USER BENEFITS AT A GLANCE

- Ideal for both worlds: The DEEP/-ER systems provide superior efficiency and scalability and are thus equally suited for HPC applications from **science and industry**.
- **Heterogeneous runtime characteristics:** The DEEP/-ER systems are especially suited for HPC applications with code parts that differ in their scalability characteristics (i.e. multi-scale and/or multi-physics simulations).
- Efficient system use: Separating the code into highly scalable and low scalable parts and its dynamic distribution to either Cluster or Booster ensures the **system resources are used in an optimal way.**
- **Easy-to-use:** Porting an application to the DEEP/-ER machines is a snap with the DEEP programming model – especially if you are using MPI already. The model is based on standards, protecting your investment in code modernization.
- The DEEP-ER prototype is additionally optimised for **I/O intensive** applications and exploits new memory technologies like non-volatile and network attached memory.
- The system software provides advanced features for **resiliency**, reducing the overhead of task and process checkpoints by exploiting a multi-level memory hierarchy.
- Open questions? The projects **provide support for proven, industry standard HPC programming models** like MPI and OmpSs (or OpenMP 4.x). Naturally, the relevant documentation will be made available as well.

TESTING THE DEEP/-ER SYSTEMS

The DEEP prototype system is installed at the Jülich Supercomputing Centre. After the official project end, the system is made available to interested external users to get familiar with the architecture and programming models and evaluate their potential. The same will apply for the DEEP-ER prototype.

You are interested in testing the DEEP/-ER systems?

GET IN TOUCH!

Contact DEEP:

Email: pmt@deep-project.eu

Web: www.deep-project.eu

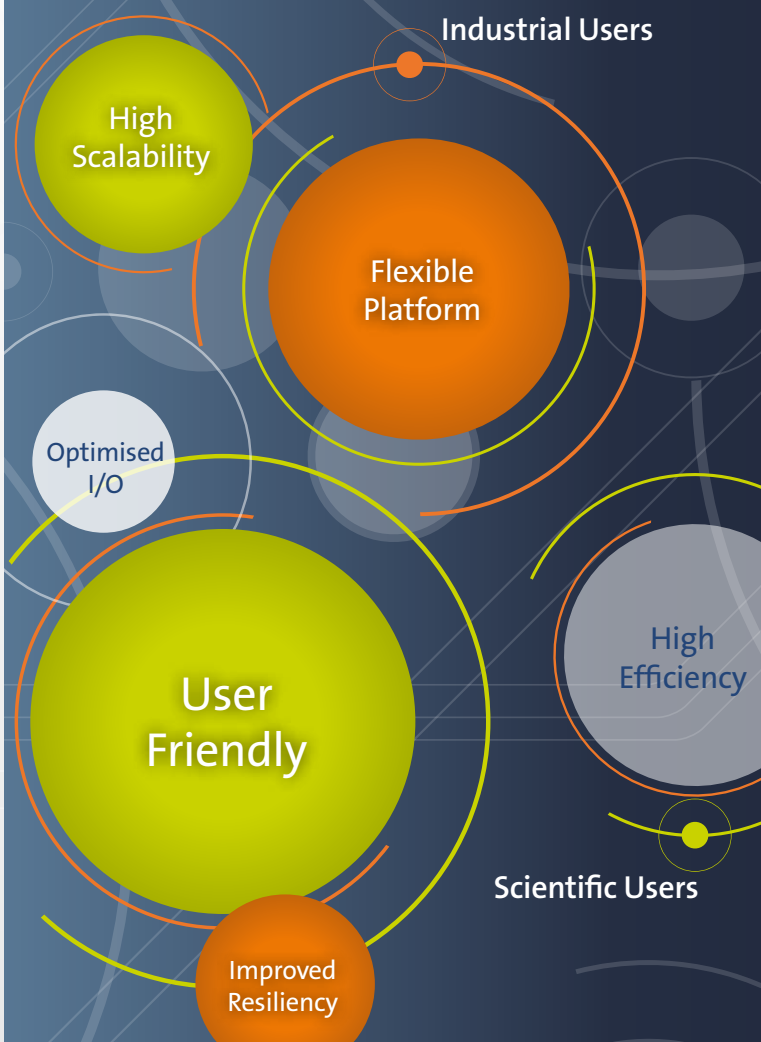
Contact DEEP-ER:

Email: pmt@deep-er.eu

Web: www.deep-er.eu

The research leading to these results has received funding from the European Commission's Sventh Framework Programme (FP7/2007-2013) under Grant Agreement numbers 287530 and 610476.

DEEP-ER

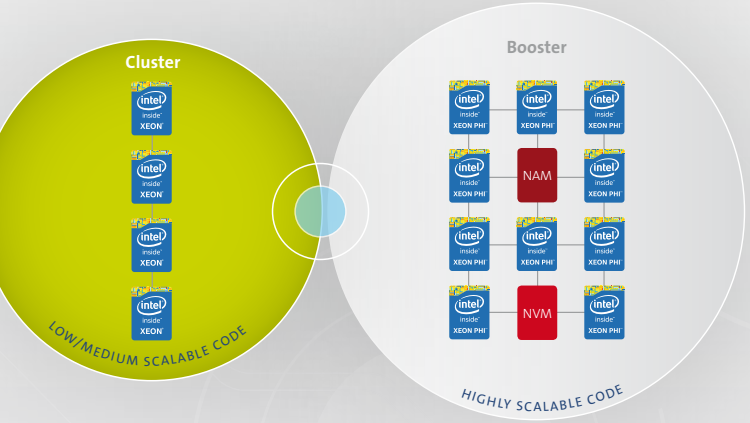


ENABLING HPC APPLICATIONS FOR EXASCALE

DEEP/-ER BASICS

The DEEP and DEEP-ER research projects develop a novel **supercomputing architecture** that will achieve **superior scalability and efficiency**. The innovative Cluster-Booster approach takes the concept of compute acceleration to a new level and combines:

- A standard Cluster using Intel® Xeon® nodes (Cluster Nodes) with
- An innovative, highly scalable Booster constructed of Intel® Xeon Phi™ co-processors (Booster Nodes)



The highly scalable code parts of an application run on the Booster, while those code parts with limited scalability benefit from the large single-thread performance of the Cluster. This code division ensures an optimal use of the system.

Proof of Concept: Applications

Eleven real-world HPC applications have been involved from the very beginning, driving a co-design process with their requirements both for the hardware and system software/API aspects. The selected applications have been continuously optimised for the Cluster/Booster architecture and for the CPU technologies used (Intel Xeon and Intel Xeon Phi). The end result are systems that fit the needs of the applications, and application showcases that demonstrate the efficiency and performance of the DEEP and DEEP-ER systems.

HOW TO USE DEEP/-ER MACHINES

To fully exploit the innovative hardware architecture in the most user-friendly way, the projects have developed a standards-based, straightforward software stack for application developers.
ParaStation MPI and OmpSs are the two key elements:

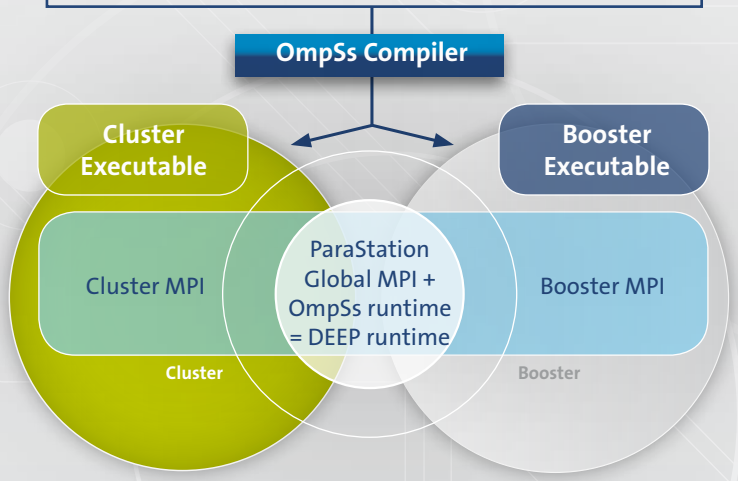
ParTec's ParaStation MPI serves as a Global MPI covering both the Cluster and the Booster part of the hardware, enabling:

- High performance communication within Cluster and Booster
- High bandwidth communication between Cluster and Booster
- Dynamic offloading of code through the MPI-2 process management interface

The OmpSs programming model extends OpenMP and provides powerful offload features:

- Transparent offload of MPI kernels between Cluster and Booster
- Support for dynamic resource allocation

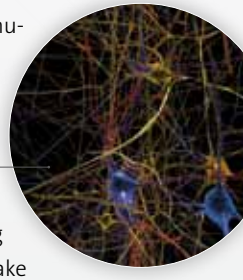
```
int main(int argc, char *argv[]){
  /*...*/
  for(int i=0; i<3; i++){
     #pragma omp task in(...) out(...) onto(comm, size*rank+i)
     foo_mpi(i, ...);}}
```



DEEP/-ER USE CASE

Towards a better understanding of the brain with DEEP

Brain simulation is making giant leaps towards a better understanding of the brain's inner workings. In DEEP, the Swiss Federal Institute of Technology in Lausanne (EPFL) has been adapting CoreNeuron – an advanced brain simulation application – to run efficiently on the platform.



Optimising CoreNeuron for manycore architectures

In manycore architectures, efficient threading and vectorisation are no longer optional. To take advantage of them, these changes were necessary:

- An elaborated load balancing strategy at the thread level, taking into account the complexity of simulating different kinds of neurons.
- Data layout changes and refactoring of the compute loops to enable vectorisation.

The result: The effects of the refactoring are already very noticeable. The simulation now achieves a very high level of parallel efficiency and is able to take advantage of using 240+ threads. Memory layout transformation and vectorisation optimisation lead to enhanced performance even on memory bound kernels.

Going even further with DEEP

Being a highly scalable application, CoreNeuron is already running on some of the most powerful supercomputers on the planet. On the DEEP system, it can leverage one of the highest levels of parallelism very efficiently and explore future heterogeneous system designs. The key to success:

- Decoupling the I/O from the computation.

The result: A speedup of more than one order of magnitude with respect to performing I/O directly from Xeon Phis in some preliminary tests on standard platforms. The design of the OmpSs offload allows direct (Cluster to Booster) and reverse (Booster to Cluster) offloading in the same way, which allows to implement the reverse offload followed in CoreNeuron in a particularly easy way.