Implementing a new computer architecture paradigm

Estela Suarez

Jülich Supercomputing Centre

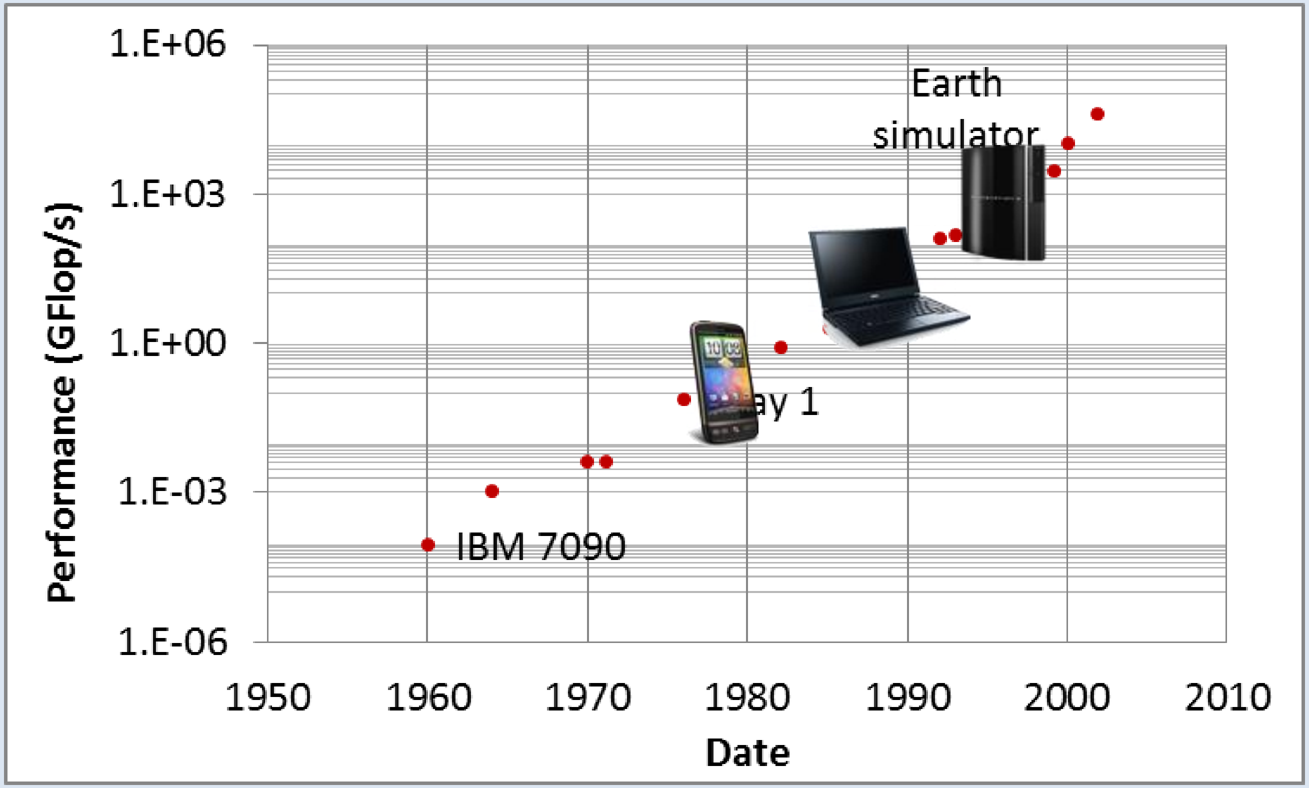
15.01.2016

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 and n° 610476

Outline

- Architecture evolution in Supercomputing
  - Cluster computing
- DEEP approach
  - The Cluster-Booster architecture
  - Hardware realization
  - Software environment
  - Applications
  - Project achievements
- Future: DEEP-ER and modular supercomputing
- Summary

Supercomputing evolution
Performance



Moore's law

- Every 18-24 months the # transistors in microprocessors doubles

Supercomputing evolution

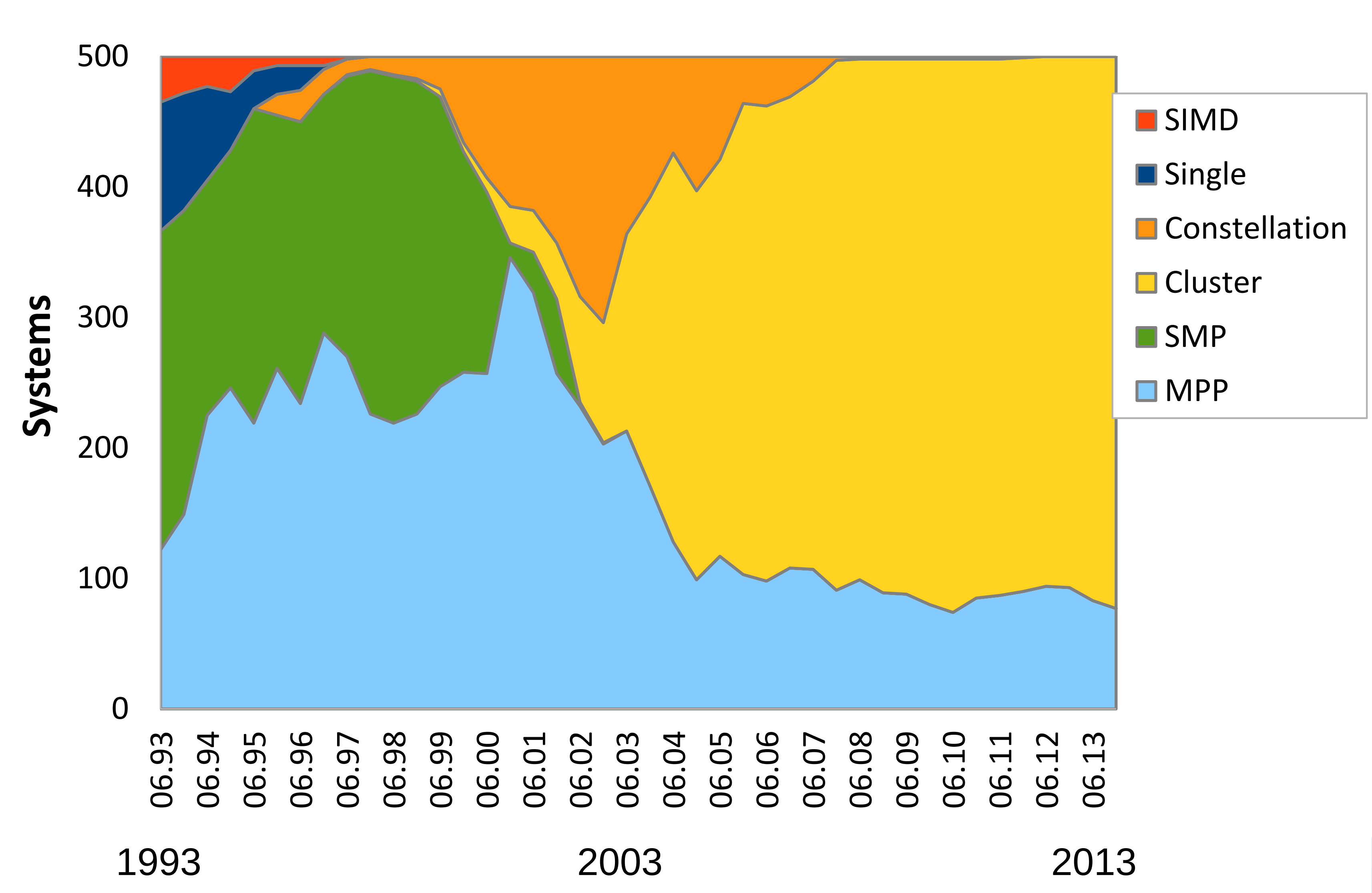
Architecture paradigms

- **1940 – 1950**: first computers are Supercomputers
  - Specialized, very expensive
- **1960 – 1980**: general purpose computers appear
  - Still special machines needed to solve very complex problems → Supercomputing (High Performance Computing - HPC)
    - Focus: floating operations (linear Algebra)
    - Special purpose technologies (fast vector processors, parallel architectures)
    - Only few machines produced

- **1990 – 2000**: integrate standard processors
  - Many „computers“ connected through fast network
    - Distributed memory → MPI
  - Both in proprietary Massively Parallel Systems (MPP) and Cluster Computing
- **2010 –** : heterogeneous cluster systems
  - Use accelerator technologies (GPU, Xeon Phi)

Supercomputing evolution

TOP 500 – Architectures

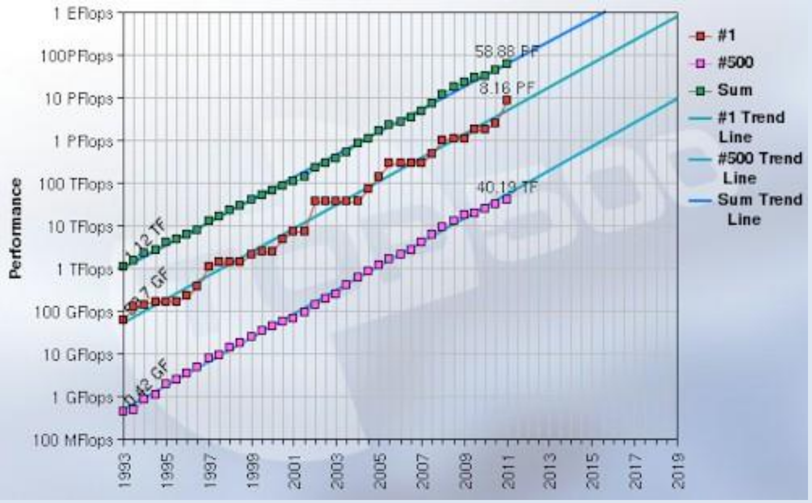


Supercomputing evolution

Challenges for Exascale

- Energy consumption
- Heterogeneity
- Programmability
- Huge levels of parallelism
- Scalability
- Resiliency
- Exploding data requirements
- Algorithms and application readiness

x1000 performance in 10 years
→ Exascale in 2020?



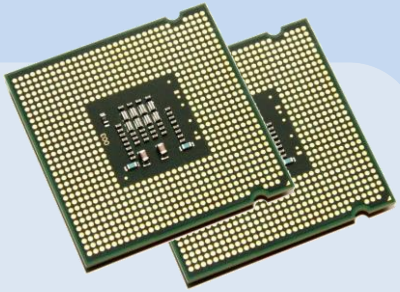
Top500

Can we solve these issues by improving today's Cluster computing? → DEEP

CLUSTER COMPUTING

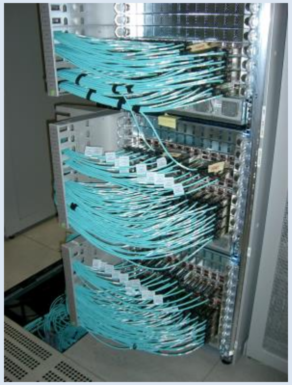
Cluster „anatomy“

Components selected/tuned to fit user needs



- **Heart**: Processor providing performance
- **Brain**: Memory / Storage
  - With many layers (Caches, DRAM, SSD, HD, Tape)

- **Nerve system**: Network
  - Often more than one (MPI, Administration, I/O...)



- **Consciusness**: Software
  - Including various middleware layers (node management, process management, MPI, etc.)
  - OpenSource
- **Balance** is more important than performance of individual components
  - Challenge: „slow“ memory and network evolution vs. computing performance

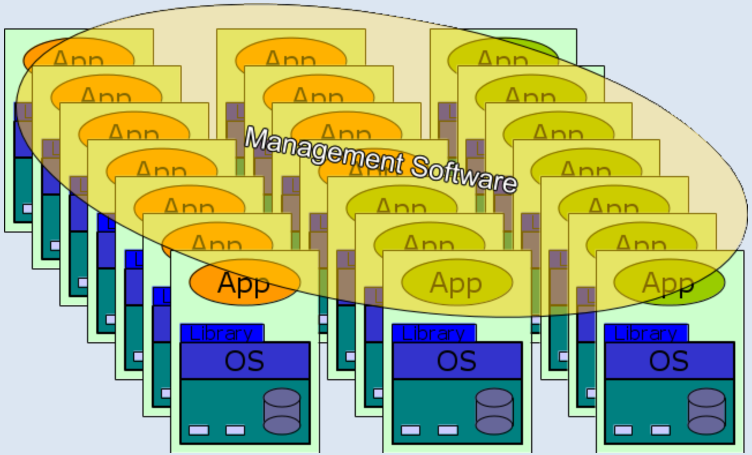
Cluster Management

Problem:

- All nodes are independent
- Each one has its own Operating System

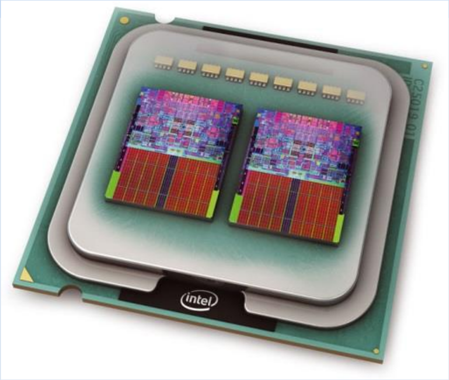
Needed functionality for a cluster consciousness:

- Node management
- Fault tolerance
- Process management
- Process-controlling (resiliency)
- Inter-process communication (MPI)



Realization of a Cluster-wide Meta-Operating System

General purpose processors



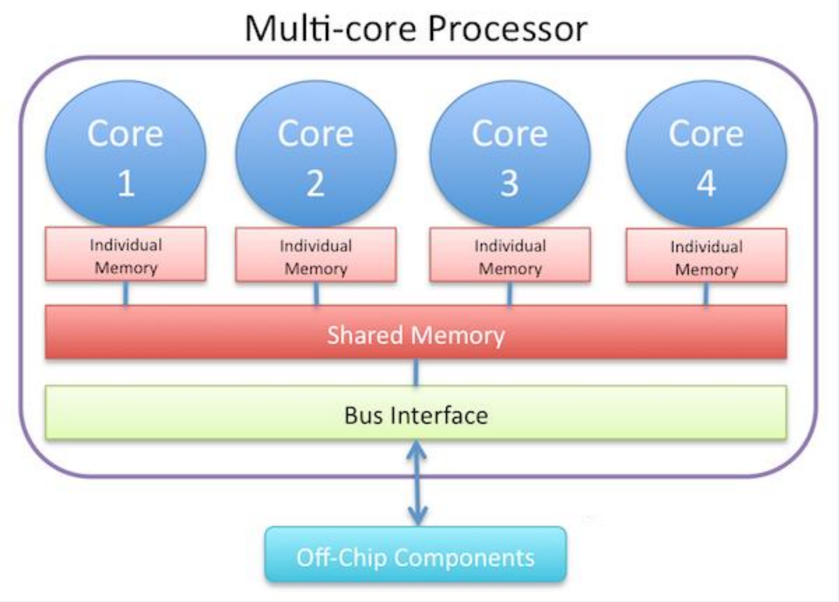
e.g.

Characteristics

- Broad range of capabilities
- Today always **multi-core**
  - Up to about 20 cores
- High single thread performance
  - High frequency
  - Out of order processing
- High memory per core
- Runs standard programming environment (MPI, OpenMP, etc.)

Disadvantages

- Limited energy efficiency
- Larger $/FLOP



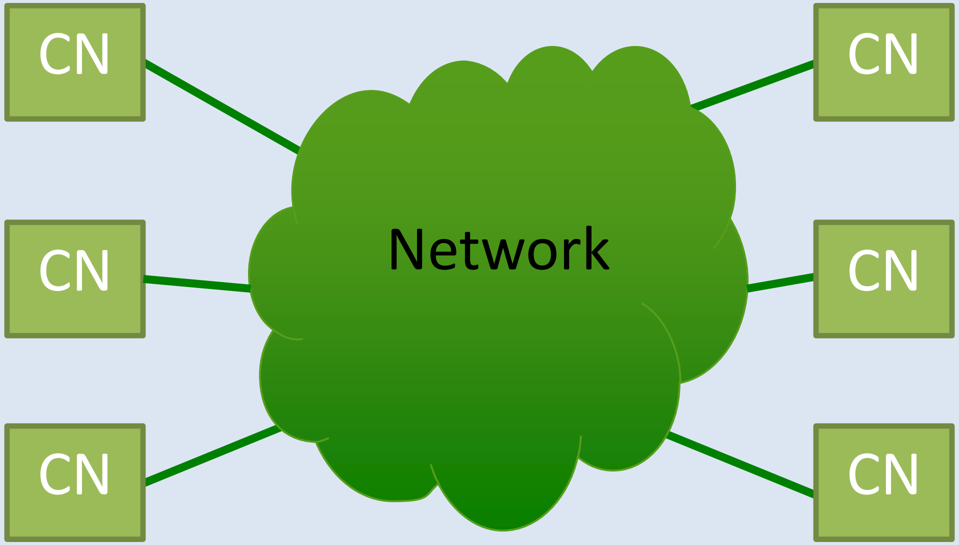
Accelerators

- **Many core** (Intel Xeon Phi)
  - 60 to 72 cores, 4 threads / core
  - Rather low single thread performance
    - In-order architecture, low frequency
  - Energy efficient ($/FLOP)
  - x86 architecture → standard programming with MPI, OpenMP, etc.
  - Can run autonomously (without host)

- **GPGPU** (Graphic cards)
  - Designed for graphics but evolved into general purpose
  - Hundreds of (weak) computing cores
  - Very energy efficient ($/FLOP)
  - Require specific programming models (CUDA, OpenCL)
  - Require a host CPU

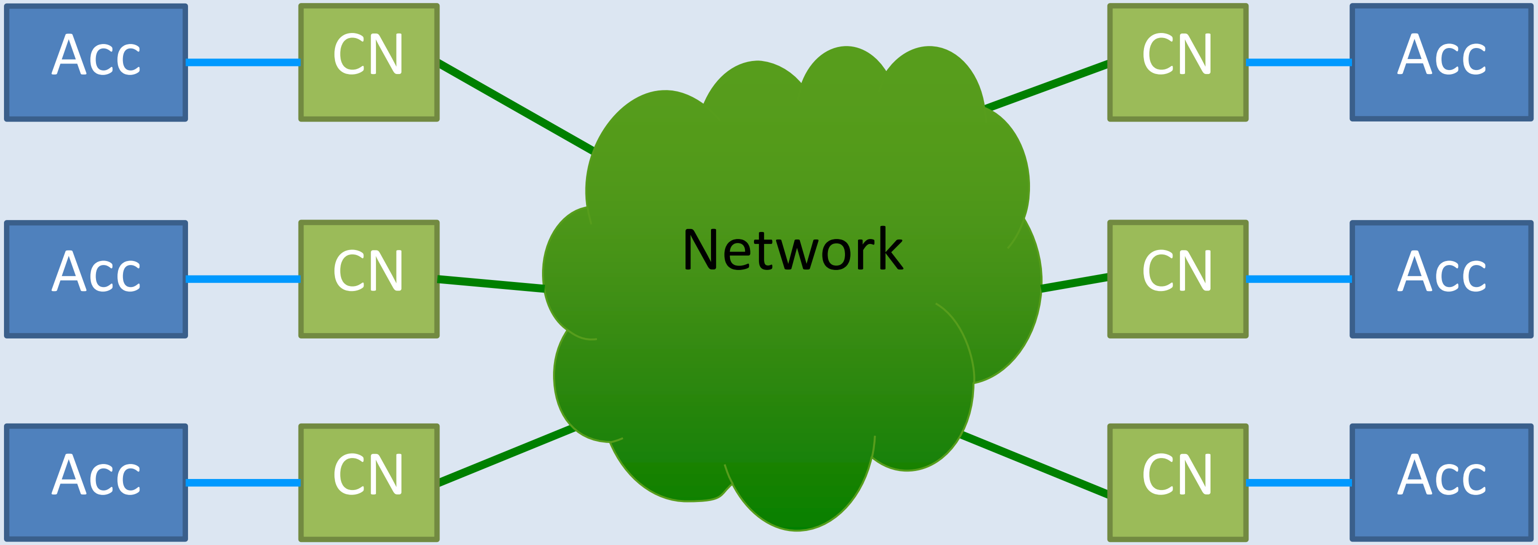
e.g.

Homogeneous cluster



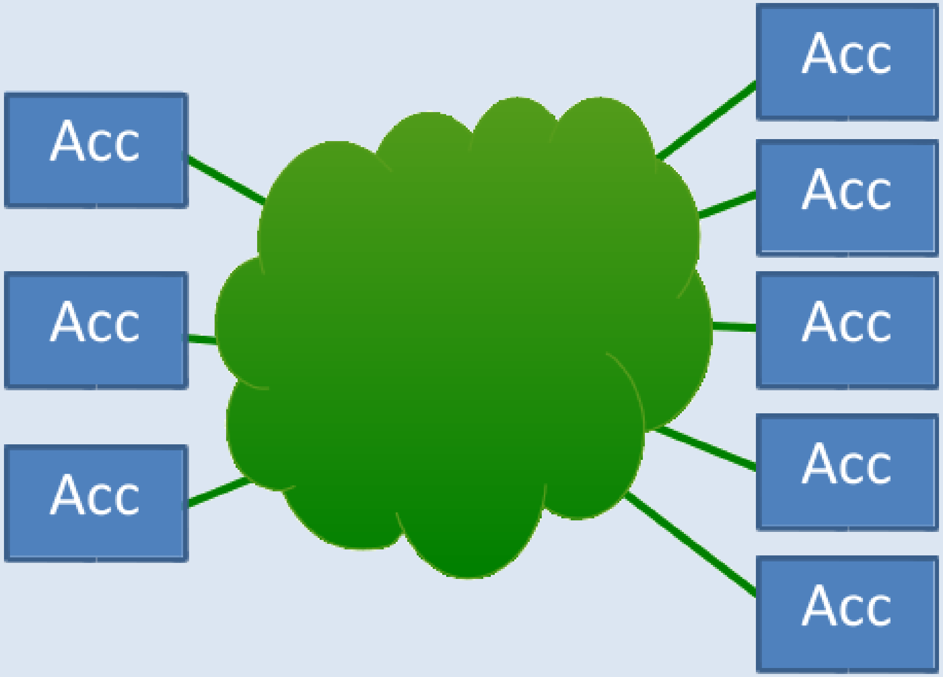
- Cluster Nodes: **general purpose** (multi-core) processor technology
  - Same processor characteristics in all nodes
- Single high-speed network connecting them all
- Good concept but limited efficiency for selected HPC applications

Cluster heterogeneity today
Accelerated cluster



- One or more accelerators attached to each Cluster Node
  - Static assignment of accelerators to CPUs
  - Accelerators cannot act autonomously
- Flat network topology
- Simple resource management

Cluster of Accelerators

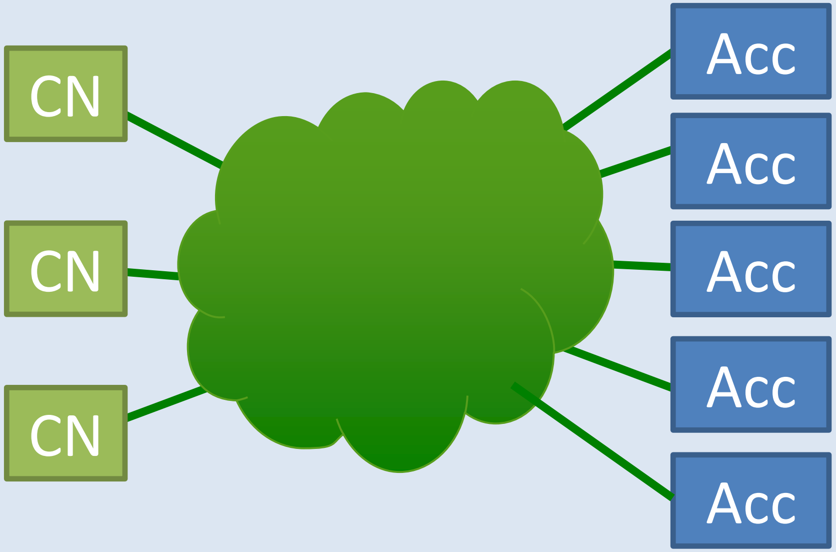


Ex. QPACE

- Node consists of an accelerator
- Directly connected to network
- Only possible with few accelerator technologies
  - Most need CPU to boot and communicate with the network

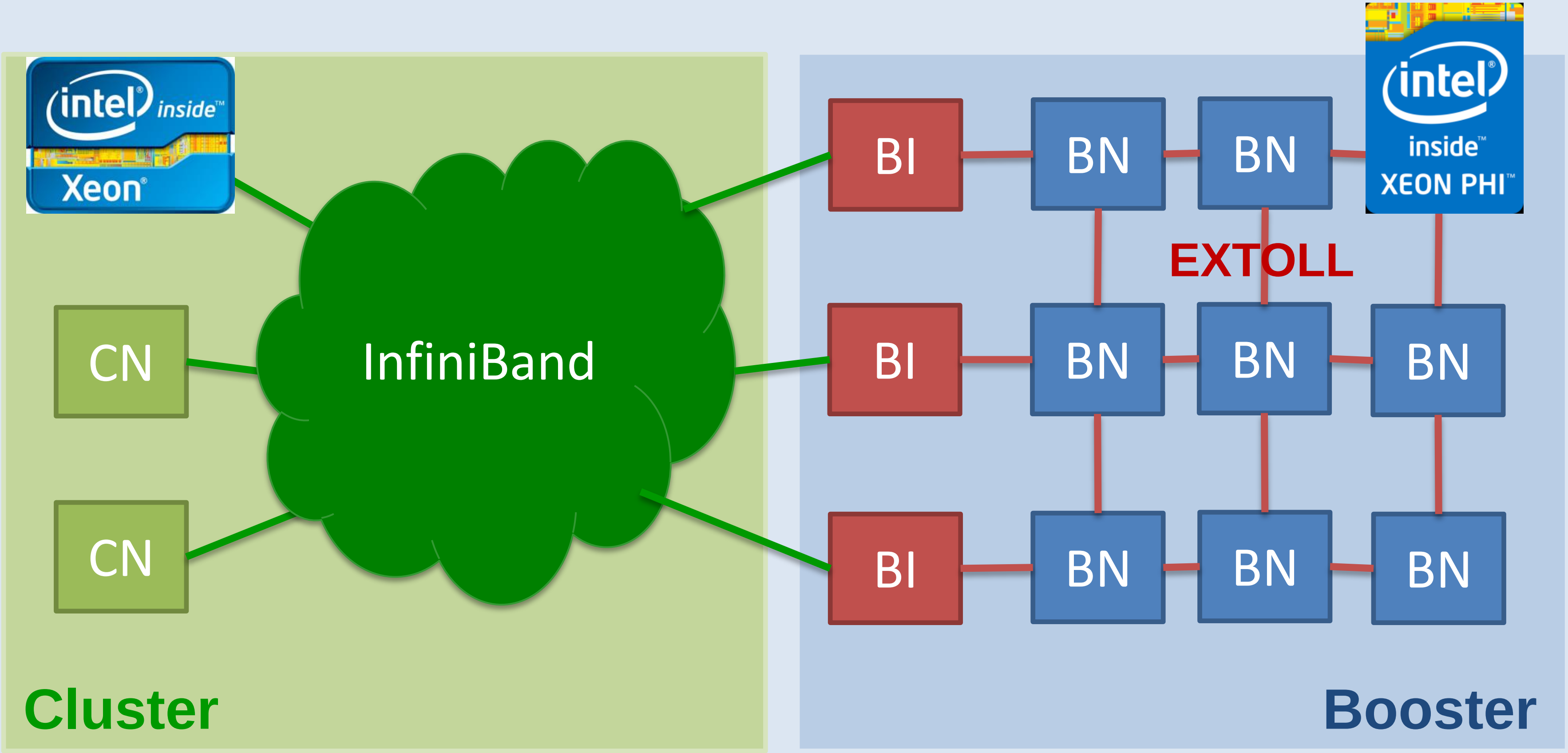
Alternative Integration

- Combine cluster-nodes with advanced, autonomous accelerators
  - All nodes act autonomously
  - All nodes attached to network
- Dynamical assignment of cluster-nodes and accelerators
- Can off-load more complex (including parallel) kernels
  - communication between CPU and Accelerator less frequently
  - larger messages i.e. less sensitive to latency



DEEP

Cluster-Booster architecture



Low/Medium scalable code parts

Highly scalable code parts

Prototype HW development
Cluster

Off the Shelf

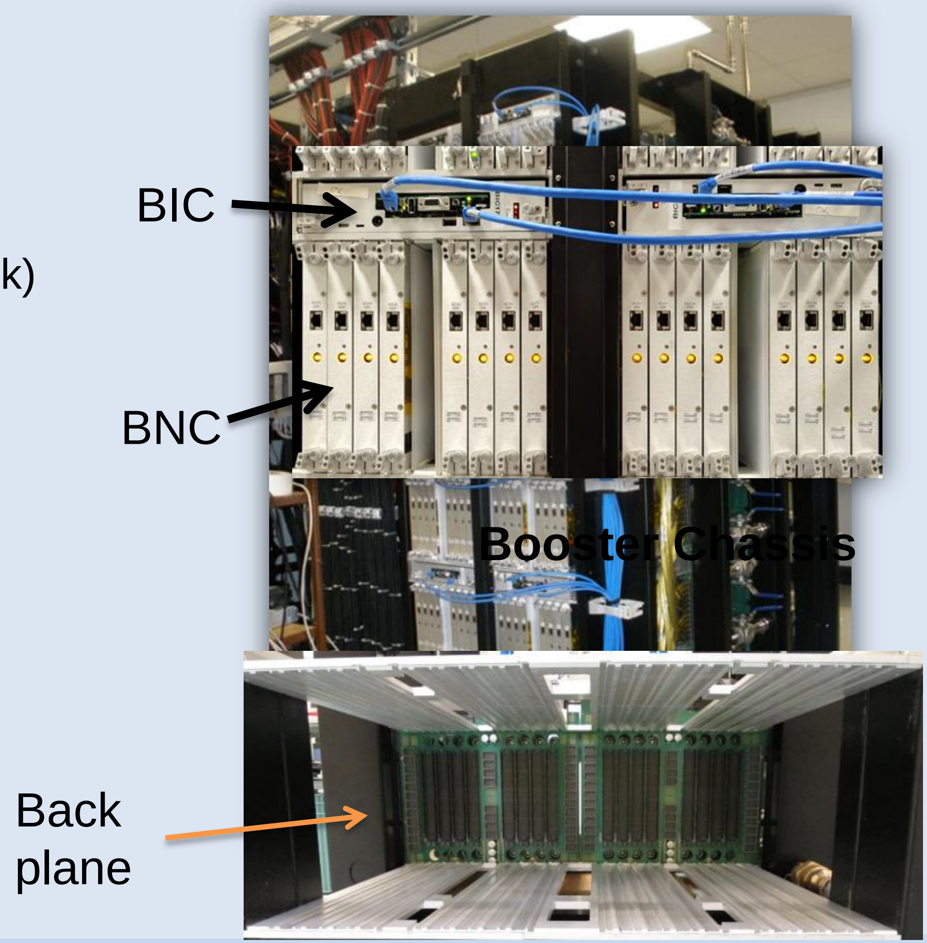
- Hot-water cooling cluster
- Compute nodes: 128
  - 2 x Intel Xeon (Sandy Bridge) CPU
  - 32 GB RAM
- Networks:
  - InfiniBand (QDR) – main network
  - Fat-tree topology
- Performance:
  - 44 TFlops peak
  - 2.8 GB/s IB bandwidth

DEEP Cluster

Prototype HW development
Booster

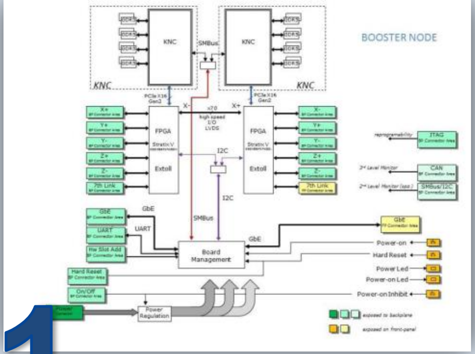
Designed and developed from scratch in DEEP

- Booster Nodes: 384
  - Intel Xeon Phi (Knights Corner)
- Network:
  - EXTOLL (direct non-switched network)
  - 3D torus (8×6×8)
- Performance:
  - 400 TFlops peak
  - 1.3 GB/s EXTOLL bandwidth
- Two main components
  - **BNC**: Booster Node Card
  - **BIC**: Booster Interface Card

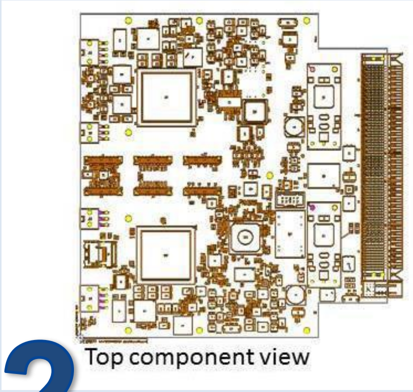


Prototype HW development

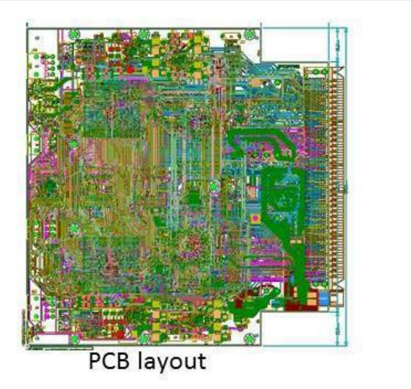
Process



1 Block design

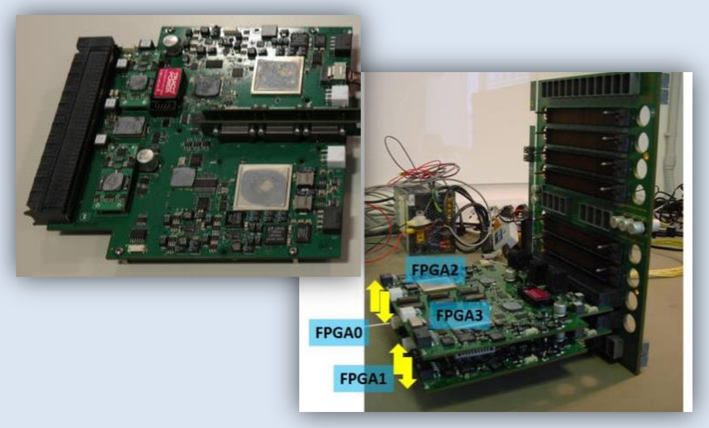


Top component view



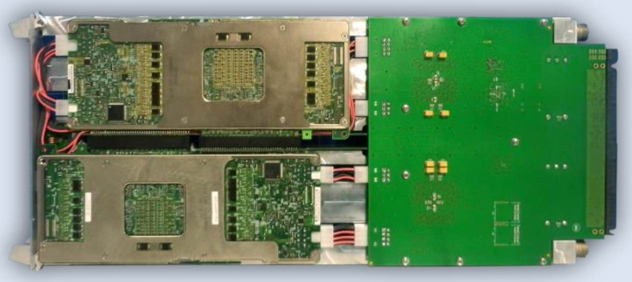
PCB layout

2 PCB design



3 First samples + test

4 Re-design + tests



5 Final board

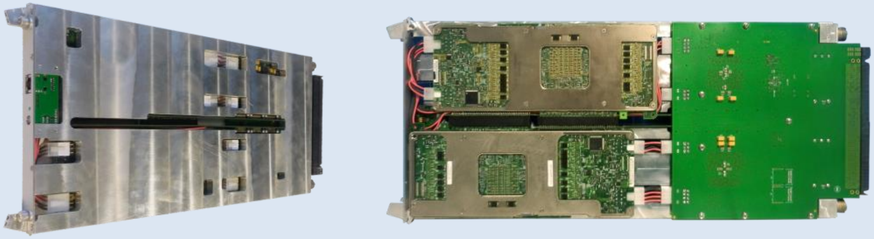
Prototype HW development

Booster Node Card

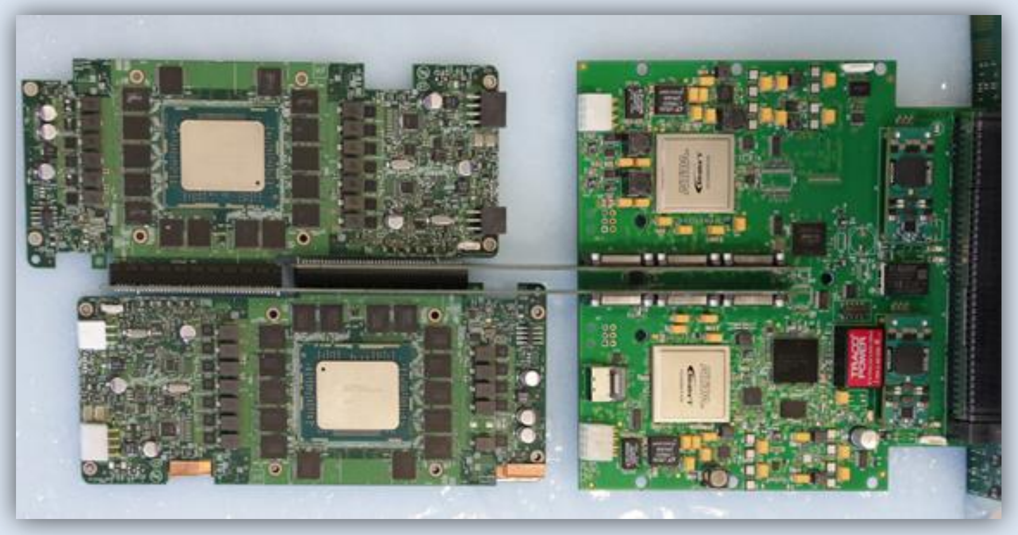
Computing heart of Booster

- Integrates two Booster Nodes
- 2x Intel Xeon Phi
  - Knights Corner
  - 61 cores each
  - 16 GB memory each
- 2x Network Interfaces (NIC)
  - Extoll network on FPGA
- Further components:
  - Board Management controller
  - Monitoring tools
  - Etc.
- Cooled via cold plate
- 16 BNCs integrated in chassis
  - 1 BIC per 8 BNCs

Assembled views



“Naked” view



Intel Xeon Phi

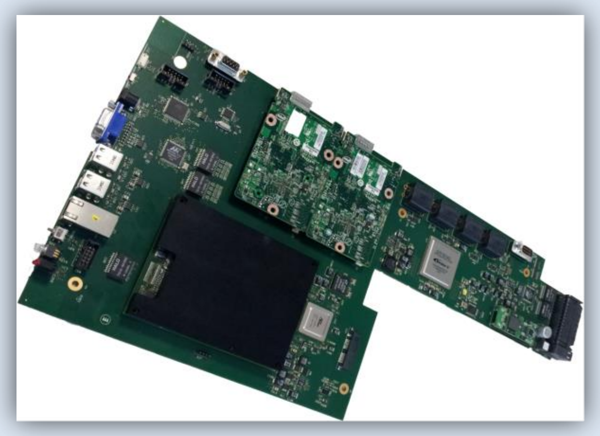
Networking
part of BNC

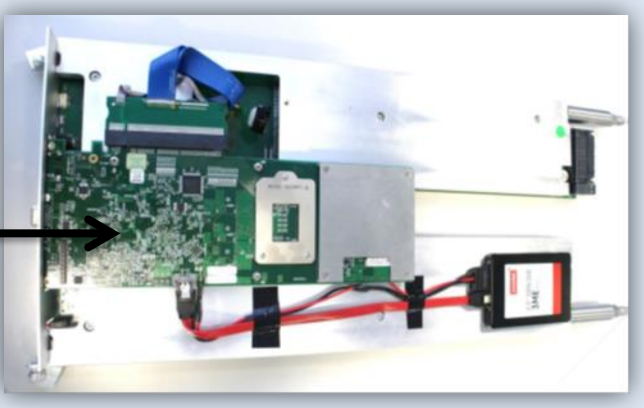
Prototype HW development
Booster Interface Card

Functions

- Boot and manage 16 Intel Xeon Phi
  - Enables Booster Nodes to work autonomously
  - Uses a server (Juno) to address memory addresses of the BNCs over the EXTOLL network
- Interface between Cluster and Booster
  - Has both InfiniBand and EXTOLL
  - Bridges between networks
- Additionally
  - Monitoring functionality for BNCs
  - Lights out management

BIC

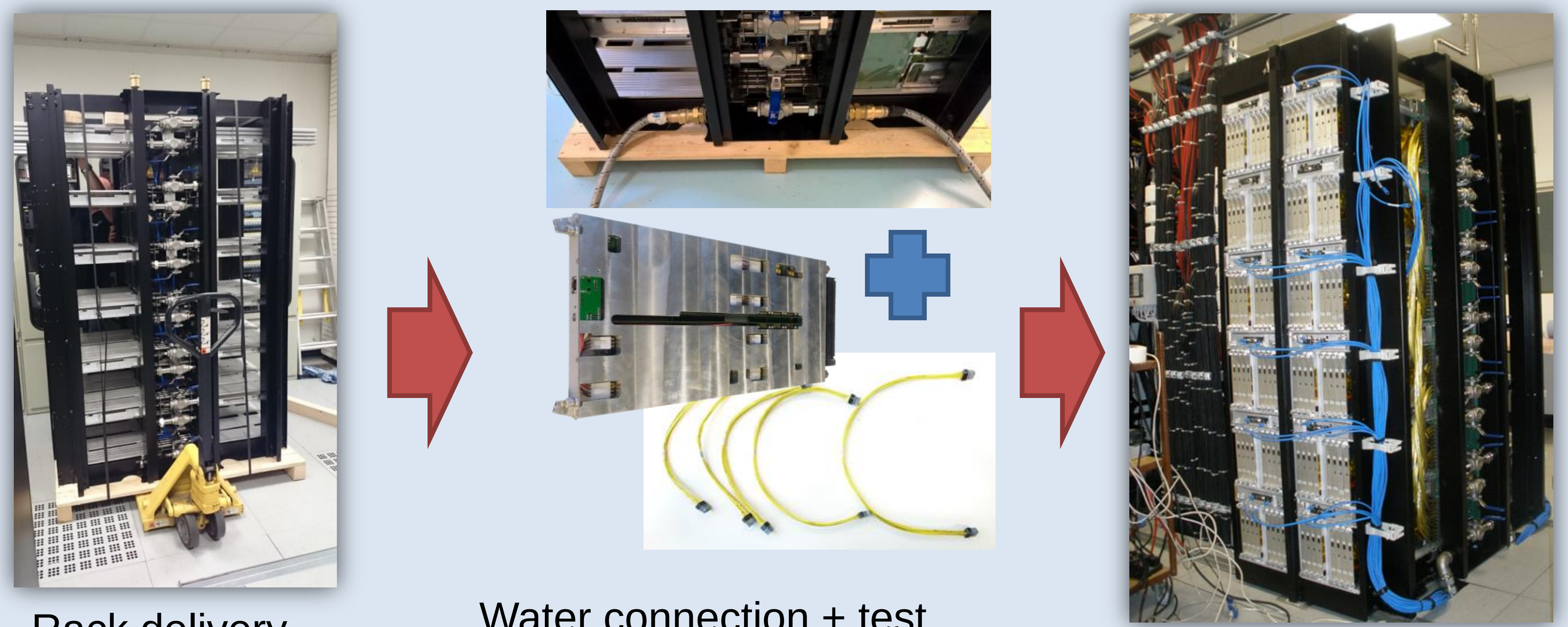




Juno server

Putting pieces together
Booster Integration/Installation

- Local infrastructure prepared beforehand

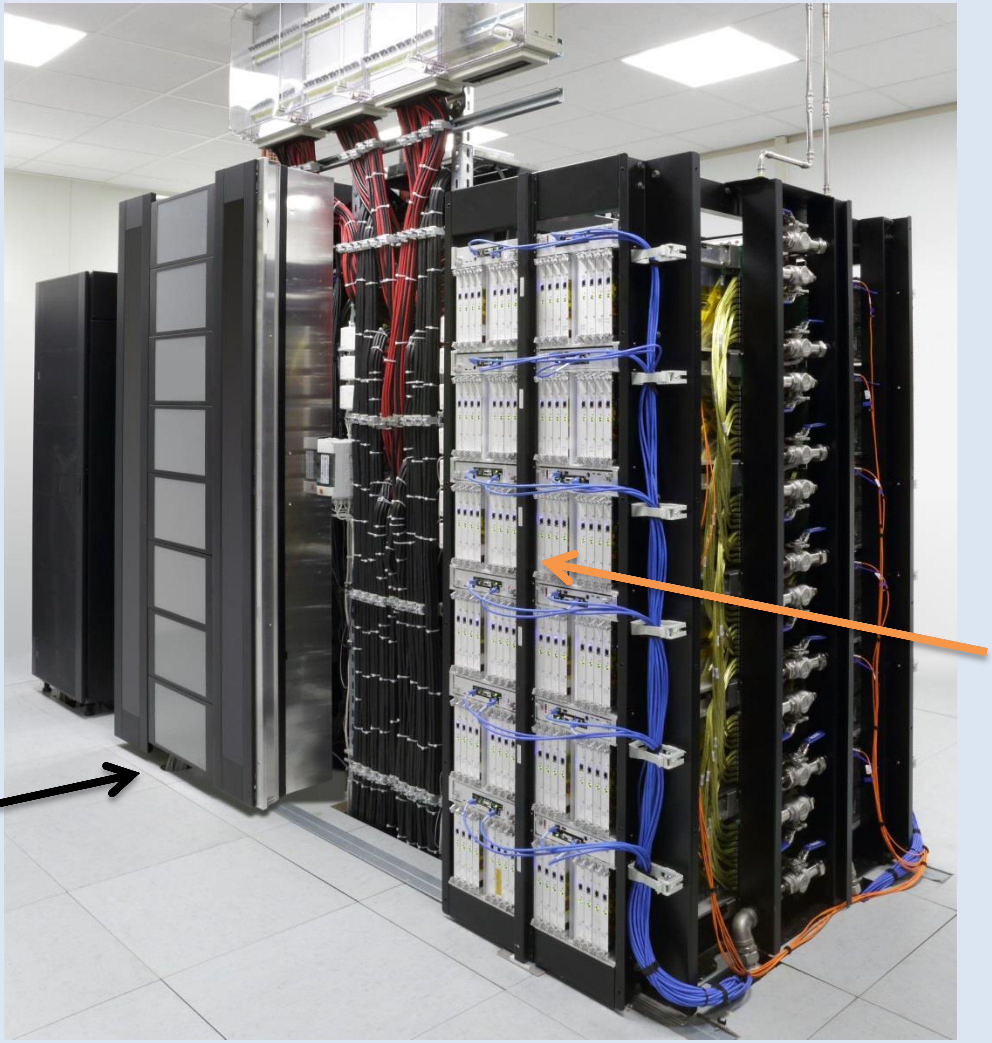


Rack delivery

Water connection + test
Board insertion + cabling

- Once the system is installed a long bringup phase starts

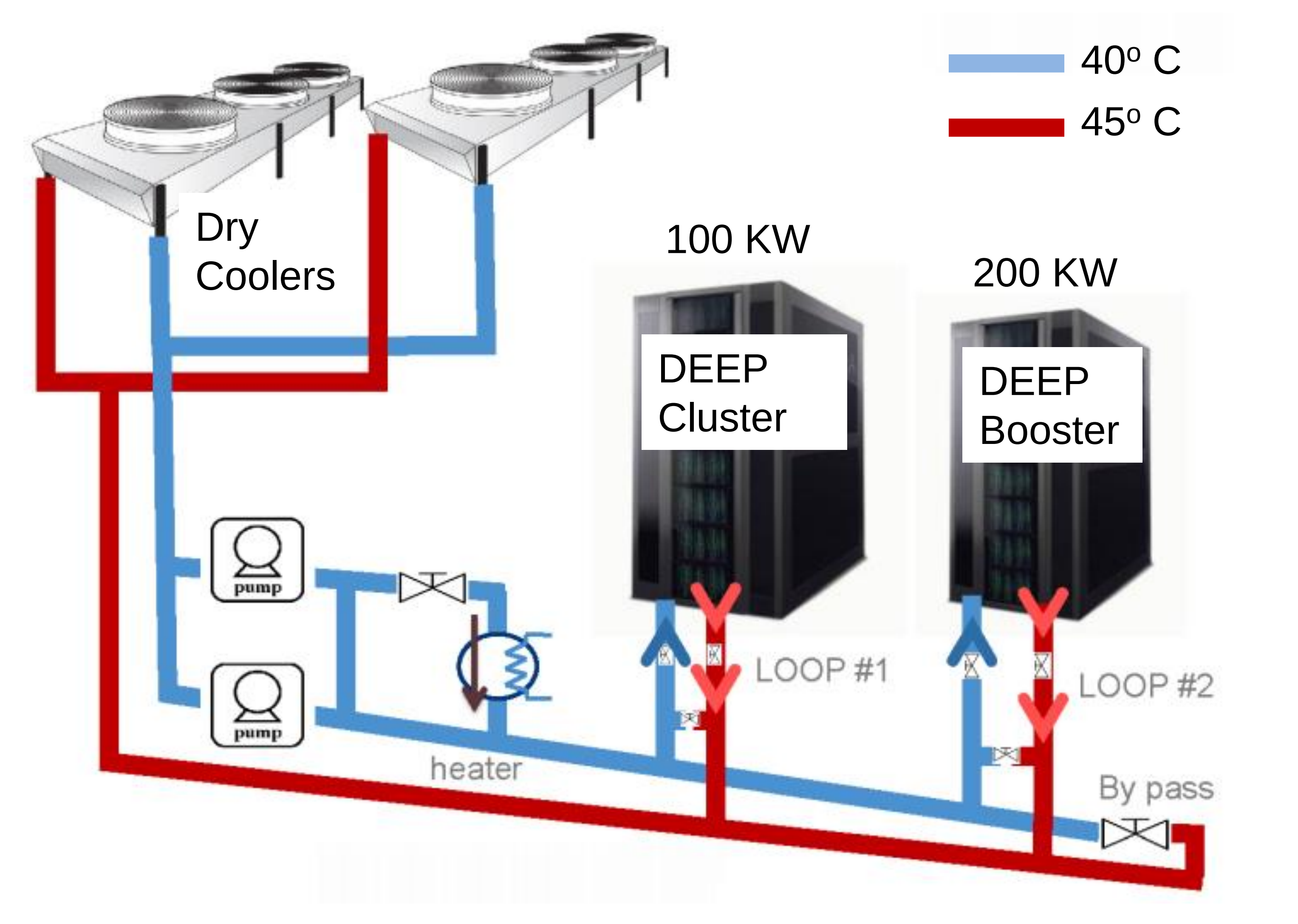
Complete DEEP System



DEEP Booster

DEEP Cluster

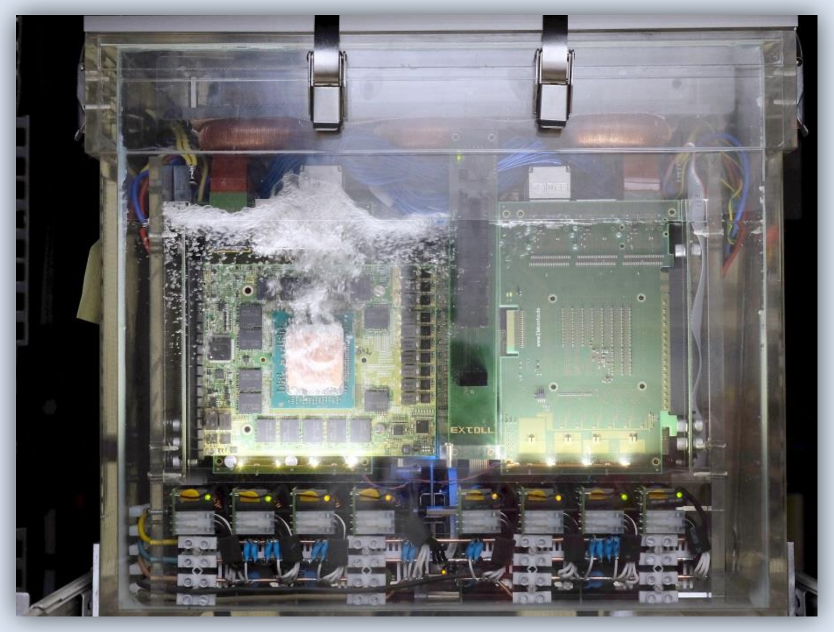
Hot-water cooling



GreenICE Booster

Alternative Booster implementation

- Interconnect EXTOLL
  - ASIC “Tourmalet”
- 32 Xeon Phi nodes
- Network: 4×4×2 topology, with Z dimension open



GreenICE Booster

Experiment with immersion Cooling

- 2-phase NOVEC liquid from 3M
- Evaporates at about 50 degrees
- Condensates again in a water cooling pipe

DEEP Software

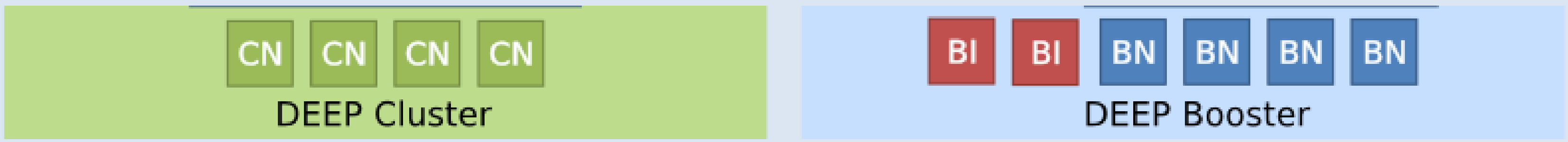
Reminder: Cluster-Booster architecture

- **Cluster**:
  - Fat-tree network (InfiniBand)
  - High single thread performance
  - Large memory per core
- **Booster:**
  - 3D torus network (EXTOLL)
  - Low single thread performance
  - Few memory per core

The DEEP Software stack must:

- Allow an easy programing of this new architecture
  - Hide underlying hardware complexity
  - Provide familiar programming environment
  - Include tools to analyse and optimise application performance

Software Architecture



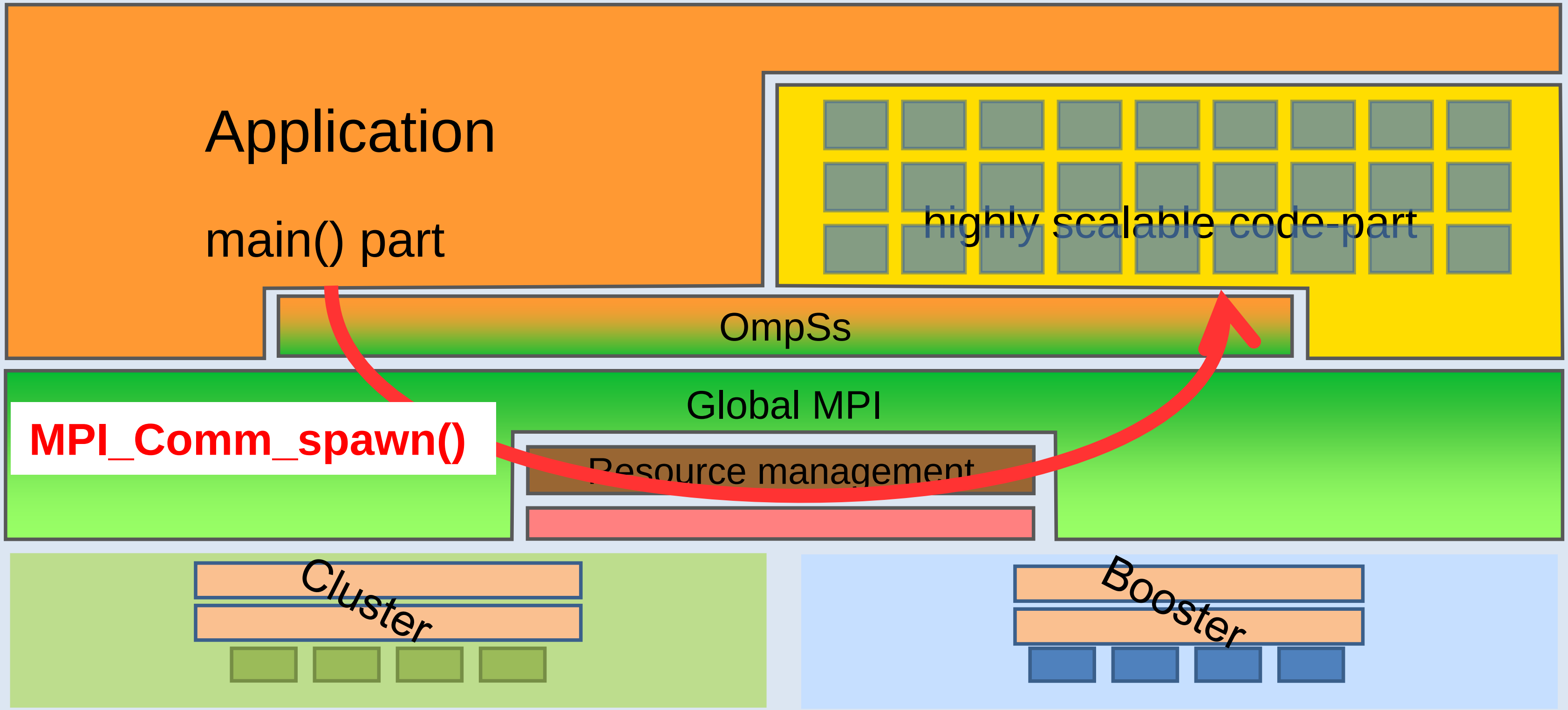
CN CN CN CN

DEEP Cluster

BI BI BN BN BN BN

DEEP Booster

Application Startup



- Application's main()-part runs on Cluster-nodes (CN)
- 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)

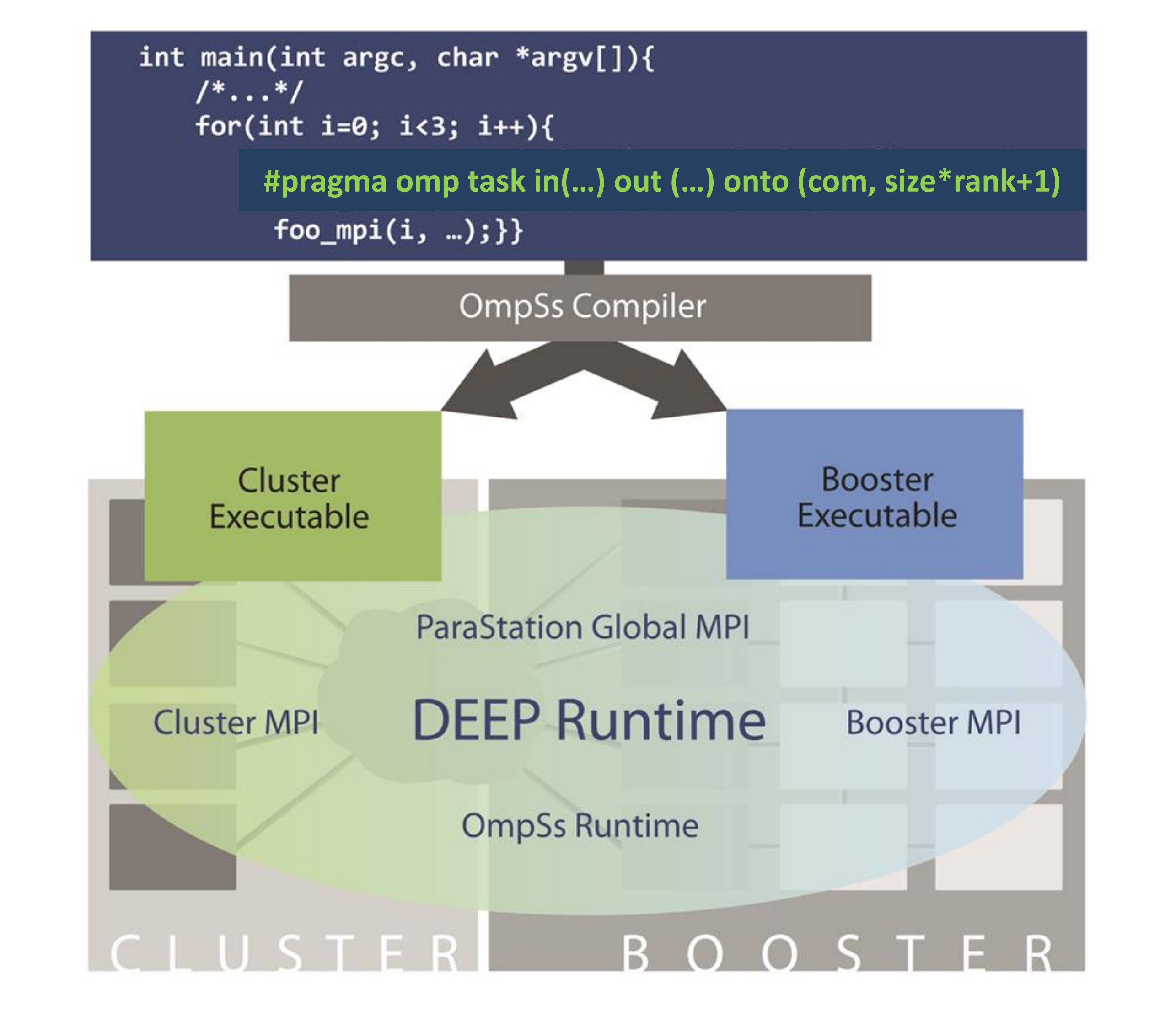
Application running on DEEP

Source code

Compiler

Application binaries

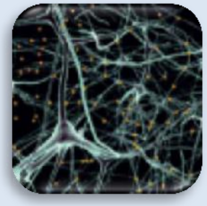
DEEP Runtime

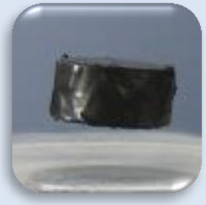


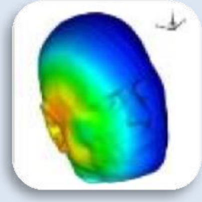
```
int main(int argc, char *argv[]){
    /*...*/
    for(int i=0; i<3; i++){
        #pragma omp task in(…) out (…) onto (com, size*rank+1)
        foo_mpi(i, …);}}
```

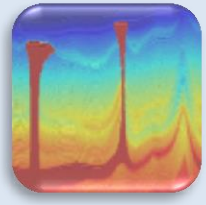
Application-driven approach

- **DEEP+DEEP-ER applications:**
  - Brain simulation (EPFL)
  - Space weather simulation (KULeuven)
  - Climate simulation (CYI)
  - Computational fluid engineering (CERFACS)
  - High temperature superconductivity (CINECA)
  - Seismic imaging (CGGVS)
  - Human exposure to electromagnetic fields (INRIA)
  - Geoscience (BADW-LRZ)
  - Radio astronomy (Astron)
  - Oil exploration (BSC)
  - Lattice QCD (UREG)

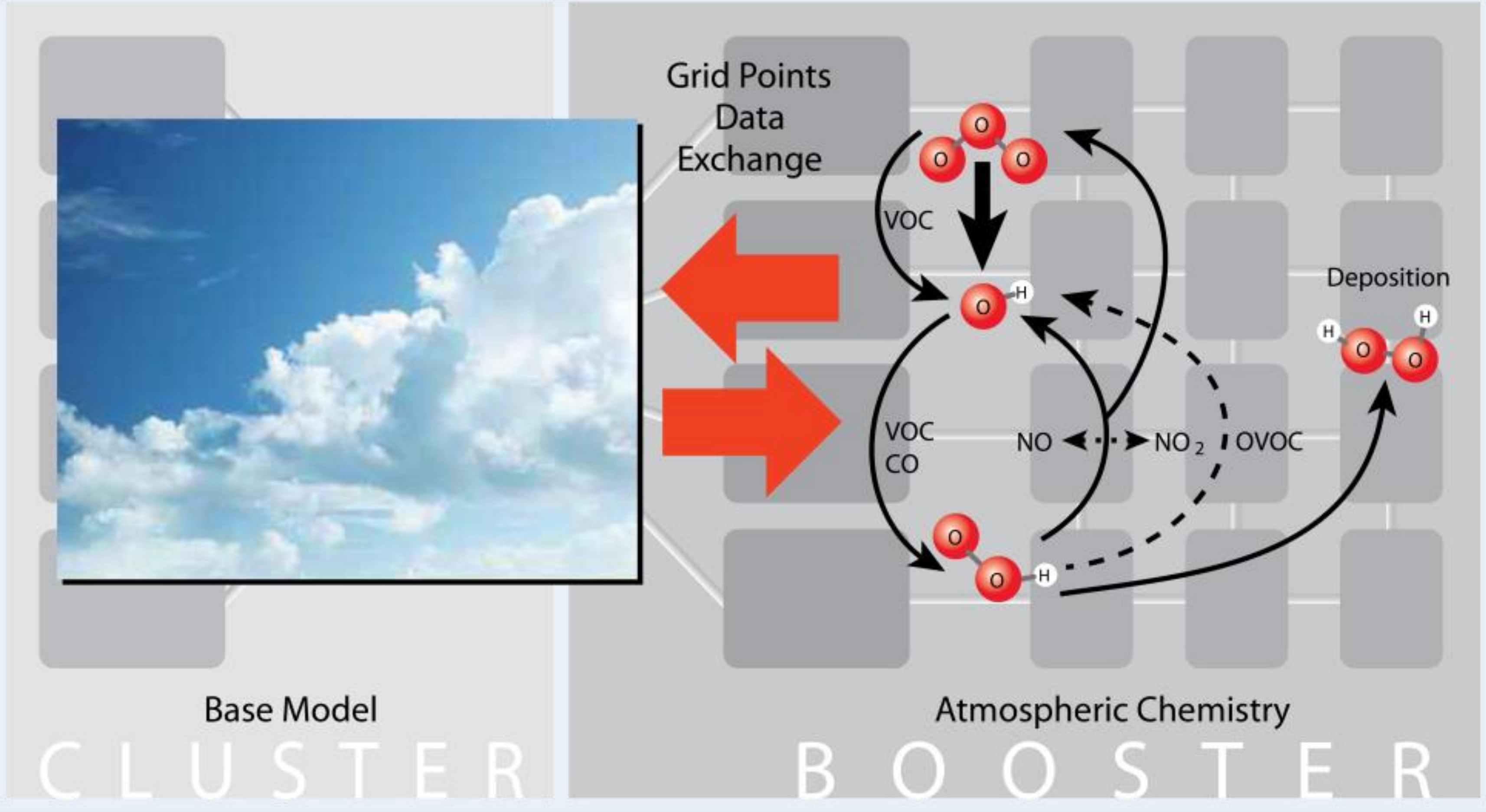






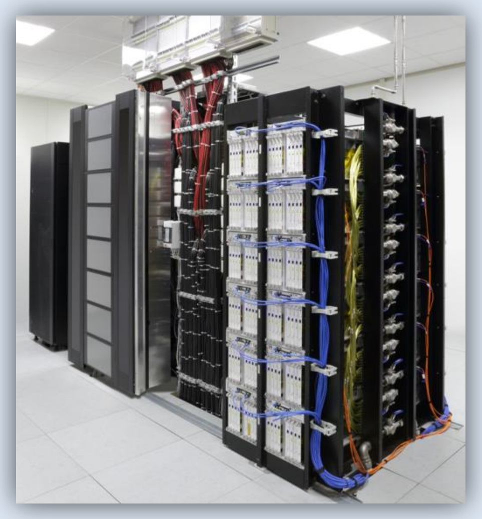


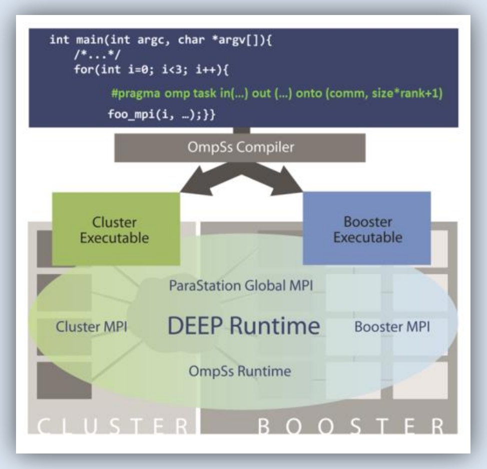
CyI – Climate Simulation



Some DEEP achievements

- **Hardware**
  - Prototyped a new architecture
  - Created unique cluster of autonomous accelerators: the Booster
  - Demonstrated EXTOLL network at scale
- **Software**
  - Made Xeon Phi autonomous via remote boot
  - Created communication bridge between networks
  - Developed efficient and portable program. environ. for heterogeneous systems
- **Applications**
  - Codes modernized and optimizated
  - Demonstrated flexibility of the Cluster-Booster concept





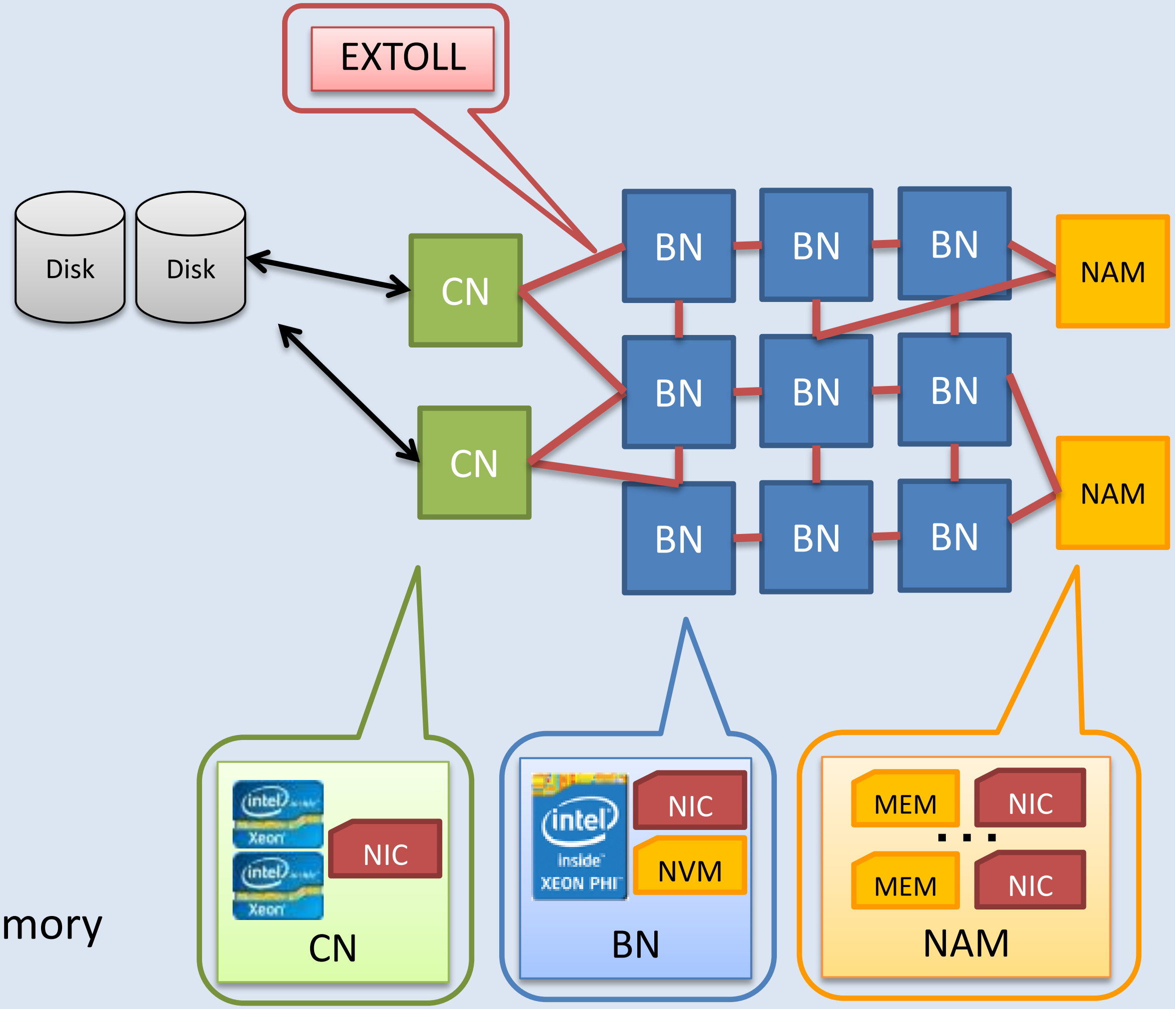
```
int main(int argc, char *argv[]){
   /*...*/
   for(int i=0; i<3; i++){
         #pragma omp task in(…) out (…) onto (comm, size*rank+1)
         foo_mpi(i, …);}}
```

FUTURE OF CLUSTER-BOOSTER ARCHITECTURE

DEEP-ER architecture

DEEP-ER addresses:

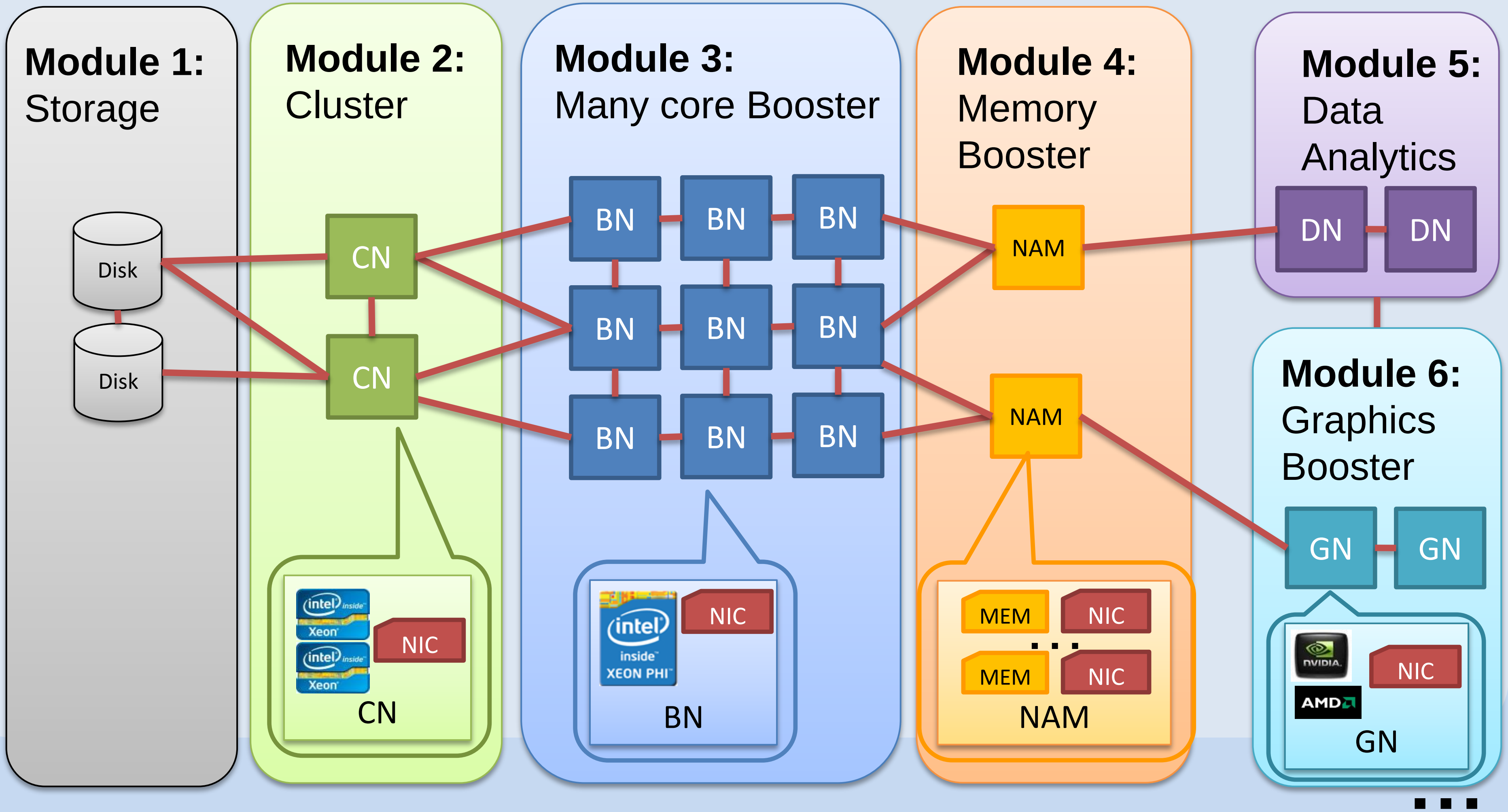
- **I/O**
- **Resiliency**



Legend:
CN: Cluster Node
BN: Booster Node
BI: Booster Interface
NAM: Network Attached Memory
NVM: Non Volatile Memory

Modular Supercomputing

Generalization of the Cluster-Booster concept



Summary

Supercomputing architectures have evolved over time:

- From special purpose machines → general purpose
- Specifically developed components → standard components
- Proprietary software → Open Source

Cluster are today the dominant supercomputing architecture

- Cost- efficient
- Flexible and adaptable
- Possibilities to make own developments

Exascale (10^{18}) brings new challenges

- DEEP and DEEP-ER try to solved them with an innovative cluster heterogeneity: the Cluster-Booster architecture
- Future: Modular Supercomputing

DEEP and DEEP-ER

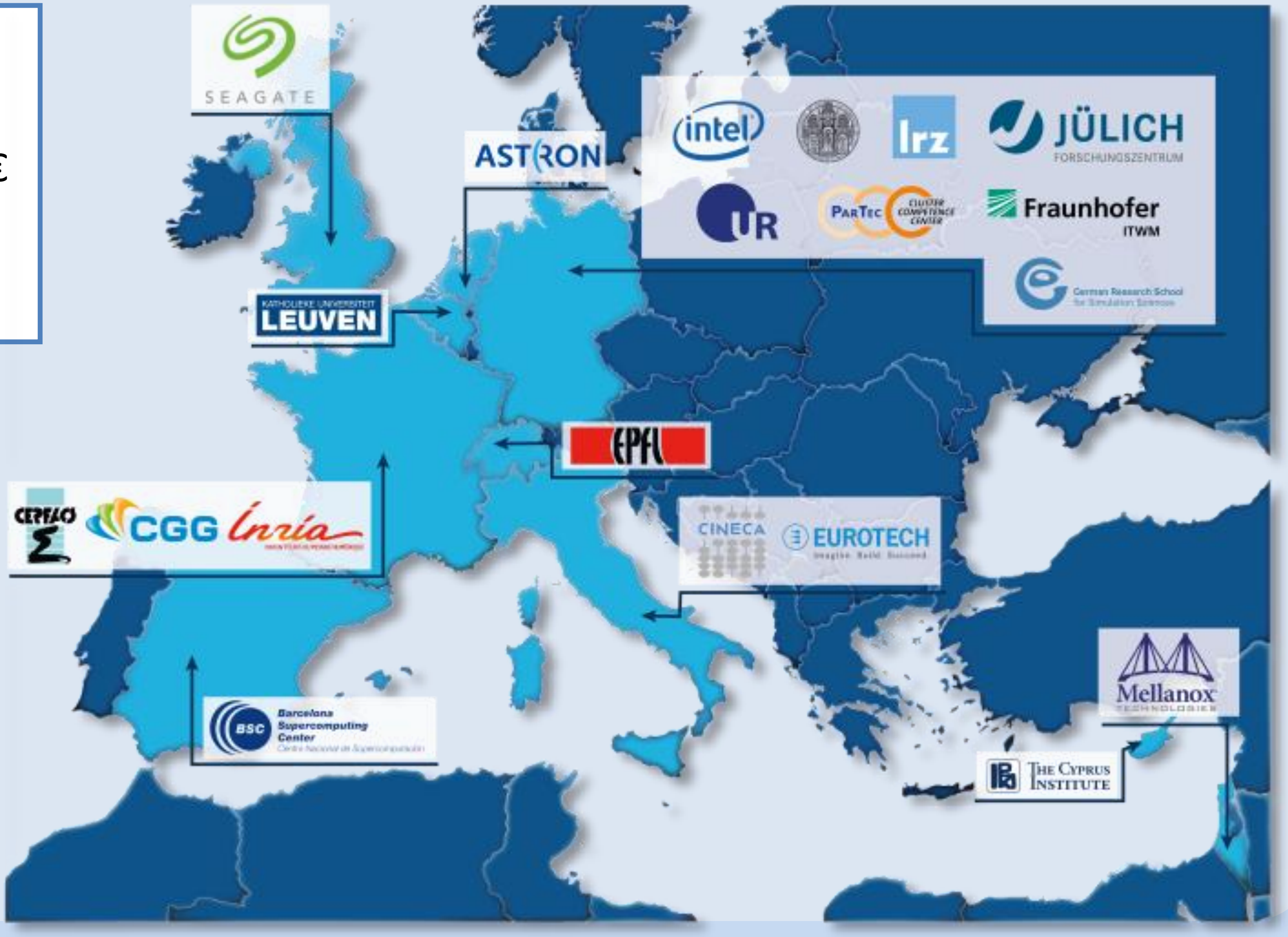
EU-Exascale projects

20 partners

Total budget: 28,3 M€

EU-funding: 14,5 M€

Nov 2011 – Mar 2017



www.deep-project.eu

www.deep-er.eu