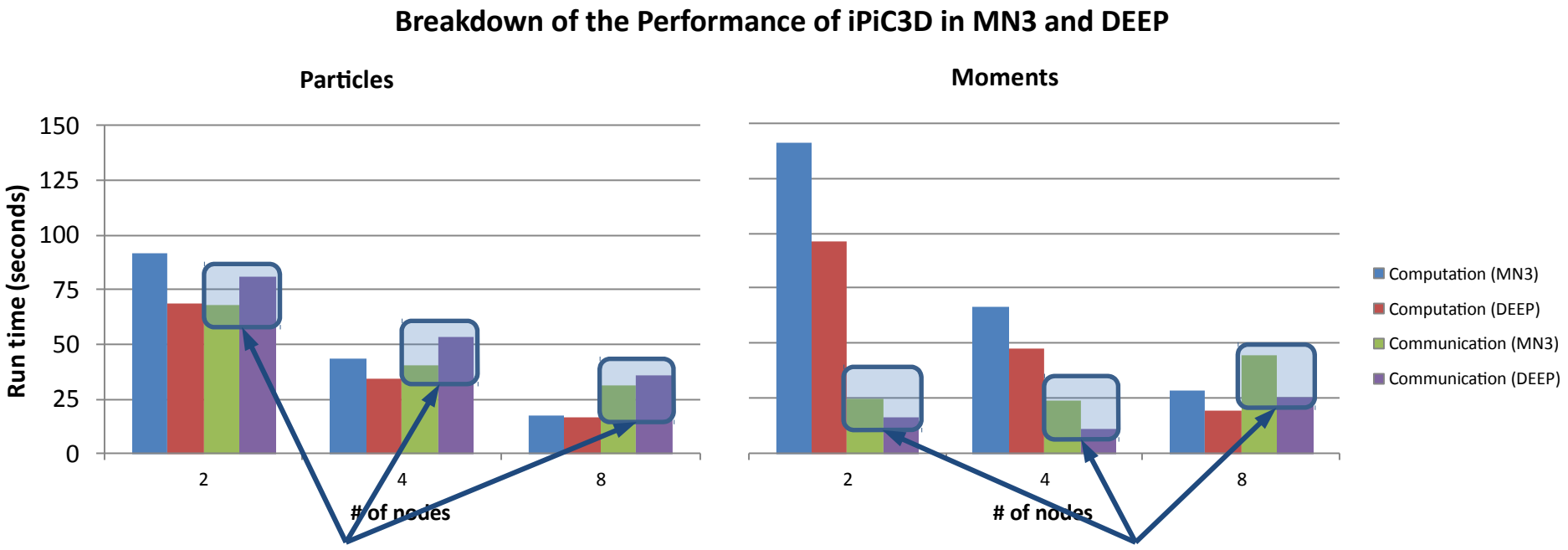
256x256 cells per node, 5x5x5x2 particles per cell, weak scaling
16 MPI processes per node, 16 MPI processes per Xeon Phi, 15 threads per MPI process on Xeon Phi
Xeon Phi to Xeon offload (m2x), and Xeon to Xeon offload (x2x)



DEEP Advantages

- Results on the DEEP system (Cluster + Booster)
  - Space Weather (Reverse offload of the fields solver)

256x256 cells, 18x18x1x2 particles per cell, strong scaling
16 MPI processes per node, 8 MPI processes per Xeon Phi, 30 threads per MPI process on Xeon Phi



Communication slower than on MN3

Communication faster than on MN3

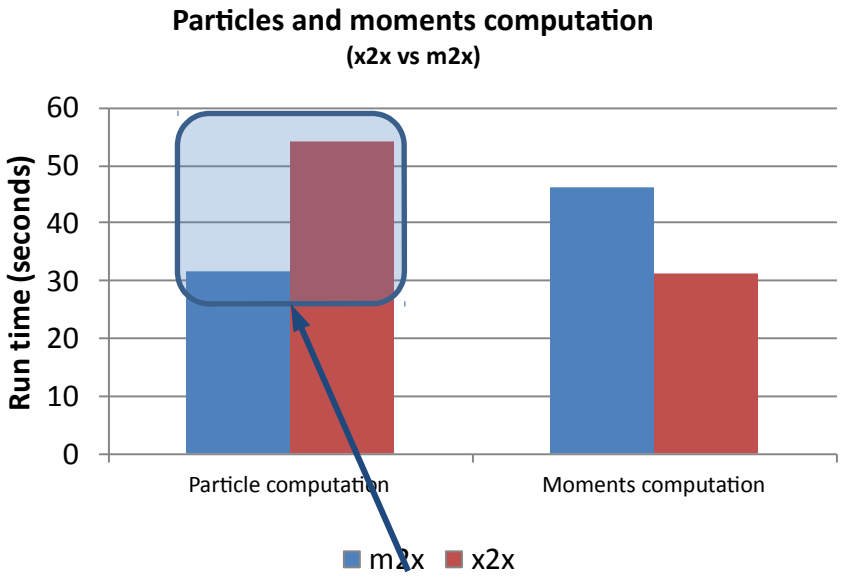
DEEP Advantages

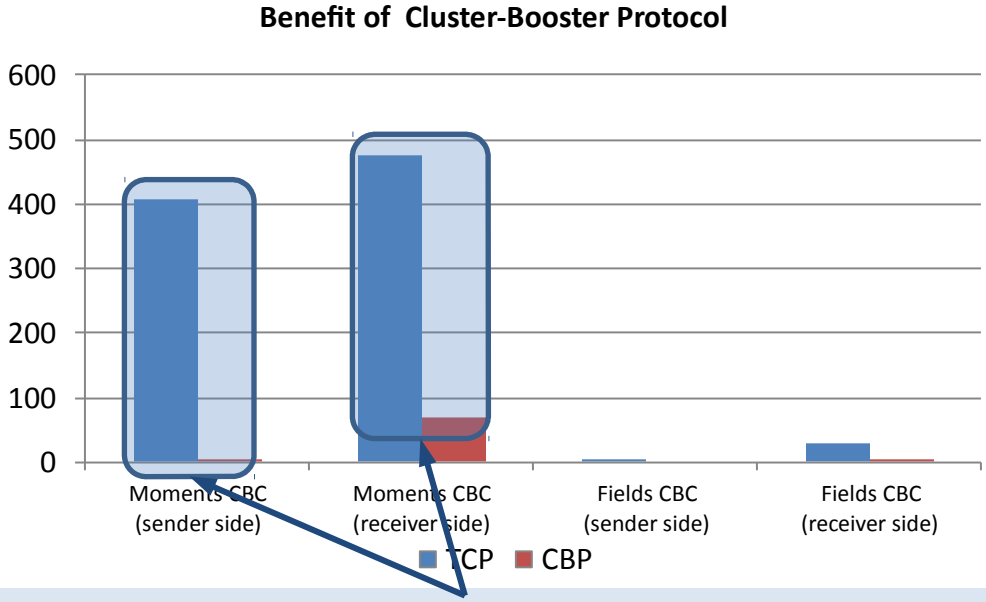
- Results on the DEEP system (Cluster + Booster)
  - Space Weather (Reverse offload of the fields solver)

256x256 cells, 18x18x1x2 particles per cell, strong scaling
16 MPI processes per node, 8 MPI processes per Xeon Phi, 30 threads per MPI process on Xeon Phi

x2x: 4 Cluster Nodes + 4 Cluster Nodes

m2x: 4 Booster Nodes + 4 Cluster Nodes





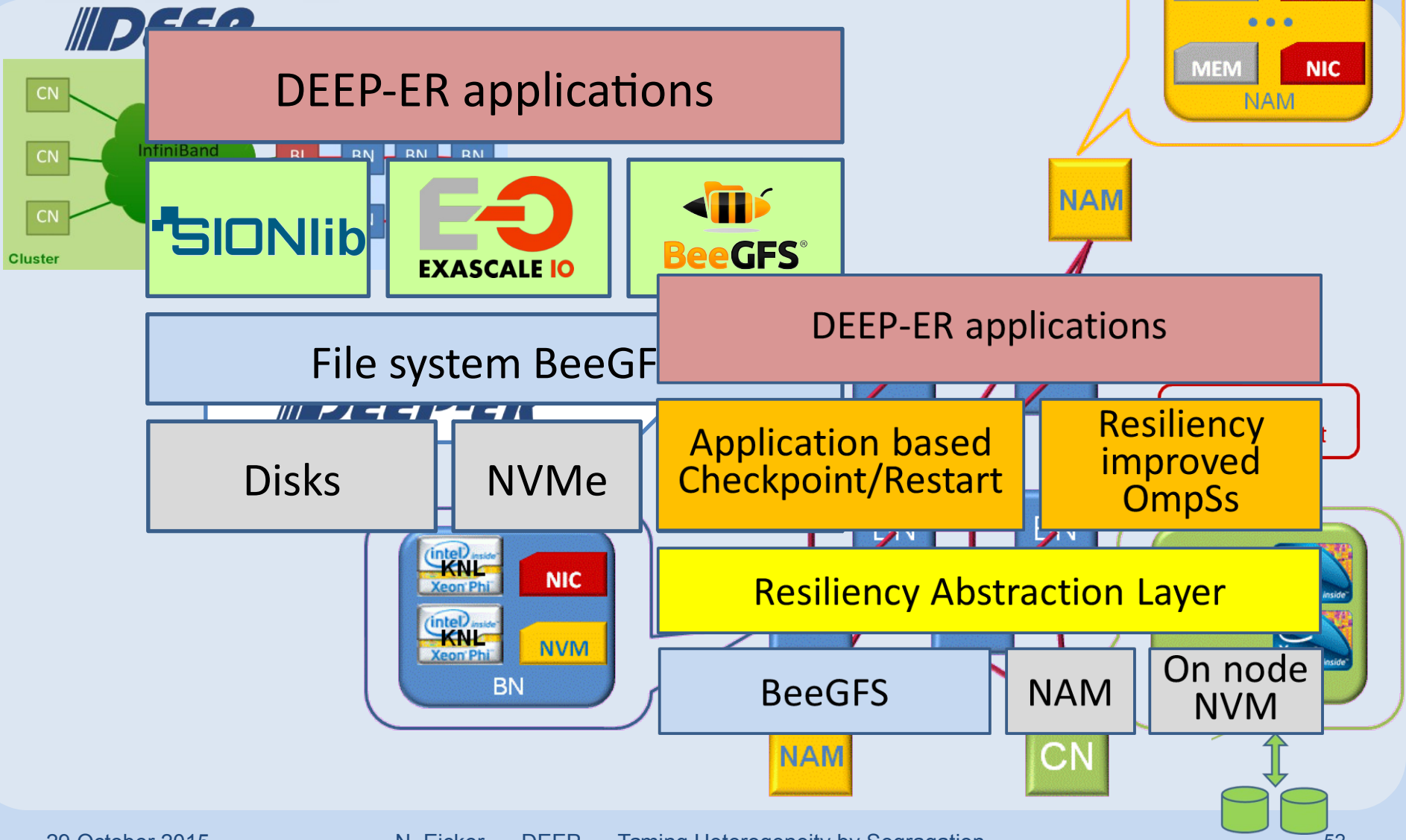
Particle computation is faster on Xeon Phi when enough particles are used

The impact of the CBP is huge for the moments

DEEP-ER Project

- 14 Partners
  - 4 PRACE hosts
  - 4 Research Centers
  - 4 Industry Partners
  - 4 Universities
  - Coordinator JSC
- 7 Countries
- Duration: 3.5 years
- Budget: 10.0 M€
- EU funding: 6.4 M€

Improving DEEP

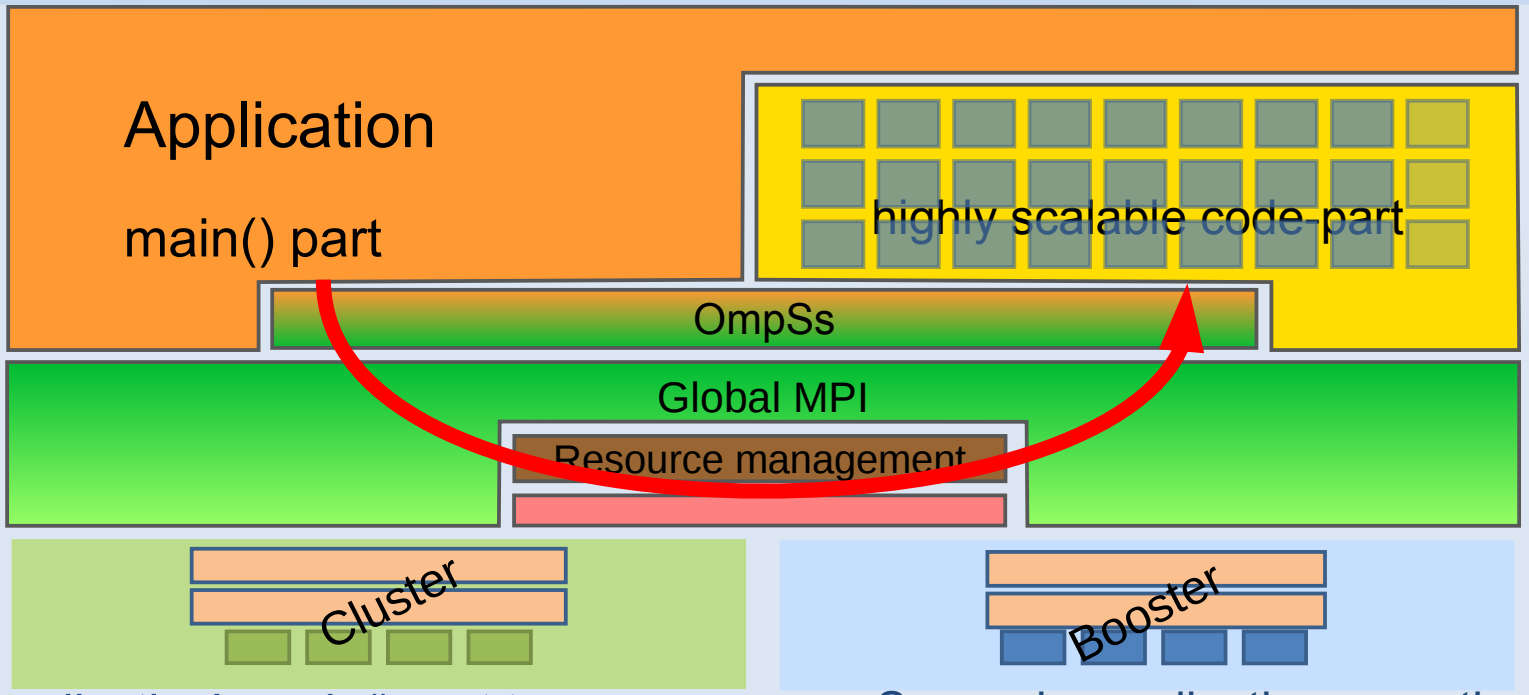


Resiliency

Resiliency

- DEEP already provides some nice features for resiliency
  - Based on the concept of state-less tasks in OmpSs
- But
  - No resources in DEEP to explore in more detail
  - Requires infrastructure to be developed in DEEP
  - More infrastructure not included in DEEP (I/O)
- DEEP-ER gives opportunity to explore in more detail
  - Implementation of the basic concepts
  - Rich potential of extensions
    - Checkpoint-Restart, Multi-level CP-RS, etc.
- More opportunities by future technologies
  - Non-volatile memory

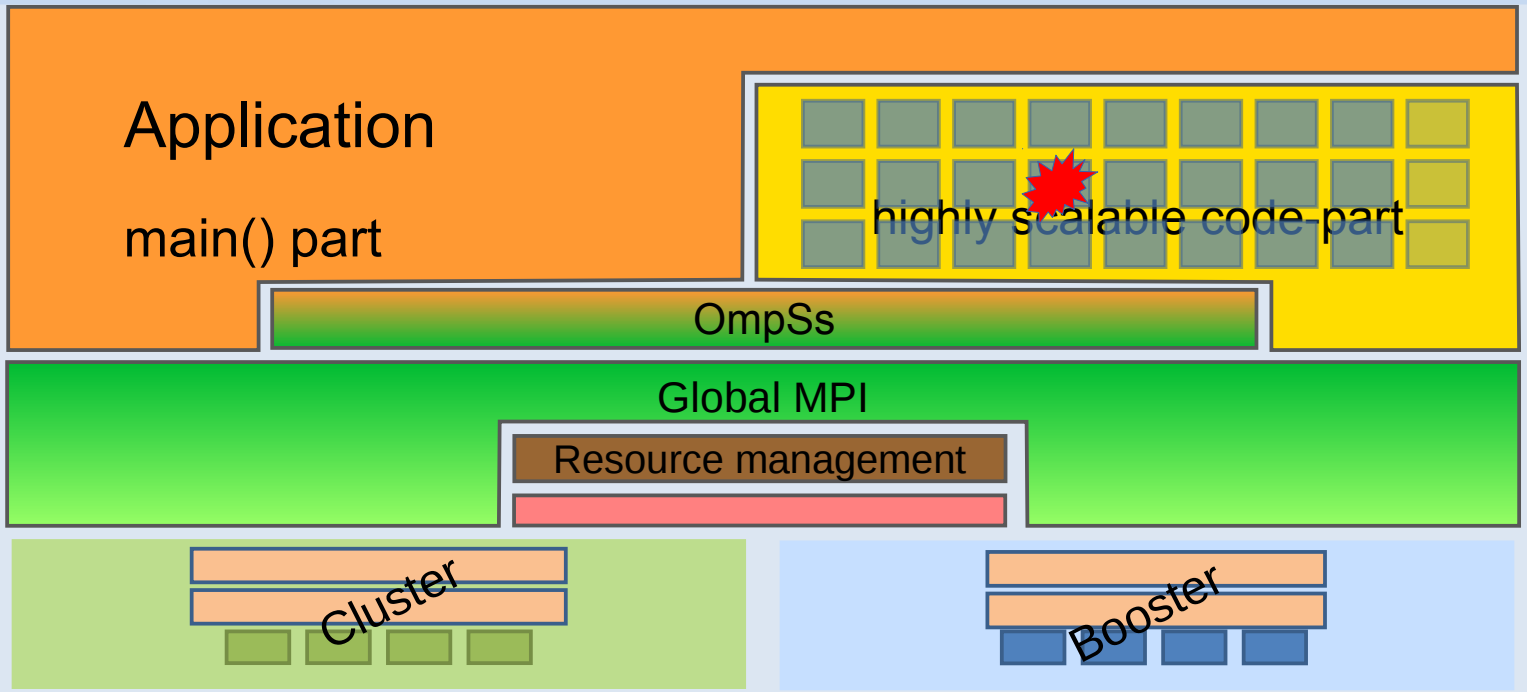
Startup in detail



- 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

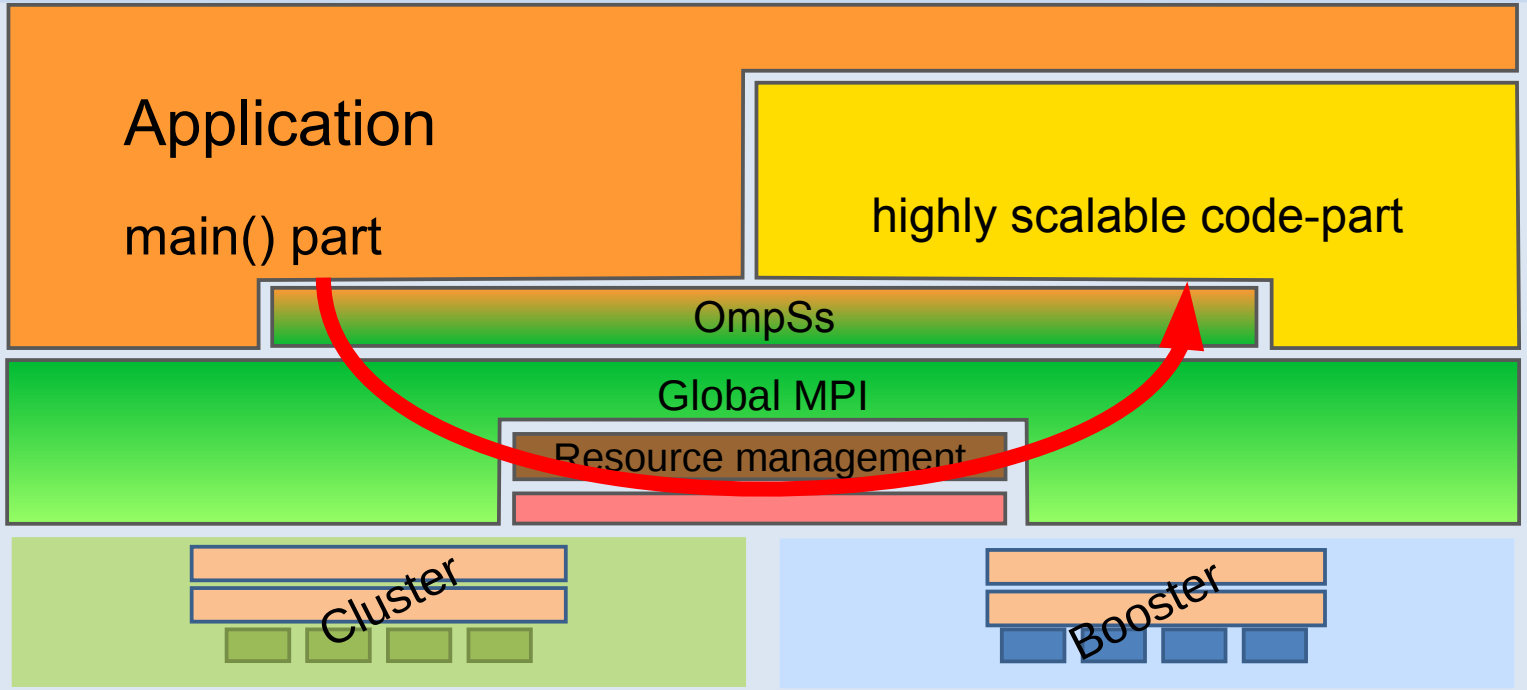
- Spawn is a collective operation of Cluster-processes
- Highly scalable code-parts (HSCP) utilize multiple Booster-nodes (BN)

Startup and failure



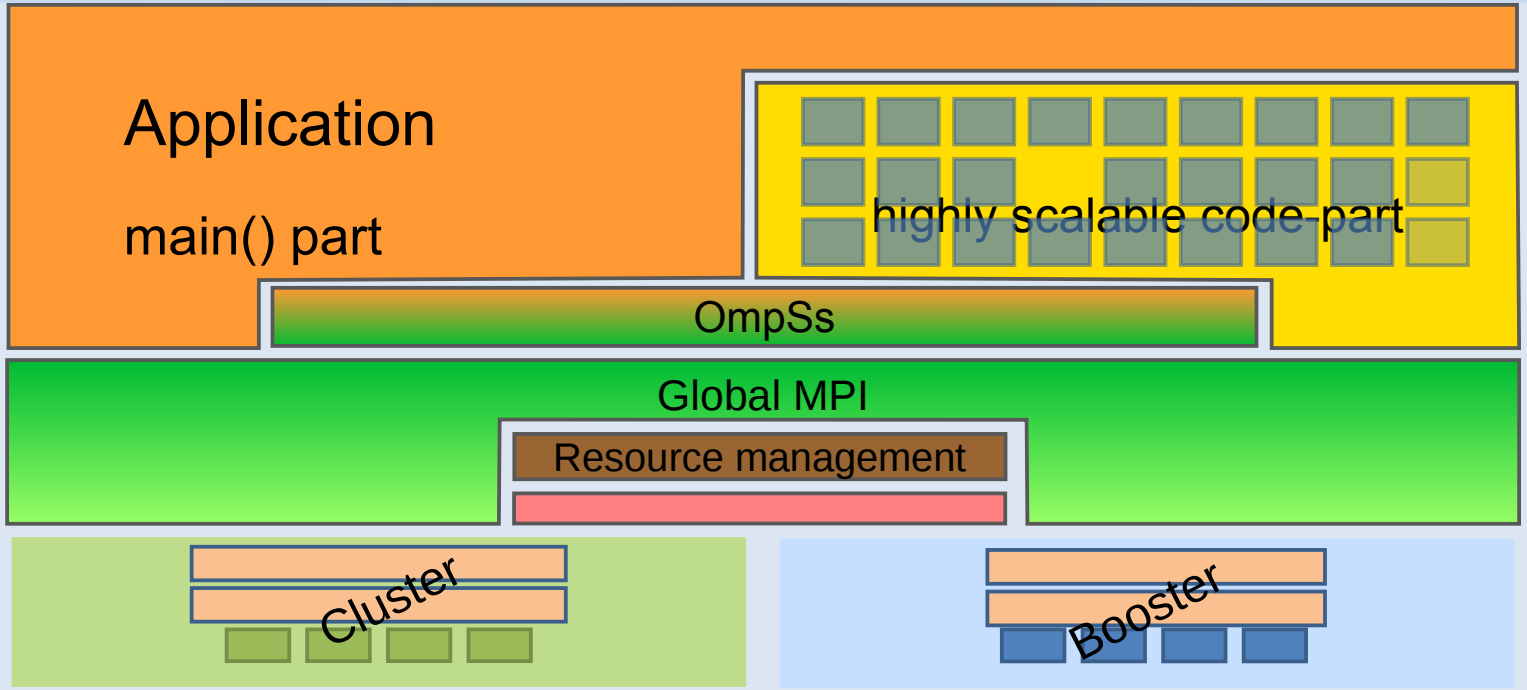
- HSCP might not make use of all resource on the Booster
  - Some spares might be held back to recover from failure
  - Depends on applications flexibility
- BN fails → HSCP not to survive
  - Booster-MPI to cleanup HSCP's resources
  - Global-MPI to ensure main()-parts survival

Restart



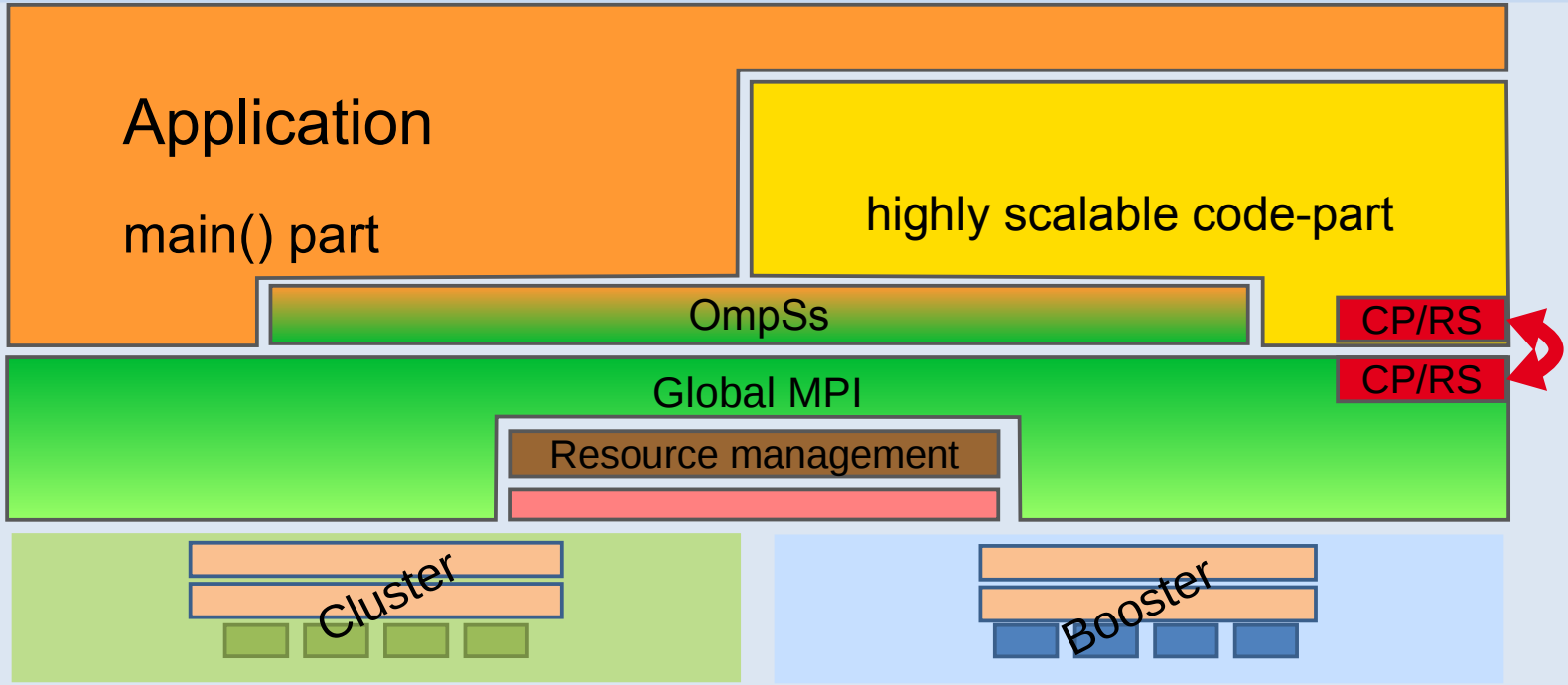
- main()-part is signaled by MPI on HSCP-failure
  - Re-start HSCP
  - Use spare-resources or rely on dynamic allocation
- Abstraction via OmpSs helps
  - Can keep track on how to start HSCP
  - Has control on HSCP's status

Restart – cntd.



- 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

Checkpointing in Software Architecture



- Tight integration with MPI required
- Application (HSCP) has to be checkpointing aware
  - Bring HSCP into defined state cooperatively
  - Actual write, global name-space, etc. to be provided by Checkpoint / Restart

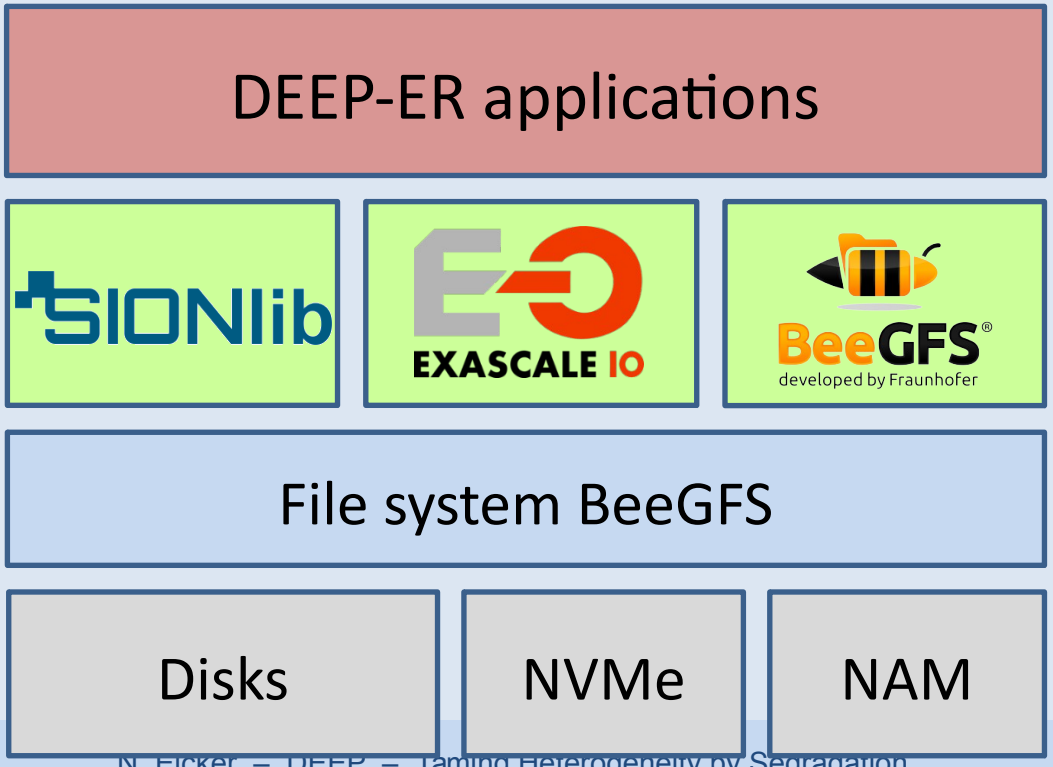
Multi-level checkpointing

- Combine local and global strategies
  - Apply strategy according to costs and probability of need
  - E.g.
    - Local checkpoint every 5 minutes
      → prevent from communication problems
    - Buddy checkpoint every 15 minutes
      → prevent from single node failures
    - Global file-system checkpoint every 60 minutes
      → prevent from buddy node failure

Adam Moody, Greg Bronevetsky, Kathryn Mohror, and Bronis R. de Supinski. 2010. Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System. In Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC '10). IEEE Computer Society, Washington, DC, USA, 1-11. DOI=10.1109/SC.2010.18 http://dx.doi.org/10.1109/SC.2010.18

Scalable Parallel I/O Layers

- Goal: Improve I/O scalability on all usage-levels
  - BeeGFS leverages architecture and novel memory technologies
  - Extended I/O APIs combine performance with ease of use



Take aways

- DEEP and DEEP-ER explore new ways to use and manage heterogeneity
  - Cluster Booster Architecture and it's implementation
  - Programming Model
- DEEP's 384-KNC Booster is up and running – mostly
- As second 32-KNC Booster utilizing immersion-cooling is on the way
- Both are application driven – co-design is important
- Try to hide the details via OmpSs' abstraction layer
- DEEP-ER extends the concept to I/O, Resiliency and innovative memory technologies like NVMe and NAM
- Jülich plans to extend its new Cluster JURECA by a 10 PF/s Booster in 2016 (based on KNL)
- More info: http://www.deep-project.eu
  http://www.deep-er.eu