Chapter 9

Modular Supercomputing architecture: from idea to production

Estela Suarez, Norbert Eicker, and Thomas Lippert

Jülich Supercomputing Centre - Forschungszentrum Jülich GmbH, Leo Brandt Strasse, 52428 Jülich (Germany)

This chapter describes a new heterogeneous architecture created at the Jülich Supercomputing Centre (JSC), its motivation and the development path that led to its final realization. Because it covers a broader scope involving several hardware platforms, the chapter is structured in the following way: first an overview of the Jülich Supercomputing Centre is given, to explain its context and background. In section 1.2 a historical view of the architectures and main systems co-developed at JSC is given, which led to the actual Modular Supercomputing architecture. Section 1.3 describes the JSC application portfolio and how the newly developed architecture aims to match it. An overview of the systems built following the Modular Supercomputing approach is given in section 1.4, elaborating on their detailed hardware implementation in section 1.5. The software and programming environments implemented to support the new architecture are described in sections 1.6 and 1.7, respectively. Section 1.8 treats the cooling and facility infrastructures. Finally, section 1.9 concludes the chapter with a look into the future of the Modular Supercomputing architecture.

1.1 The Jülich Supercomputing Centre (JSC)

The *Forschungszentrum Jülich* (FZJ) [32] is with a staff of over 5,000 employees one of Europe's largest research centers. It pursues cutting-edge interdisciplinary research to address the grand challenges facing society in the fields of health, energy and environment, and information technologies. As a member of the German research *Helmholtz Association* [29] (HGF), FZJ hosts world-class instruments used by internal and external researchers to conduct their work: supercomputers for simulations, unique analytical and characterization equipment, imaging techniques for medicine, nanotechnology tools, etc.

The *Jülich Supercomputing Centre* [33] (JSC) within FZJ plays a key role in the above-mentioned activities. Founded in 1987, JSC has extensive expertise in providing supercomputer services and support to national and international user communities. With over 200 employees, JSC is the largest of the three national supercomputing centres in Germany. In 2009 JSC became the first European supercomputing center with Petaflop capability in PRACE (*Partnership for Advanced Computing in Europe [20]*), which bundles European computing centres to collectively offer computing resources to users.

The operation of large scale HPC-systems is only one of the aspects that JSC covers to provide supercomputing services to the wider scientific community. Additionally, JSC strives to guarantee optimal user support, continuously developing the simulation methodology, parallel algorithms and new programming and visualisation techniques, as well as carrying out intensive own research in core areas of the computational sciences. Researchers who apply for computing time at the JSC's systems are supported by a continuously growing number of domain-specific Simulation Laboratories (*SimLabs* [51]). At present, JSC runs nine SimLabs covering diverse fields such as Biology, Plasma Physics, Climate Sciences, and Neuroscience. The SimLabs offer support to specific scientific communities, and contribute through co-design to the research and development of HPC technologies.

Furthermore, JSC actively participates in designing and building its next generation supercomputers. Key in this context are the in-house development of open source software and tools, and the development of innovative HPC architectures. For this purpose tight, long-term collaborations (the so-called *Exascale Laboratories*) have been established with the world-leading technology providers IBM, Intel, and NVIDIA. The Exascale Labs, in combination with German- and European-funded R&D initiatives, enable JSC applying co-design strategies and exploring new concepts and technologies.

1.2 Supercomputing architectures at JSC

Through very close collaboration with the providers, JSC is able to address its users' requirements when procuring production systems. To achieve the best possible price-performance ratio the focus is put in the selection and optimal integration of the best suited off-the-shelf components. In particular, JSC has since long designed the architecture of its systems employing its expertise in cluster configuration, network topologies, and cluster management software [53].

New concepts and technologies are first tested in prototypes and, once validated, employed in the next generation large scale production systems. For this purpose a number of small to medium size computing systems is being operated at JSC. Current examples are an Intel Xeon Phi (KNL) system for the lattice-QCD community (*QPACE3* [48]), the two pilot systems for the Human Brain Project [27] (*JURON* and *JULIA*), and the DEEP prototypes [45]. The latter will be explained in more detail in section 1.5.

Based on the experience gather with prototypes, the design of the JSC production systems is chosen. The following three subsections describe how this work has led to an architectural evolution at the computing centre: from a dual supercomputing approach to a Modular Supercomputing architecture.

1.2.1 The dual supercomputer strategy

Back in 2004, JSC constructed a modern computer room to host its new ***JUMP*** system: an IBM p690 Cluster with a total of 1,312 processors and 5 Terabytes internal memory (128 gigabytes per node). JUMP achieved initially a maximum performance of 5.6 TFlop/s (it became number 21 in the TOP 500 list [54] at the time of its inauguration), to be later scaled-up to a total of

9 TFlop/s. However, already quite early in the operation cycle of JUMP, the JSC-team realized that for the given number of users and allocation of computing time, a further growth of capability computing on JUMP would have actually decreased the overall efficiency. The reason was that many applications were not scaling well to very large number of processes. These users, who were working at their sweet spot, would not have profited from policies preferring jobs with very large processor numbers.

As a solution, the ***dual supercomputer strategy*** [40] was conceived. It consisted on running two large-scale HPC-systems simultaneously: a highly flexible, general purpose cluster (for low/medium scalable applications), and a massively-parallel system (for highly scalable applications).

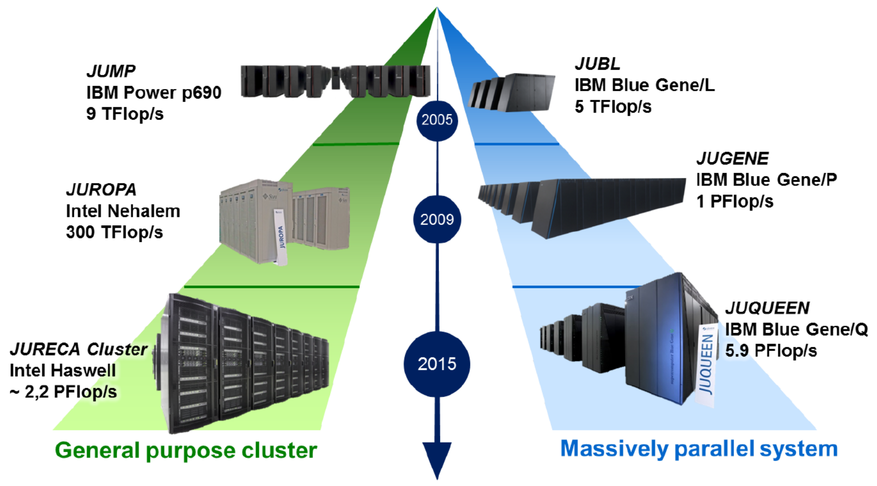


FIGURE 1.1: Dual architecture strategy driven at JSC for the past years.

The realisation of the dual concept started in 2005/2006 with the installation of ***JUBL***, a 46 Teraflop/s BlueGene/L system with 16,384 processors. Before being admitted to the machine, the scalability of a code had to be proven (up to a minimum of 1,024 processors). At the same time, JSC made a strong investment in terms of personnel resources to provide training and support for users in order to scale-up applications and increase the JUBL portfolio. This lead to a substantial improvement on scalability for several applications, a number of them up to the full machine.

However, in many cases the scientific problem was intrinsically in conflict with a massively parallel execution of the code and ran much more efficiently on the JUMP cluster. This finding confirmed the importance of continuing the dual hardware strategy, regularly upgrading and modernising its two branches in a staged manner, with each system reaching a life-time of about five years (see figure 1.1).

General purpose cluster branch The JUMP system was replaced in 2009 by ***JUROPA***, a large compute cluster increasing the power of its predecessor by more than a factor of 25. This cluster was designed and built by JSC together with partners from industry. The operating system software running on the machine was *ParaStation*, developed jointly by ParTec (a Munich based software company) and JSC. With a peak performance of about 300 TFlop/s and 79 Terabyte main memory, *JUROPA* was the 10^{th} fastest computer of the world at the time of deployment.

JUROPA was substituted in July 2015 by the 2.2 PFlop/s ***JURECA Cluster*** [34], which continues in operation today. Its configuration is described in detail in section 1.5.3.

Massively parallel branch Given the success of JUBL and to keep-up with the large user demand, JSC deployed already by end of 2007 the IBM BlueGene/P system ***JUGENE***, which was the second fastest computer on the world at the time of installation (222.8 TFlop/s peak performance). After a later scale-up [5], JUGENE reached 1 PFlop peak performance and was kept in operation until 2012.

At that time the IBM BlueGene/Q system ***JUQUEEN*** [52] was installed. With almost 6 PFlop/s performance, nearly half a million processors (28,672 nodes with 16 cores each) and 448 Terabytes main memory (16 GB/node), JUQUEEN became the fastest system in Europe at the time of installation. This water-cooled system was also among the most energy efficient supercomputers in the world, with a performance/power ratio of approximately 2 Gigaflop/s per Watt. To help users migrating their codes and leverage performance, porting and tuning workshops were organised. In this context, the *High-Q Club* [30, 6] was established: a showcase for production codes able to scale up to the entire JUQUEEN, i.e. capable of using all its 458,752 cores.

JUQUEEN continues in operation today and its decommissioning is planned for Q1/2018.

Storage system Data movements between the JSC machines are minimised by running a common *GPFS* file system on all of them. This storage system is mounted from the central Jülich Storage Cluster (***JUST***) [31], which is in fact the only physical connection between the two branches of the dual supercomputing approach. To keep pace with the performance increase of the compute systems, also JUST is regularly upgraded. The last JUST upgrade was done in 2013, close after JUQUEEN's installation [6]. As a result, the storage bandwidth increased from about 60 GByte/s to 200 GByte/s (by increasing the number of disks), and a much better protection against failures was achieved (thanks to the installation of the GNR (GPFS Native Raid) software). The next generation of JUST is planned for early 2018.

1.2.2 The Cluster-Booster concept

Even if a user can have access to both JUQUEEN and JURECA, distributing a code over both platforms is cumbersome due to their different software environments and the lack of an inter-system high-speed network connection. To fill the gap between the two branches of the dual supercomputing strategy the ***Cluster-Booster architecture*** [15] was created. Primary goals were to enable a larger amount of codes exploiting the advantages of highly scalable systems, and improving the energy-efficiency and scalability of cluster computers.

For the latter, the standard approach is building clusters using heterogeneous nodes, in which multi-core CPUs are hosting one or more accelerators (many-core CPUs or graphic processing units (GPU)). This *host-device(s)* approach is efficient and scalable, but presents some caveats: latency penalties and bandwidth limitations on the GPU-to-GPU communication due to the node's PCIe bus, GPU-specific programming environments requiring high code re-factoring efforts, static assignment of GPU to CPU resources, etc.

The Cluster-Booster architecture proposed to integrate heterogeneous computing resources at the system level, instead of doing so at the node level (see figure 1.2). This translates into extracting the accelerators from the host nodes and moving them into a stand-alone cluster of accelerators, which received the name of ***Booster***. The Booster was attached to a standard HPC-Cluster via a high-speed network. This connection, together with a uniform software stack running over both parts of the machine, enables them to act together and be used by applications as one single system.

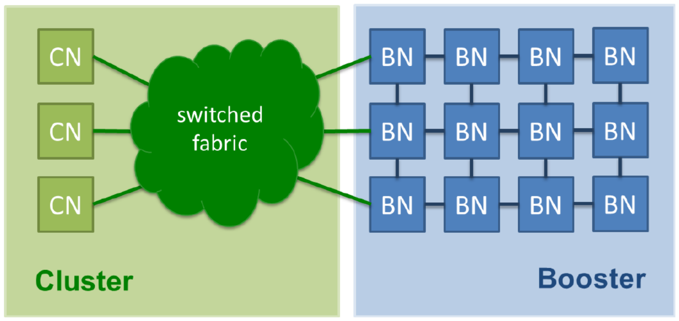


FIGURE 1.2: Sketch of the Cluster-Booster architecture. *CN*: Cluster node (general purpose processor), *BN*: Booster node (autonomous many-core processor).

The ratio of the amount of work to be executed by the commodity CPUs in the Cluster and the accelerators in the Booster is different in each application. Accordingly, in the Cluster-Booster architecture no constraints are put on the combination of nodes that an application may select, and resources are reserved and allocated dynamically. This has two important effects: Firstly each application can run on a near-optimal combination of resources and achieve excellent performance. Secondly all the resources can be put to good use by a system-wide resource manager allowing combining the set of applications in a complementary way, increasing throughput and efficiency of use for the overall system.

Until recently the type of kernels to be offloaded onto accelerators was very limited due to their inefficient data-exchange capabilities with other accelerators. In the Booster, however, accelerators communicate directly with each other through the high-speed network, allowing for full codes with intensive internal communication to run on the system. In this way, the Booster can be regarded as a massively parallel system on its own, so that highly scalable codes running well on BlueGene should fit also very well to the Booster. Applications with low/medium scalable parts (as the ones described in section 1.3) can run the highly scalable parts on the Booster and leave those less scalable to profit from the high single-thread performance of the Cluster.

The Cluster-Booster architecture enables for the first time mapping the intrinsic scalability patterns of applications onto the system hardware. In this way, the limitation posed by the less-scalable parts of codes is alleviated, and the overall scalability of the full code should improve.

The first two prototypes of the Cluster-Booster architecture were developed and built within the European-funded research projects[1] ***DEEP*** [44] (*Dynamical Exascale Entry Platform*) and ***DEEP-ER*** [43] (*DEEP-Extended Reach*). The DEEP prototype and its software stack were designed to support mainly HPC applications. In DEEP-ER additional non-volatile memory (NVM) layers were integrated and network-attached memory technologies were prototyped. Both extensions enabled advanced scalable high-performance I/O and resiliency strategies. The precise hardware and software configurations of the DEEP and DEEP-ER prototypes are described in sections 1.5.1 and 1.5.2, respectively.

The Cluster-Booster approach is now going into production, with the recent installation of a Booster attached to the JURECA Cluster. The JURECA Booster, a 5 PFlop Intel Xeon Phi (KNL) system, has been deployed end of 2017. Its precise hardware configuration is described in section 1.5.3.

[1]Along this chapter the term *DEEP projects* will be used when referring globally to the research projects DEEP, DEEP-ER, and DEEP-EST. These three European funded projects are strongly connected with each other, but do focus on different research topics. The DEEP project (2011-2015) introduced the Cluster-Booster concept, building the first hardware prototype and developing its full software and programming stack. The DEEP-ER project (2013-2017) focused on I/O and resiliency: it built a small-size, memory-enhanced Cluster-Booster prototype and implemented advanced I/O and resiliency software. The DEEP-EST project (2017-2020) will realise a Modular Supercomputing prototype addressing the requirements of HPC and HPDA users.

1.2.3 The Modular Supercomputing architecture

While the Cluster-Booster architecture is being implemented in production, the JSC-team works already in the development of its next generation systems. The underlying concept in a Cluster-Booster system is to break with the tradition of replicating many identical (potentially heterogeneous) compute nodes, and integrate instead the heterogeneous computing resources at the system level. This concept is being extended and generalised in the ***Modular Supercomputing architecture***.

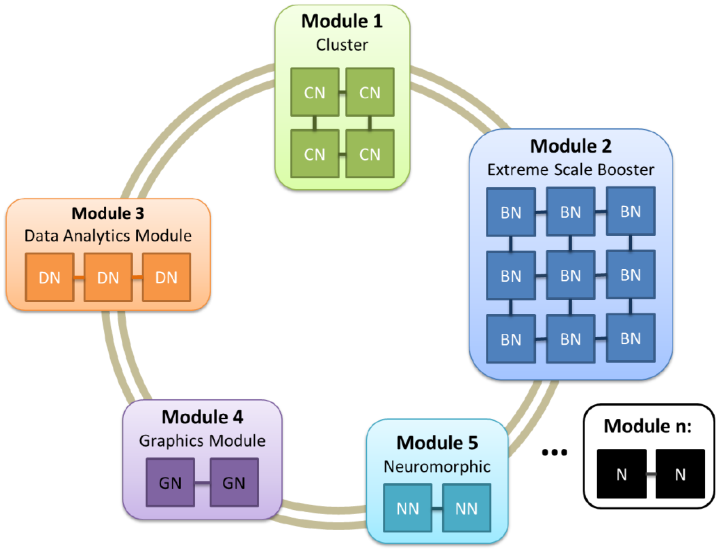


FIGURE 1.3: Sketch of the Modular Supercomputing architecture (*CN*: Cluster node, *BN*: Booster node, *DN*: Data Analytics node, *GN*: Graphics node, *NN*: Neuromorphic node).

The Modular Supercomputer architecture (figure 1.3) connects compute modules with different hardware and performance characteristics with each other to create a single heterogeneous system. Each module is a parallel, clustered system of potentially large size. A federated network connects the module-specific interconnects. This approach brings substantial benefits for heterogeneous applications and work-flows since each part can be run on an exactly matching system, improving time to solution and energy use. It is therefore ideal for supercomputer centres running heterogeneous application mixes (higher throughput and energy efficiency). The Modular Supercomputing architecture also offers valuable flexibility to system operators, allowing the set of modules and their respective size to be tailored to the centre's actual portfolio and usage. Modules with disruptive technologies, such as neuromorphic or quantum devices, can also be included in a Modular Supercomputer to satisfy the needs of specific user communities.

The Cluster-Booster machines deployed until now (see section 1.5) are in principle Modular Supercomputers with two modules. However, their designs and software stacks was not foreseen for further extensions. The real generalisation of the concept will be first realised in the ***DEEP-EST*** (*DEEP - Extreme Scale Technologies*) research project, which started in July 2017 and will run for three years. One of the most important contributions expected from DEEP-EST is the development of resource management software and scheduling strategies to deal with any given number of compute modules. To demonstrate its capabilities, a three-module prototype will be built, which shall cover the needs of both HPC and high performance data analytics (HPDA) workloads.

The implementation of the Modular Supercomputing architecture in a large-scale production system will start already in parallel to DEEP-EST. The installation of the ***JUWELS*** Cluster is planned already in Q2/2018, with further modules coming around the 2020 time-frame.

1.3 Applications and workloads

The provision of supercomputer resources at JSC is executed by the *John von Neumann Institute for Computing* (NIC). This independent institution organises and coordinates half-yearly a peer-review process with international expert-referees that evaluate the project-applications and allocate the computing time according to their scientific excellence. This process, which has been meanwhile adopted by other German and European HPC centers, ensures an effective usage of the HPC infrastructure. Users come mostly from German and European universities and research institutes, with a smaller proportion of industrial provenance. The relative quantity of regional, national, European and international users is determined by the funding sources that contribute to cover the costs of the computers procurement and operation. This leads to JUQUEEN (which is funded by national sources), being used mostly by German and European users, while JURECA (funded by the Helmholtz Association) serves mainly regional and national users.

The main role of JSC is enabling outstanding research in fields encompassing astrophysics, computational biology and biophysics, chemistry, earth and environment, plasma physics, computational soft matter, fluid dynamics, elementary particle physics, computer science and numerical mathematics, condensed matter, and materials science. Accordingly, the application portfolio running on the JSC systems is highly multidisciplinary, as outlined in figure 1.4. These applications are very diverse not only with regards to the research fields that they address, but also to the algorithms, numerical methods and parallelisation strategies that they employ. Therefore, all computer architectures developed at JSC aim at fulfilling a very wide set of user requirements.

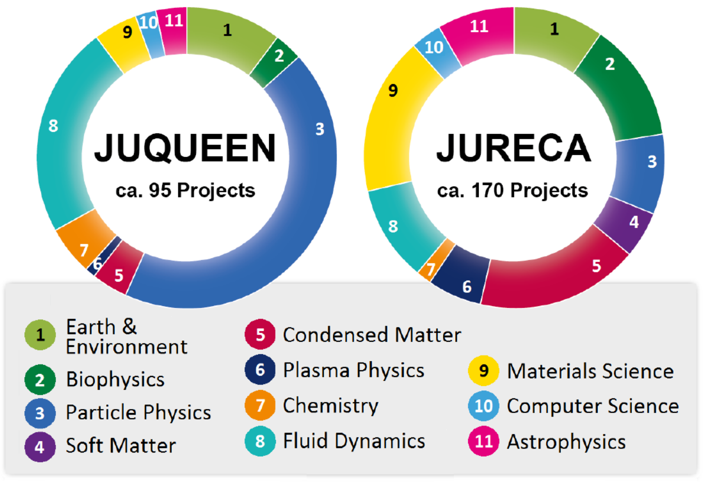


FIGURE 1.4: Computing time assigned to different research fields on the JUQUEEN and JURECA systems (as of May 2017).

The dual supercomputing strategy described in section 1.2.1 has been very successful in fulfilling the needs of both highly scalable HPC applications (with the massively parallel platform), and low/medium scalable codes requiring high single-thread performance (with the general purpose cluster). Analysing the respective applications, one can observe that the highly scalable codes present often very regular communication patterns and manage relatively reduced amount of data, while the less scalable ones manage more data and access it using more complex communication patterns.

However, a more detailed analysis revealed that many of the latter codes include highly scalable code-parts, too. In particular, applications simulating multi-scale or multi-physics problems, or coupling different models to describe highly complex phenomena, do show a mix of concurrency patterns: parts of these applications are highly scalable, while others are intrinsically limited in scalability. Due to Amdahl's law these codes are not able to scale-up properly in a BlueGene/Q machine like JUQUEEN. However, ignoring their high-scalable parts and running them on the general purpose cluster would waste both resources and power. The Cluster-Booster concept described in section 1.2.2 was designed to address this issue.

New user communities with new requirements are joining the JSC portfolio. Examples are High-Performance Data Analytics (HPDA), machine learning, and interactive supercomputing. One of the main challenges raised by the convergence between HPC and HPDA is finding an architecture that can match well the requirements of both application fields.

Traditional HPC applications are usually iterative and rely heavily on a small number of numerical algorithmic classes (like the original Berkeley *seven dwarfs* [4]) that operate on relatively small data sets and accrue very high numbers of floating point operations across iterations. HPC systems have been optimised according to these requirements, and it felt justified to rank these machines purely on their Flop/s performance for DGEMM [54]. With the years this has led to rather monolithic systems where the amount of memory per core is steadily decreasing.

However, the complexity and memory requirements of HPC codes are increasing, leading to a dissonance with these traditional systems. In addition, the desire to support the HPDA workloads rapidly emerging from the *Big Data* community clearly requires a change in systems architecture, since these will exhibit less *arithmetic intensity* and require additional classes of algorithms to work well (see e.g. advance deep learning neural network algorithms). Moreover some scientific fields like brain research are expected to make use of both technologies – HPC and HPDA – to the same degree in the future. The Modular Supercomputing Architecture (see section 1.2.3) has been designed precisely to provide a platform best fitting diverse, increasingly complex, and new-coming applications.

To make sure that the implementations of first the Cluster-Booster and now the Modular Supercomputing architecture really became usable and beneficial for a large and multidisciplinary portfolio of applications as the one shown in figure 1.4, a number of *co-design* applications were included in the DEEP projects.

1.3.1 Co-design applications in the DEEP projects

A total of 15 application teams bringing full-fledged HPC and HPDA applications are or have been actively involved in the DEEP projects. Their requirements strongly influence the choice of hardware components for the prototypes, which functionality the software and programming environment provide, and how the architecture itself is evolving in time. At the same time, code-modernisation has brought significant improvements to the applications themselves, in terms of better performance, new capabilities, and higher resilience to failures. In this context, it is important to mention that the changes included in the codes are not exclusive to run on a Modular Supercomputer, but are more general and in the end beneficial on any modern HPC system.

The DEEP co-design applications are listed below, including their field of research, the name of the application, and the institution driving the particular application effort within the DEEP projects. More details on the codes are available at the given bibliographic references.

- **Neuroscience simulations**: NEURON [37] by EPFL (Ècole Polytechnique Fédérale de Lausanne) and NEST [41] by NMBU (Norges Miljø- og Biovitenskapelige Universitet).

- **Space weather simulation**: iPiC3D [55] and xPic [35] by KU Leuven (Katholieke Universiteit Leuven).
- **Climate simulation**: EMAC [12] by CYI (The Cyprus Institute).
- **Computational fluid engineering**: AVBP [10] by CERFACS (Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique).
- **High temperature superconductivity**: TurboRVB [17] by CINECA (Consorzio Interuniversitario del Nord-Est per il Calcolo Automatico).
- **Seismic imaging**: RTM [8] by CGG (Compagnie Générale de Géophysique) and FWI [46] by BSC (Barcelona Supercomputing Center).
- **Human exposure to electromagnetic fields**: GERShWIN [39] by INRIA (Institut National de Recherche en Informatique et en Automatique).
- **Computational earthquake source dynamics**: SeisSol [50] by LRZ (Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften).
- **Radio astronomy**: the SKA data analysis pipeline [47] by ASTRON (Netherlands Institute for Radio Astronomy).
- **Lattice QCD**: CHROMA [38] by the University of Regensburg.
- **Molecular dynamics**: GROMACS [26] by NCSA (Bulgarian National Center for Supercomputing Applications).
- **Data analytics in Earth Science**: piSVM and HPDBSCAN [25] by the University of Island (Haskoli Islands).
- **High Energy Physics**: CMSSW [11] by CERN (Conseil Européen pour la Recherche Nucléaire).

The above codes have demonstrated the high flexibility of the concept, which enables very different use models on the system. Developers of monolithic, high scalable applications (e.g. CHROMA) typically run their codes fully on the Booster. Applications combining different physical models, such as EMAC – which couples and atmospheric with a chemistry model – were distributed between various modules according to their concurrency levels (Cluster and Booster in the EMAC case). Results obtained with xPic demonstrate the benefit of mapping this way the specific characteristics of an application onto the hardware modules [35]. Codes aiming at interactive supercomputing such as CoreNeuron ran the simulation on the Booster, steering and visualising it on the Cluster side. Also users running application pipelines employ different modules for each component of their work flow. This is the case of the space weather application from KU Leuven, which analyses satellite data on the Data Analytics Module, uses the Cluster to calculate the propagation of the ejected solar particles from Sun to Earth, and distributes their interaction with the Earth's magnetosphere between Cluster and Booster.

The large variety of fields covered by the DEEP co-design applications is well visible in the list above. Their co-design input, based on their diverse requirements and code characteristics, has led to the system design described in the following sections.

1.4 Systems overview

The **Modular Supercomputing Architecture** proposes a new way of integrating heterogeneous hardware at system level. To demonstrate the potential of this new philosophy and progressively refine and improve different aspects of the concept, various prototype systems have been built before its final realisation in a production machine.

1. The DEEP prototype: the first Modular Supercomputer ever built, with one Cluster and two Booster modules. The second of these Boosters, a small-size prototype called GreenICE Booster, was built to test a newer generation network fabric and an innovative immersion cooling technology (see section 1.5.1).
2. The DEEP-ER prototype: this second generation Cluster-Booster system is of much smaller size than the previous one, but is enhanced with a multi-level memory hierarchy that enable new I/O and resiliency strategies.
3. The JURECA production system: the existing JURECA Cluster has recently (end 2017) been complemented with the 5 PFlop JURECA Booster, becoming the first Modular Supercomputer in production.

Their hardware components (details in section 1.5) have been chosen taking the newest technology available at each point in time. The software stacks (see sections 1.6 and 1.7) are very similar in all platforms and aim always at exposing the standard functionality and programming environments commonly available in HPC systems.

1.4.1 Sponsors

European, national (German), and regional (North Rhine-Westphalia) funding sources have contributed to the development, construction and procurement of the above mentioned Modular Supercomputers.

The DEEP projects (DEEP, DEEP-ER and DEEP-EST) received funding from the European Commission's programs for research, technological development and demonstration, at funding rates that vary depending on the framework program and the partner provenance (academia vs. industry). The matching co-finance comes from internal budget of the partners involved in the projects. JSC co-financed its own personnel costs and the hardware components. Additionally, JSC paid fully the operational costs for the prototypes. JSC's internal budget comes from national sources, channelled through the Helmholtz Association (HGF). Ultimately, the amount of funding received from the HGF comes to 90% from the German Federal Ministry of Education and Research, and to 10% from the North Rhine-Westphalia Ministry for Culture and Science.

Funding for the procurement and operation of the JURECA system, on the other hand, is of pure national (German) origin and was granted by the Helmholtz Association through the program *Supercomputing & Big Data*.

1.4.2 Timeline

Figure 1.5 shows the time frames in which the DEEP and JURECA systems have been designed, developed, and deployed.

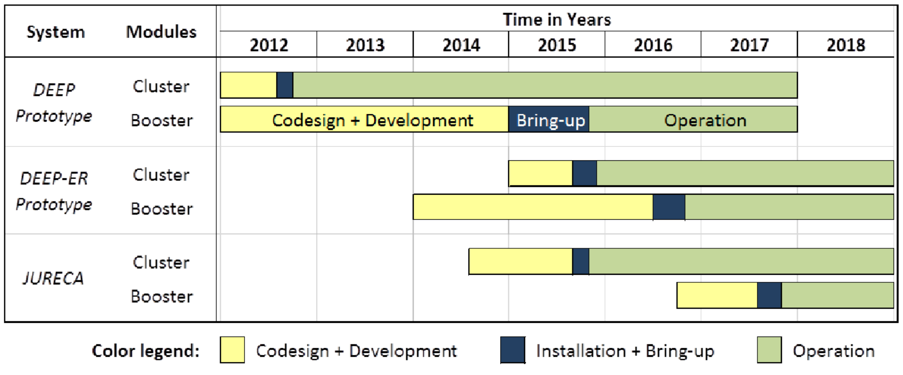


FIGURE 1.5: Timeline for the development of the DEEP, DEEP-ER and JURECA systems.

The DEEP prototypes[2], and specially the first generation (DEEP), are the result of ambitious research and development projects. The first (DEEP) Booster, in particular, was constructed using custom hardware components, designed and built exclusively for the project. This approach requires higher development effort and implies also higher risks, which translate into long

[2]When using the term in plural (*DEEP prototypes*), the authors refer globally and indistinctly to the hardware platforms deployed in the DEEP and DEEP-ER projects. *DEEP prototype* refers to the first generation (built in the DEEP project), and *DEEP-ER prototype* to the second (built in the DEEP-ER project).

design, construction, and bring-up phases for this system. There was however no alternative, since at the time no off-the-shelf cluster of autonomous accelerators was available on the market.

The situation changed with DEEP-ER, thanks to new self-booting many core processors appearing in the market. Therefore, this time the prototype is based on standard components, what accelerated the whole process. Custom hardware was built only for the network attached memory (see section 1.5.2), which is included in figure 1.5 as part of the Booster.

The JURECA system has been developed in co-design with the provider companies. To satisfy production quality and stability of the system, a much more conservative philosophy was applied, based on the selection of off-the-shelf components. This approach largely reduces the design and development phases and simplifies the bring-up and installation processes.

1.5 Hardware implementation

The Cluster parts of the DEEP prototypes and JURECA are relatively standard general-purpose systems, employing the newest Intel Xeon generation at the time of their deployment. Their Boosters are based on Intel Xeon Phi many-core processors. The first generation (Knights Corner, KNC) was used in DEEP to create a unique custom-hardware Booster, while the second generation (Knights Landing, KNL) was integrated in DEEP-ER and JURECA using off-the-shelf components.

Table 1.1 gives an overview of the hardware configuration for all the systems built at JSC following the Modular Supercomputing architecture. For the sake of brevity, only the main compute parts of the systems are included in the table. In the first generation DEEP prototype only the large scale Cluster and Booster are described, leaving the small GreenICE Booster for section 1.5.1. In the case of the JURECA Cluster only the compute nodes are described. For the sake of presentation the GPU accelerated partition of JURECA – which amounts to only 4% of the system – is skipped. For more details on the JURECA Cluster configuration, including a description of its 12 visualization nodes and 12 login nodes, see [34].

The remainder of this section describes the hardware implementation of the three systems in more detail. System software and programming environments are explained in sections 1.6 and 1.7, respectively. Information on the room infrastructure is given in section 1.8.

1.5.1 First generation (DEEP) prototype

The DEEP prototype consists of two modules (Cluster and Booster), served by a common set of login nodes and external storage servers. All

TABLE 1.1: Hardware Configurations

	Feature	**DEEP (1^{st} generation prototype)**	**DEEP-ER (2^{nd} generation prototype)**	**JURECA**
Cluster	Deployment date	2012	2015	2014
	Integrator	Eurotech	SAR	T-Platforms
	Chassis architecture	Aurora		V5050
	Node architecture	Aurora Blade	SuperMicro	V-Class V210S/F
	PCIe	Gen 2	Gen 3	Gen 3
	CPU	Intel Xeon	Intel Xeon	Intel Xeon
	processor number	E5-2680	E5-2680v3	E5-2680v3
	microarchitecture	Sandy Bridge	Haswell	Haswell
	cores(threads)	2×8(32)	2×12(48)	2×12(48)
	frequency (GHz)	2.7	2.5	2.5
	CPUs per node	2	2	2
	memory per node (GB)	32 RAM	128 RAM 400 NVM	128/256/512 RAM
	Network technology	InfiniBand	EXTOLL	InfiniBand
	generation	QDR	Tourmalet A3	EDR
	topology	fat-tree	3D torus	full fat-tree
	Compute Racks	1/2 (double sided)	1/3	29
	Cluster node count	128	16	1,872
	Cluster peak performance (TFlop/s)	45	16	1800(CPU) + 440(GPU)
Booster	Deployment date	2012	2015	2014
	Integrator	Eurotech	none (JSC)	DELL
	Chassis architecture	Aurora		
	Node architecture	Aurora Blade	Intel Adams Pass	C6320P
	PCIe	Gen 1	Gen 3	Gen 3
	CPU	Intel Xeon Phi	Intel Xeon Phi	Intel Xeon Phi
	processor number	7120X	7210	7250F
	microarchitecture	KNC	KNL	KNL
	cores(threads)	61(244)	64(256)	68(272)
	frequency (GHz)	1.2	1.3	1.4
	CPUs per node	1	1	1
	memory per node (GB)	16 RAM	16 MCDRAM 96 DDR4 400 NVM	16 MCDRAM 96 DDR4
	Network technology	EXTOLL	EXTOLL	Omni-Path
	implementation	FGPA (Altera Stratix V)	ASIC Tourmalet A3	
	topology	3D torus	torus	full fat-tree
	Compute Racks	1 (double sided)	1/4	23
	Booster node count	384	8	1640
	Booster peak performance (TFlop/s)	500	20	5000
Total	Racks	2 (double sided)	1	52
	Total peak performance (TFlop/s)	545	36	7200

together the DEEP prototype reaches a total peak performance of about 550 TFlops. Part *a)* of figure 1.6 shows a picture of the first generation DEEP prototype installed at JSC. The front rack (with blue cables) is the Booster. The one in the back (with screens) is the Cluster. Between them a rack of air-cooled power supplies is located.

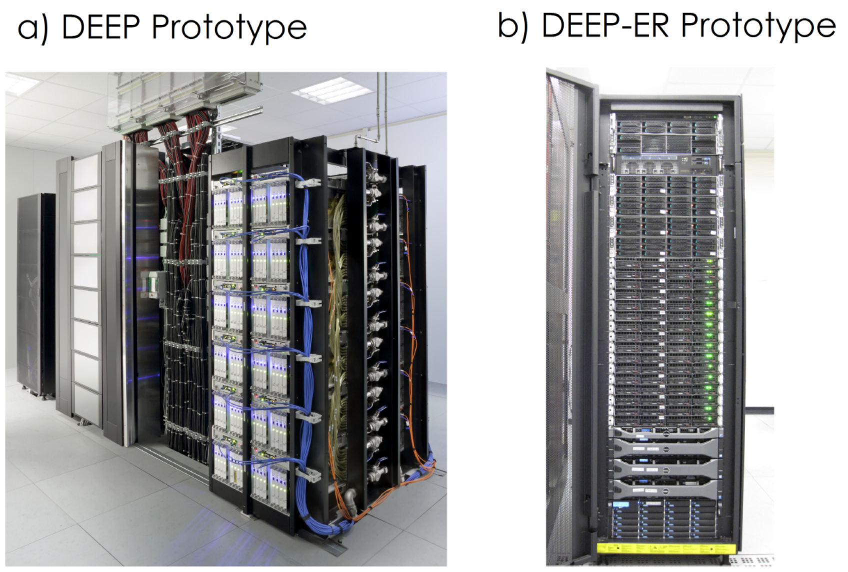


FIGURE 1.6: Pictures of the DEEP prototypes. *a)* First generation(DEEP); *b)* Second generation (DEEP-ER).

DEEP Cluster The DEEP Cluster, installed in 2012, is an off-the-shelf Aurora Cluster manufactured by the company Eurotech. The Aurora line comes with double-sided racks (H 2260 mm × W 1095 mm × D 1500 mm), hosting up to 16 chassis. These contain a number of direct liquid-cooled node-blades. In the case of the DEEP Cluster, only 8 chassis were installed, each containing 16 Intel Xeon (Sandy Bridge) direct-liquid cooled node-blades. The 128 blades, equipped with two Intel Xeon E5 2680 processors each, are interconnected by a Mellanox InfiniBand ConnectX fabric using QDR-generation technology.

DEEP Booster The DEEP Booster is based on custom hardware, designed and built entirely within the DEEP research project. The first step for building a Booster, i.e. a cluster of accelerators, is to select a *stand-alone accelerator* processor. End of 2011, when the DEEP project started, there was no such device on the market. All existent devices at that time (GPUs and many-core processors) required a host processor both for booting and for managing communication through a standard high-speed network, making them *slaves* of a host CPU.

The DEEP team found a solution in using the Intel Xeon Phi (KNC) co-processor in combination with the EXTOLL [18] interconnect. The EXTOLL network is a switchless solution in which host interface, network interface controller, and router are completely integrated in one-chip. Each chip gives 7 interconnect links, what naturally allows bulding a 3-D torus using six of them. The 7^{th} link provides further connection possibilities (e.g. short-cuts in the 3-D torus) and has been employed in the DEEP Booster to attach the interface nodes. Further details of the EXTOLL design and capabilities are given in [16]. Each compute node in the DEEP Booster contains one KNC attached via PCIe to an Altera Stratix V FPGA running the EXTOLL network protocol. The FPGA implements EXTOLL's NIC and acts as the PCIe root port.

The physical connection between the Cluster and Booster sides of the system is done via interface nodes. Each of them contains a general purpose (Intel Xeon i7) server CPU, with an InfiniBand HCA and an EXTOLL NIC pluged to a PLX PCIe switch. The CPUs of the interface modules run the inter-module bridging protocol (see section 1.6.3) and play an initiator role in the remote-booting process of the Booster nodes. Without a host CPU, booting the KNC is achieved exploiting EXTOLL's ability to transparently forward PCIe packets [16].

The DEEP Booster rack is, as the Cluster, a double-sided, direct water-cooled Eurotech's Aurora rack. It holds 12 chassis, each containing 16 double-node blades (i.e. 32 individual nodes) and two interface nodes. Compute and interface blades are plugged into a passive backplane, through which the chassis-internal network links are routed. The inter-chassis links are implemented with copper cables. The Booster network topology is a (8×6×8) 3D torus. EXTOLL 7^{th} links connect inside each chassis the Booster compute nodes with the interface nodes, and through them with the Cluster part of the Modular Supercomputer.

GreenICE Booster The ASIC version of EXTOLL (code-named Tourmalet) was available too late for its integration in the large DEEP Booster, so that FGPAs had to be employed to implement the EXTOLL network protocol. The main difference between the FGPA and the ASIC versions is that the latter achieves about 7× higher link bandwidth and 8× lower latency. To evaluate the use of this high-speed network in a Booster-like system, a small prototype – called *GreenICE Booster* – was built in the last months of the DEEP project [7]. It differs from the large DEEP Booster previously described not only in its network implementation, but also in its overall physical integration and cooling.

Each GreenICE Booster node consists of a KNC (Intel Xeon Phi 7120D) and an EXTOLL network interface card (NIC) (a Tourmalet PCIe card [19] in which the EXTOLL ASIC is embedded). The EXTOLL NICs are interconnected to each other via copper cables that realize a 3-D torus topology. Groups of 8 KNC and 8 Tourmalet cards share the same backplane, which is mainly responsible for conducting the PCIe signals between KNC and NIC and for providing the necessary electrical power to both of them. A GreenICE chassis includes 4 such dense backplanes, i.e. a total of 32 Booster nodes.

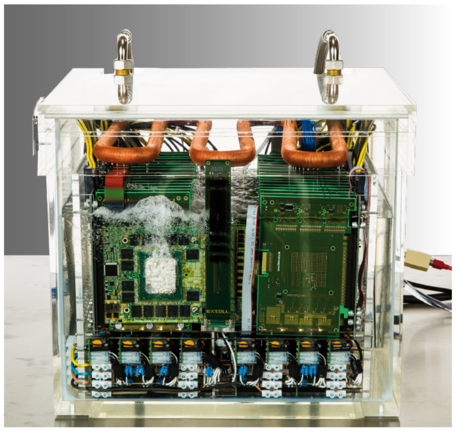


FIGURE 1.7: Picture of the GreenICE Booster chassis (outside its rack).

The GreenICE chassis is built as an hermetically closed container, filled with the Novec [1] cooling liquid produced by 3M. All KNC and Tourmalet cards together with the backplanes and cables are immersed in this fluid, which dissipates the heat produced during operation. The Novec liquid boils at 49° C. In direct contact with the hot electronic components, Novec evaporates. Its gas bubbles – less dense than the liquid form of Novec – transport the absorbed heat to the upper part of the chassis basin, where they get in contact with a water cooling serpentine. There Novec condensates and the liquid droplets fall down to the lower part of the basin, closing the cooling cycle. Figure 1.7 shows a picture of the GreenICE Booster chassis deployed in the DEEP project.

Interface nodes connect the GreenICE Booster with the DEEP Cluster[3], and are implemented as standard air-cooled Xeon servers (equipped with a Tourmalet card and a Mellanox FDR InfiniBand adapter) sitting in the upper part of the rack. Their role is the same as in the large Booster: remote-booting the KNCs and driving the communication between the DEEP Cluster and the GreenICE Booster.

[3]There is no direct connection between the large Booster and the GreenICE Booster, as none of the co-design applications intended to distribute code over the two Boosters.

1.5.2 Second generation (DEEP-ER) prototype

The DEEP-ER prototype consists of one Cluster and one Booster modules integrated in a single, standard 19" rack (see Part *b)* of figure 1.6), which also holds the storage system (one meta-data, two storage servers and 57 TB storage in spinning disks). A uniform high-speed EXTOLL interconnect runs across Cluster and Booster, connecting them internally, between each other, and with the storage. The Tourmalet A3 product generation has been selected. It relies on an ASIC-based NIC that provides six[4] links of 100 Gbit/s bandwidth each, and contains logic for highly efficient message transmission and DMA.

The DEEP-ER system is smaller than its predecessor (all together 24 nodes with a total 36 TFlop/s peak performance) but is enhanced with advanced memory technologies, in particular non-volatile and network attached memories (NAM). With them, a multi-level memory hierarchy has been built, which provides scalable I/O performance and enables the implementation of innovative I/O and resiliency techniques (see 1.7.3). The total memory capacity of the system is 8 TByte.

Given the size of the system and the strong focus of the DEEP-ER project on software development, the construction of the prototype was kept as simple as possible, employing off-the-shelf, air-cooled hardware components. A climate machine in the computer room, provides the needed cooling capacity for the prototype.

DEEP-ER Cluster The DEEP-ER Cluster is composed of 16 SuperMicro servers (1U high ×19" wide). Each server hosts an Intel Xeon (Haswell) processor, one EXTOLL Tourmalet card, and and a non-volatile memory device (see 1.5.2). The latter are connected to the processor board via PCIe gen3. Copper cables running between the Tourmalet cards build-up a 4×2×4 torus, with links going towards the storage servers, the Booster part, and the NAMs.

DEEP-ER Booster The DEEP-ER prototype profited from the evolution of the Intel Xeon Phi processor line. The second generation of this product, code-named *Knights Landing* (KNL) is a self-booting many-core processor, what greatly eases the construction of a Booster. A further important feature for DEEP-ER is KNL's larger memory capacity, with high bandwidth 16 GB MCDRAM, and the possibility to additionally plug 6 memory DIMMs. In the prototype these were populated with a total of 96 GB DDR4 memory, fitting the requirements established by the co-design applications.

The 8 servers that constitute the Booster part of the system are of the model code-named *(*Adams Pass), from which four of them fit into a 2U server. However, in our case only the two lower serves could populate each chassis. The reason is that the NVMe cards had to be located on the upper

[4]The 7th link, though available also in the Tourmalet ASIC, is not easily accessible in the current PCIe board and has therefore not been used in the DEEP-ER Booster.

slot (connected via a flat PCIe cable), since there was no enough space within the server enclosure to fix both the NVMe device and the EXTOLL Tourmalet. The network topology chosen for the Booster is a (2×2×2) torus.

Memory technologies All nodes in the DEEP-ER prototype (in both Cluster and Booster) feature a **non-volatile memory (NVM)** device for efficiently buffering I/O and storing checkpoints. The chosen technology is Intel's DC P3700, an SSD replacement device with 400 GByte capacity that provides high speed, non-volatile local memory, attached to the node with 4 lanes of PCIe. Extensive experiments and a wide range of measurements with I/O benchmarks and application mock-ups have been performed, which show substantial performance increases over conventional best-of-breed SSDs, in particular for scenarios with many parallel I/O requests, and over state-of-the art I/O servers.

DEEP-ER has also introduced an innovative memory concept: the **network attached memory (NAM)**. It exploits the remote DMA capabilities of the EXTOLL fabric, which enable remotely accessing memory resources without the intervention of an active component (such a CPU). Packed into a PCIe add-on card, the NAM combines Hybrid Memory Cube (HMC) devices with a state-of-the-art Xilinx Virtex 7 FPGA to create a high-speed memory device that is directly attached to the EXTOLL fabric, and is therefore globally accessible by all nodes in the system. The FPGA implements three functions: the HMC controller, the EXTOLL network interface with two full-speed Tourmalet links, and the NAM logic. The HMC controller has been developed by UHEI and its design has been released as Open Source [28].

A *libNAM* library has been implemented to give system and application software running on the DEEP-ER prototype access to the NAM memory pool, and to enable the execution of any pre-defined functions in the NAM logic. As a first NAM use-case, a checkpointing/restart function has been implemented. It uses the NAM FPGA to pull the required data from the compute nodes and locally calculate the parity information.

The DEEP-ER prototype holds two NAM devices, each with 2 GByte capacity. Their small size is due to limitations of current HMC technology. Future implementations can, however, increase capacities and may trigger a rethinking of memory architectures for HPC and data analytics. In fact, the NAM-concept is going to be further developed within the successor project DEEP-EST.

1.5.3 JURECA

The JURECA Cluster, running since 2015 as JSC's general purpose production cluster, has received a Booster module that is attached to it since end of 2017.

JURECA Cluster The JURECA Cluster [34] (see Part a) in figure 1.8) consist of 1872 compute nodes, accompanied by additional 24 nodes used for login, visualization, etc. Each node is equipped with two 12-core Intel Xeon E5-2680 v3 processors clocked at 2.5 GHz. The compute nodes present various memory sizes and a maximum memory bandwidth of 136 GB/s. Most of the nodes (1,680) have 128 GB of DDR4 memory. From the remaining nodes, 128 contain 256 GB and 64 have even 512 GB. Furthermore, 75 nodes are accelerated with two nVIDIA K80 graphic processing units (GPGPU), each.

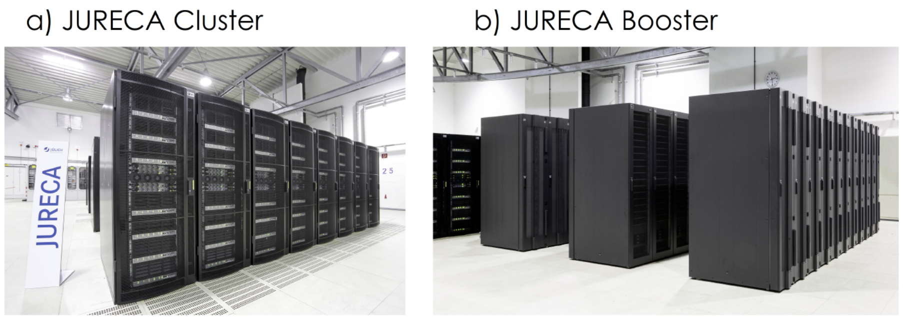


FIGURE 1.8: JURECA system, composed of interconnected Cluster and Booster

The interconnect is Mellanox EDR (extended data rate) InfiniBand (100 Gb/s link bandwidth and 1μs MPI latency). Each node hosts a ConnectX-4 HCA, attached to a first layer of 36-port switches. The spine of the fabric is build out of three 648-port Omni-Path director switches, leading to a full fat-tree topology without any pruning. With it, full bisection bandwidth and non-blocking communication for appropriate communication patterns is achieved.

In order to provide I/O capabilities to JURECA's users it is connected to JSC's central GPFS storage cluster JUST via a set of Mellanox gateway routers bridging between the internal EDR InfiniBand fabric and the facility's Ethernet backbone using 40 G technology.

JURECA Cluster system usage Figure 1.9 shows how the JURECA Cluster reaches nearly 90% productive usage. The first four months of operation (July to October 2015) correspond to the first phase of the installation, in which only 4 of the Cluster racks were available to users. The Cluster started full operation in November 2015. The down-time in December 2016 was due to problems with the GPFS file-system, caused by a software bug.

The different colors in figure 1.9 show the distribution of the compute time over the various pools of users. Regional users are grouped under *FZJ* and *JARA*; German-wide users belong to *NIC*; national and international users working closely together with JSC through specific research collaborations are accounted under *FZJ-obligations*.

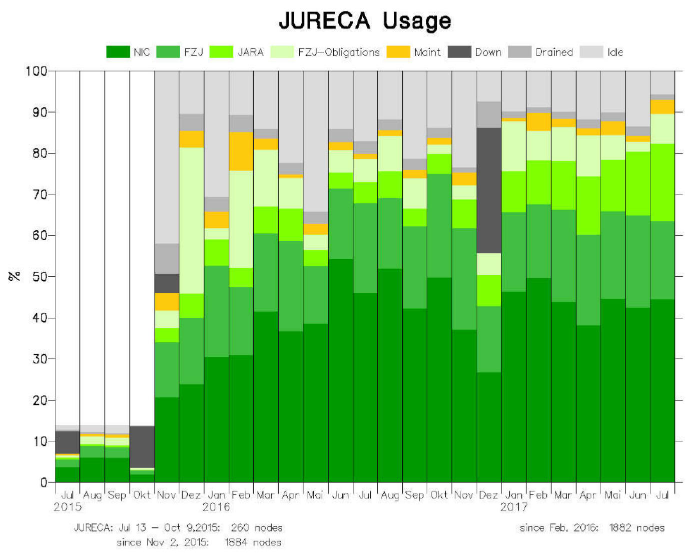


FIGURE 1.9: Usage of the JURECA Cluster.

JURECA Booster Recently JSC has extended the JURECA system by a Booster based on Intel's Xeon Phi processors of the KNL generation (see Part b) in figure **??**). For this, a total of 1,640 Booster nodes each one hosting a 68-core Intel XeonPhi 7250-F many-core CPU are utilized. Each processor contains 16 GB of high-bandwidth MCDRAM and is accompanied by 96 GB of DDR4 memory. The -F version of KNL has Intel's Omni-Path fabric on board. The Booster nodes form their own full fat-tree fabric, which allows high flexibility in the communication patterns of the highly scalable parts of the applications to run on the Booster. The fat-tree topology is set-up with a first layer of 48-port switches, accompanied by a spine-layer build out of Omni-Path director switches.

In order to embed the JURECA Booster in the existing environment, two types of gateway nodes are included in the system. On the one hand 198 MPI Router nodes attach the Booster to the JURECA Cluster bridging between their respective Omni-Path and InfiniBand fabrics. Each of the MPI Router nodes hosts a Mellanox ConnectX-4 Host Channel Adapter and a Intel Omni-Path Host Fabric Adapter. The nodes are powered by a single 14-core Intel Xeon E5-2690 v4 processor, which translates between the two fabric protocols and hosts 64 GB of DDR4 memory. On the other hand, 26 GPFS router nodes bridge to the facility Ethernet backbone using 40 G Ethernet. These nodes are populated with two 14-core Intel Xeon E5-2690 v4 processors each. Besides 32 GB of DDR4 memory, each of the GPFS router nodes host a Intel Omni-Path Host Fabric Adapter and a Intel XL710 Dual Port 40 G NIC.

1.6 System Software

The software environment used in the DEEP prototypes and the one installed in JURECA have many points in common. The few differences between them come mostly from the more conservative philosophy inherent to a production system as JURECA, compared to the more experimental character of the DEEP research prototypes (newer, higher-risk approaches can be tried out when full-time operation is not a requirement). The most important software components are summarised in table 1.2.

1.6.1 System administration

The cluster management software running on the general-purpose clusters at JSC is ParaStation V5 [9], an Open Source middleware developed and maintained by the ParaStation consortium, in which JSC is one of the key members. ParaStation contains multiple components covering the most important functionalities for management of large-scale, high-performance computer, These range from low level (e.g. process management for parallel and serial jobs, support for various high-speed network, hardware monitoring, resource management), to the higher layers of the programming environment (e.g. MPI, high-performance parallel file copy).

Its MPI programming environment (ParaStation MPI) is based on MPIch and relies on its own communication library, named *pscom*. It is highly scalable and extendible through a plugin system and supports many different interconnects without application re-linking. The best communication path between each two nodes is chosen automatically at application start-up. Within the DEEP projects *pscom* has been extended with new plugins for EXTOLL and the EXTOLL-InfiniBand network-bridging protocol (see section 1.6.3. This way, ParaStation MPI covers all possible communication paths within the DEEP prototypes. For JURECA, new *pscom* plugins for the Booster's Omni-Path network and the Omni-Path to InfiniBand bridging protocol have been recently implemented.

1.6.2 Schedulers and resource management

One of the main differences between DEEP's and JURECA's software environments is the utilization of different packages for resource management and scheduling. JURECA employs SLURM [49], while the Torque [3] and

TABLE 1.2: Software configuration

Feature	**DEEP prototypes**	**JURECA**
Login Node OS	Linux CentOS 6.9	Linux CentOS 7
Compute Node OS	Linux CentOS 7	Linux CentOS 7
Cluster Management	ParaStation ClusterSuite	ParaStation ClusterSuite
Parallel Filesystem	GPFS 4.2 (/home) BeeGFS 2015.03 (/work)	GPFS 4.2
Compilers	Intel 2017.2.17 Fortran, C/C++ Compiler	Intel 2017.2.17 Fortran, C/C++ Compiler
	GNU 5.3 PGI 13.4	GNU 5.4.0 PGI 17.3
Parallel Programming	ParaStation MPI 5.1.9 OpenMP Intel MPI 5.1 OmpSs	ParaStation MPI 5.1.9 OpenMP Intel MPI 2017.2.174 MVAPICH2 (2.2-GDR) NVIDIA CUDA 8 OpenCL OpenAcc
Notable Libraries	Intel Math Kernel Library 2017.2.174	Intel Math Kernel Library 2017.2.174
	SIONlib 1.7.1 HDF5 1.8.18 netcdf 4.2 PETSc 3.7.4 SCR 1.1.8 Exascale10	SIONlib 1.7.1 HDF5 1.8.18 netcdf 4.4.1.1 PETSc 3.7.6 (and 3.7.5)
Job Scheduler Resource Manager	Moab / SLURM Torque/SLURM + ParaStation psmgmt	SLURM SLURM + ParaStation psmgmt
Debugging Tools	Intel Inspector TotalView	
Performance Tools	Scalasca 2.0b3 Intel Advisor 17.1 Intel Inspector 17.1 Intel VTune 17.1 Vampirtrace 5.14.3 Extrae/Paraver 2.5.1	Scalasca 2.3.1 Intel Advisor 17.2 Intel Inspector 17.3 Intel VTune 17.4 TotalView 2017.0.12 Allinea Performance Reports

Maui [2] combination is used in the DEEP and DEEP-ER prototypes. This is the result of a strategic decision taken at JSC shortly before the deployment of the JURECA Cluster. At that time, and to profit from SLURM's much wider use and active development in the HPC community, JSC decided to move from Torque-Maui/Torque-Moab into SLURM for all its production cluster systems. Sticking to the new policy, the DEEP prototypes originally installed with the old management software have been migrated to SLURM. The modifications to the resource management required to support a Modular Supercomputer, which had been implemented in Torque/Maui within the DEEP project, are currently being ported to SLURM.

Independently of the underlying software, in order to properly support a Modular Supercomputer the resource management system must be able to handle its heterogeneity. In particular it must be able, for any particular job, to simultaneously allocate nodes within several compute modules, while being aware of their different node configurations.

Figure 1.10 shows schematically three fictive application-workloads with different profiles running on a Modular Supercomputer composed of 5 compute modules.

- Workload 1: a *classic* HPC application with a large highly scalable code part running on the Booster and a smaller low scalable part running on the Cluster.
- Workload 2: a typical HPDA application, which runs mostly on the data-analytics module, using the Cluster module for pre-processing and the Graphics module for visualisation of the results.
- Workload 3: a neuroscience application starting on the Cluster module, running simulations on the Neuromorphic module, and using GPGPUs on the Graphics module to run arithmetic-intensive kernels.

The management and scheduling systems employ heuristics to determine the optimal distribution of user-codes on the system, and to order them in the queue. Goals of the optimisation algorithms are to give all applications their required resources, and globally achieve maximal system utilisation. From the system perspective the latter is more important, as it leads to a higher scientific throughput. Full system utilisation is possible with a good mix of diverse applications or, looking it from the opposite perspective, with a system design tailored to the specific application portfolio.

To explain how the users interact with the scheduler in a Modular Supercomputer, let's consider as an example Workload 1 in figure 1.10: an application that starts its execution on the Cluster part of a Modular Supercomputer and runs part of its code on the Booster module. In that case, the user places a request to the job scheduler via a modified version of the `qsub` command:

```
qsub -l nodes=cn:cluster+bn:booster "mpiexec -np n <exec>"
```

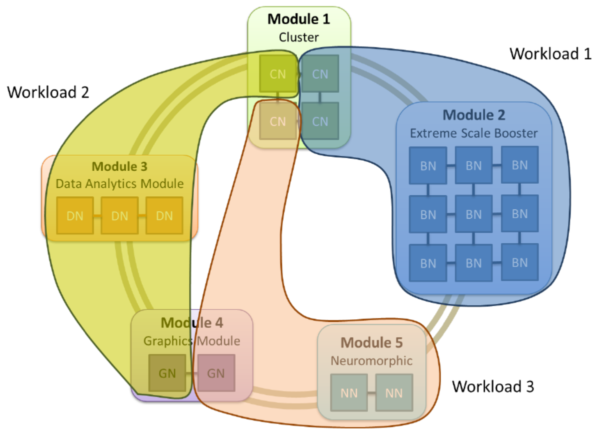


FIGURE 1.10: Three application workloads running on a Modular Supercomputer.

where *cn* and *bn* denote the number of Cluster and Booster nodes, respectively. Within the application, `MPI_Comm_spawn` calls are used to run their highly scalable code parts on the Booster, as described in section 1.7.1. The scheduler then passes lists of all the allocated resources to ParaStation's process management system.

Resource allocation may be *static* (with all needed nodes allocated before the job starts and remaining reserved until if is terminated), or *dynamic*. In the latter case, an application calling `MPI_Comm_spawn` may request more Booster nodes than initially allocated. When this happens, the MPI runtime forwards the request to the resource manager. Depending on the current load of the machine, it may block the application until enough nodes are available, or until an optionally specified timeout expires, in which case the call fails. When the resources are available, these are associated to the application, which starts running on the reserved nodes. If the allocation is *dynamic*, nodes are freed and allocated to other users as soon as the spawned tasks are done. In the *static* case all resources are freed once the full application run is completed.

1.6.3 Network-bridging protocol

In a Modular Supercomputer the interconnect does not necessarily need to be the same in all modules. In fact, the first DEEP prototype and JURECA do present different network technologies in their Cluster and Booster parts (see section 1.5).

For applications to run distributed on modules with different interconnects, a very efficient network bridge is needed. In DEEP, a so-called *Cluster-Booster protocol* was written to bridge between InfiniBand (FDR) and EXTOLL. Invisible to the user, this bridging protocol is called by the MPI library when data has to be transfered between application parts running on separate modules. DEEP's Cluster-Booster protocol provided significant performance benefits compared to a standard store-and-forward implementation. A detailed discussion of the design and implementation of this protocol, and benchmarks of the results obtained can be found in [14].

For the JURECA system an own bridge-protocol is being developed at JSC, together with its partners ParTec and Intel. It will run on JURECA's interface nodes and seamlessly connect the InfiniBand EDR network of the JURECA Cluster, with the Omni-Path fabric of its Booster.

1.6.4 I/O software and file system

Some of the first DEEP users found the memory and storage capacity of the first generation DEEP prototype was too small, specially on the Booster side of the machine. Therefore, the second generation prototype (DEEP-ER) introduced a memory hierarchy (see 1.5.2) as basis to an scalable I/O infrastructure. The resulting I/O software platform combines the parallel I/O library SIONlib with the BeeGFS parallel file system. Together, they enable the efficient and transparent use of the underlying hardware and provide the functionality and performance required by data-intensive applications and multi-level checkpointing-restart techniques.

The I/O library SIONlib [23, 24] acts as a concentration-layer for applications to most efficiently use the underlying file system. SIONlib compacts all the data that applications performing task-local I/O need to store into one or very few large files, easily manageable by the file system. SIONlib runs on JURECA and the DEEP prototypes. In particular, it plays a key role on the second generation (DEEP-ER) system, where it builds a bridge between the I/O and resiliency components of the software stack. SIONlib is used to copy local checkpoints into the NVM of a companion (*buddy*) node for redundancy, and to efficiently store checkpoint-data in the global file system. Both functions work in combination with the scalable checkpointing library SCR (see section 1.7.3).

The file system on JURECA is GPFS. Though the DEEP prototypes are also connected to this central JSC infrastructure (used for the `/home` directories), their effective (`/work`) global parallel file system is BeeGFS [21]. BeeGFS provides a solid, common basis for high-performance, parallel I/O operations. Advanced functionalities, such as a local cache layer in the file system, have been added within the DEEP-ER project. The cache domain – based on BeeGFS on demand (BeeOND) [22] – stores data in fast node-local non-volatile memory devices and can be used in a synchronous or asynchronous mode. This speeds up the applications' I/O operations and reduces the frequency of accesses to the global storage, increasing the overall scalability of the file system.

1.7 Programming model

An important amount of the work performed in the DEEP projects was related to the development of a programming environment that maximally reduces the effort of porting applications to the new platform. The components of the programming model were consciously selected to be the de-facto standard in HPC: MPI+OpenMP. Adjustments and extensions have been implemented on their lower layers to properly support the Modular Supercomputing architecture.

This approach aims at hiding the hardware complexity from the end-user. Runtime environments supporting the hardware features of each module have been implemented. For example, the MPI library running on each module is specifically optimized for the network fabric installed on it, and it automatically calls the low-level network bridging protocol whenever an application communicates between two modules with different fabrics. Admittedly, in such a case the interface nodes increase the communication latency. The impact of this restriction is however minimal for the performance of the overall application, since the programming model for a Modular Supercomputer foresees partitioning applications at boundaries involving low-frequency communication and limited data volume. This is a consequence of the fact that large portions of the application – comprising internal inter-node communication – run on each compute module, so that collective operations happen mostly within the modules, and not between them. Therefore, in this scenario the term *kernel*, which is commonly used to describe parts of codes running on accelerators, is not appropriate. On a Modular Supercomputer the term *part* has been chosen to describe the portion of an application offloaded from one compute module to another.

1.7.1 Inter-module MPI Offloading

The actual offloading process between modules is kept as close to existing standards as possible. The dynamic process model of MPI-2, namely `MPI_Comm_spawn`, has been chosen to perform the offloading mechanism (see figure 1.11). It allows spawning processes from one module to another and provides an efficient way of exchanging data between them using MPI semantics.

`MPI_Comm_spawn` is a collective operation performed by a subset of the processes of an application starting on a specific module. Its call requires as input the binary-name and the number of new processes to be started. A new inter-communicator is returned, providing a connection handle to the children. Each child calls then `MPI_Init`, as usual, and gets access to the inter-communicator via `MPI_Get_parent`. Both parts of the applications – the part containing the `main()` function, and the offloaded part – have their own `MPI_COMM_WORLD`s

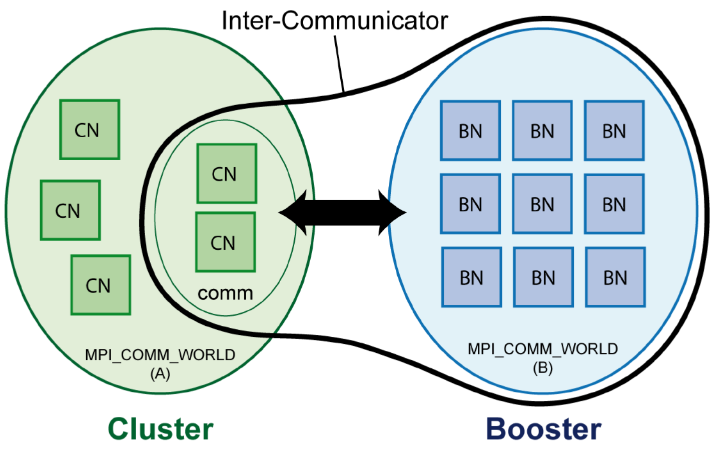


FIGURE 1.11: `MPI_Comm_spawn` schematics, describing the example of an application starting on the Cluster and offloading a part of its code to the Booster. (*CN*: MPI process on Cluster node; *BN*: MPI process on Booster node)

providing full MPI functionality on either side, and are connected to each other via inter-communicators.

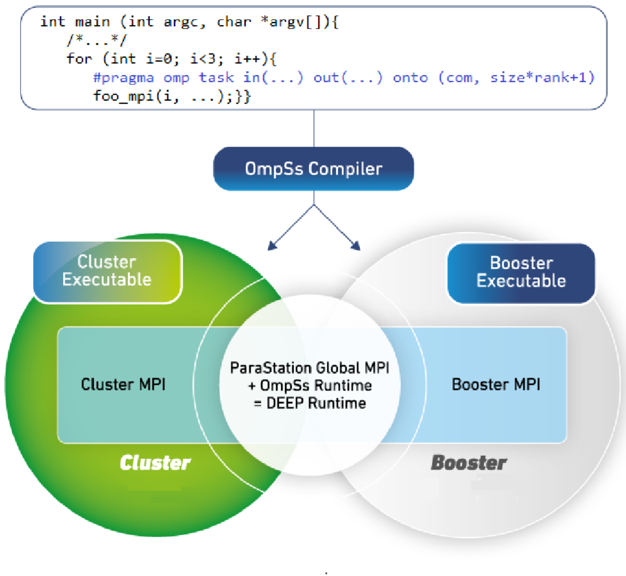
The intra-module MPIs, together with the offloading mechanism and the network-bridging protocol constitute a *global MPI*, i.e. an MPI implementation that is usable on all node types and allows for communication between nodes sitting on different modules. Its implementation uses ParTec's ParaStation MPI (see section 1.6.1), which runs on all Modular Supercomputers at JSC.

The code changes needed to map an application (xPic) onto the two parts of the DEEP-ER prototype using ParaStation global MPI, and results demonstrating the gained performance gain are discussed in [35].

1.7.2 OmpSs abstraction layer

For a programmer, actively employing the `MPI_Comm_spawn` primitive means coordinating and managing two or more sets of parallel MPI processes, explicitly sending the required data from one side to the other. This approach may become cumbersome for large and complex applications. To reduce this porting effort, an abstraction layer sitting on top of the global MPI has been implemented within the DEEP project. It enables application developers offloading large complex tasks by simply annotating with *pragmas* the parts of their codes that shall run in a different compute module to where the `main()` function is located.

The abstraction layer is based on the OmpSs data-flow programming



```
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, ...);}}
```

FIGURE 1.12: Workflow of an application running across the Cluster and Booster modules with the OmpSs abstraction layer.

model [13, 42]. This OpenMP 4.0-like environment exploits task-level parallelism and supports asynchronicity, heterogeneity and data movement. Using OmpSs, an application's code is annotated with OpenMP-like pragmas that indicate data dependencies between the different tasks of the program. Taking these dependencies into account, the OmpSs runtime decides on the order of the tasks and whether concurrent execution is allowed, creating a task dependency graph at run-time. All this information is used to schedule the tasks on the available devices.

Figure 1.12 shows the workflow of an application that starts on the Cluster module and sends a part of its code (an OmpSs task) to the Booster. For this purpose, a single OmpSs pragma is included in the source code. It indicates the data that the task needs as input, the data that it generates as output, and the destination to where it shall be sent, with a given communicator and size. The OmpSs source-to-source compiler interprets this pragma and generates two executables: one for the part of the code to run on the Cluster, and one for the part that shall run on the Booster. The OmpSs runtime cooperates then with the global MPI layer to transparently manage all data transfers between the MPI processes running on the two modules.

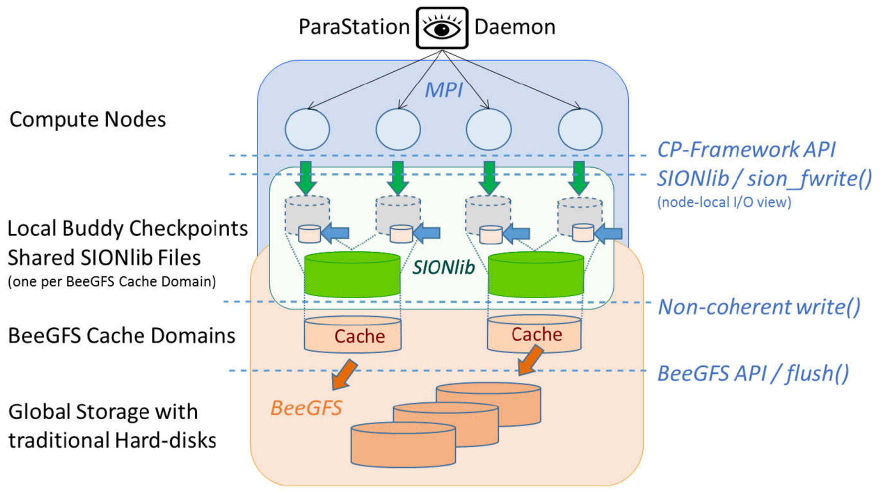


FIGURE 1.13: Buddy-checkpointing software stack in the DEEP-ER prototype.

1.7.3 Resiliency software

Within the DEEP-ER project OmpSs has been further extended to improve application's resiliency against transient hardware failures. Failed OmpSs tasks can now be restarted at their last checkpoint, without losing the work that had been performed in parallel by other OmpSs tasks of the same code. To support this task-based resiliency functionality, the ParaStation management daemon has been extended with an interface for querying resiliency-related status information from the MPI layer and thus also from the OmpSs runtime environment. ParaStation MPI itself is now able to detect, isolate and clean up failures of MPI-offloaded tasks, which can be then independently restarted without requiring a full application recovery.

Complementing the task-based resiliency mechanism, traditional user-level application checkpoint-restart is applied. The Scalable Checkpoint-Restart library (SCR) offers a flexible interface for applications to perform checkpoints and restart from them in case of failure. The user only needs to call the library and indicate the data required by the application to restart execution. SCR decides based on a failure model (specifically developed for the DEEP-ER prototype) where and how often checkpoints are performed, and keeps a database of checkpoints and their locations in preparation for eventual restarts.

Results obtained by applicatiosn showing the benefit of applying the I/O and resiliency features developed in DEEP-ER are discussed in [36].

A *buddy-checkpointing* functionality has been implemented in the DEEP projects – combining SCR, ParaStation MPI, SIONlib, and BeeGFS – to save data on the node-local storage and keep a copy on a companion node (see figure 1.13). In case of hardware failure, the data lost on the damaged node can be recovered from its *buddy* and the application can continue with minimum time-loss. SCR keeps track of the association between host nodes and buddies; SIONlib takes care that all MPI processes running on a single node jointly write their checkpoint-date into a single file in the buddy-node; and BeeOND saves the data itself on the cache-file system on the local NVM, transferring it asynchronously to the permanent global storage.

Last, but not least, SCR and SIONlib have been extended to enable the use of the DEEP-ER network attached memory (NAM) pool. Together, they call libNAM to trigger parity computation on the NAM. This enables application developers to transparently perform checkpointing/restart to/from the NAM without modifying their codes.

1.8 Cooling and facility infrastructure

The DEEP prototypes and the JURECA system are manufactured by different hardware providers with diverse integration approaches, what translates into the use of different cooling technologies.

- DEEP prototype (first generation): Eurotech's Cluster and Booster modules use direct water cooling for all nodes. The small GreenICE Booster uses two-phase immersion cooling to extract heat from electronics, plus a water cooling loop to condensate the immersion-coolant. Power supplies, login nodes and external storage are air cooled.
- DEEP-ER prototype (second generation): completely air-cooled.
- JURECA Cluster and Booster: air cooling with water-cooled rack doors.

Physically, the DEEP and JURECA systems are located in two different computer rooms at JSC, which present also quite different infrastructures. JURECA sits in the same large hall than JUQUEEN and JUST, while the DEEP prototypes are hosted in a smaller room specially reserved for experimental systems. Since the cooling technology and room infrastructure in the production hall is rather standard, here only the free cooling infrastructure of the DEEP room is described.

Free cooling in the DEEP computer room One of the objectives of the DEEP projects is to investigate means to keep the power consumption of HPC systems at bay. In this context, the first generation DEEP prototype was designed to employ free cooling.

The use of dry coolers instead of chillers to keep the cooling liquid of a system at an acceptable temperature (up to 40° C) saves a significant amount of power. However, the use of warm-coolant also poses some challenges in

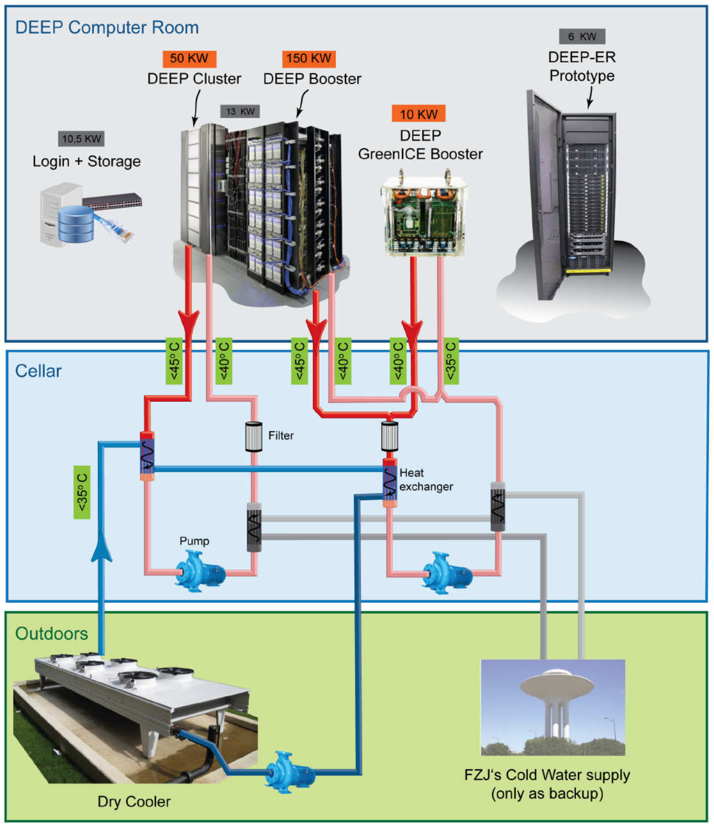


FIGURE 1.14: Schematics of cooling infrastructure for the DEEP prototypes. The DEEP prototype (Cluster and Booster) use direct water cooling, as well as the DEEP GreenICE Booster. Login nodes and the second generation prototype (DEEP-ER) employ air-cooling.

system integration and maintenance. To efficiently cool a high performance computer at these temperatures, the chosen coolant has to be very effective in removing and transporting the heat away from the electronics. Direct water cooling was the solution chosen for the first DEEP prototype. Aluminium cold-plates were designed with a *Manhattan* profile exactly matching the node boards. These cold plates are plugged via quick-disconnects into chassis-level water distribution bars, with themselves are connected via tubes to the room infrastructure pipes.

Figure 1.14 depicts a simplified scheme of the water cooling infrastructure implemented for the DEEP prototypes. The Cluster and Booster modules of the first DEEP generation are both water-cooled systems. The internal (primary) cooling loop of each part of the system runs from the DEEP computer room down to the cellar of the building, where pumps and heat exchangers are located. One of the heat exchangers puts the primary cooling loop in contact with the secondary loop. This runs from the cellar to the outer part of the building, where a dry cooler is located. The dry cooler is ultimately responsible for keeping the temperature of the secondary loop below 35° C, what guarantees an inlet temperature in the primary loops below 40° C. A second heat exchanger brings the primary loop in contact with the central cool water distribution of Forschungszentrum Jülich. This connection is foreseen only as backup, for the very rare occasions (the few hottest days in the summer) in which the outside air temperature becomes too high to operate the system using only free cooling.

The primary cooling loop of the DEEP Booster is split into two branches: one for the large Booster, and one for the small GreenICE Booster. Valves and control sensors at various parts of the infrastructure guarantee that the coolant is kept at the desired temperature and pressure at each point of the loop. Additionally, probes of the primary loop coolants are taken regularly to analyse their chemical composition. Conductivity and pH are particularly carefully controlled.

1.9 Conclusions and next steps

The Modular Supercomputing architecture is the result of more than a decade experience gathered at the Jülich Supercomputing Centre (JSC) in the co-development, operation, and maintenance of high performance computers. The motivation behind this new approach to heterogeneous computing is to enable a most efficient use of the computing resources, while providing application developers with all necessary tools to take the step from Petascale to Exascale computing.

JSC recognized very early that no single technology could ever satisfactorily fulfill the requirements of all its diverse user communities. Specially, if the resulting HPC system should become both user-friendly and energy efficient. Therefore, JSC initiated a dual supercomputing approach by simultaneously operating two production machines: a general purpose cluster and a massively parallel computer. Users were channelled to the system best suiting their needs: low/medium-scalable with high data-management to the cluster, and high-scalable to the massively parallel. However, an intermediate type of applications was identified, which could profit from some characteristics of both architecture branches. This lead JSC to create the Cluster-Booster architecture, which basically brings the dual supercomputing approach into one single platform. This concept has been implemented already in two prototype systems (DEEP and DEEP-ER), and is currently going into production with JURECA.

The Modular Supercomputing architecture generalizes the idea of segregating heterogeneous resources into individual, interconnected compute modules. Advantages are expected – as demonstrated with the Cluster-Booster approach – from the high flexibility that the concept offers to users, its perfect match for very diverse application requirements [35], and the possibility to naturally integrate exotic technologies not yet widely available in the HPC environment.

The DEEP-EST research project will create a first incarnation of the Modular Supercomputer architecture and demonstrate its benefits. The exact configuration of the DEEP-EST prototype, which shall be installed in 2019, will be determined in close co-design between applications, system software and system component architects. It will include three modules: a general purpose Cluster module and an extreme scale Booster (together supporting the full range of HPCs applications), and a Data Analytics module (specifically designed for high-performance data analytics (HPDA) workloads). Goal is to cover the needs of both HPC and HPDA user communities, contributing to the convergence of both worlds. It shall also allow methodologies applied on HPC and HPDA to be combined for solving increasingly complex simulation and data analysis scenarios. Proven programming models and APIs from HPC (MPI and OpenMP/OmpSs) and from HPDA will be extended. Added to a significantly enhanced resource management and scheduling system, the resulting software stack shall enable straightforward use of the new architecture and achieve highest system utilisation and performance. Six ambitious, full-fledged applications from HPC and HPDA domains will drive the co-design, serving to evaluate the DEEP-EST prototype and demonstrate the benefits of its innovative Modular Supercomputer architecture.

Already now JSC is preparing the first step towards the implementation of the concept at large scale. In fact, the JUWELS system will be a Modular Supercomputer. Its first module, a general purpose Cluster, will be installed in Q2/2018. In the next few years, it will be complemented with further modules, with hardware and software designs tailored to JSC's user's needs.

1.10 Acknowledgments

The authors would like to thank the JSC staff and all people and partners involved in the DEEP projects, for their engagement and strong commitment, which led to the results described in this chapter. Specially, authors thank Dr. D. Krause for discussions proof-reading the JURECA sections, Dr. W. Frings, U. Frings and Dr. F. Janetzko for the information on JURECA and JUQUEEN user profiles included in figures 1.4 and 1.9, and A. Kreuzer and J. Kreutz for proof-reading the DEEP prototype's data in tables 1.1 and 1.2.

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) and 610476 (DEEP-ER), and from the Horizon 2020 Programme (H2020-FETHPC) under Grant Agreement n° 754304 (DEEP-EST). The present publication reflects only the authors' views. The European Commission is not liable for any use that might be made of the information contained therein.

Bibliography

[1] 3M. Novec. http://multimedia.3m.com/mws/media/569865O/3mtm-novectm-649-engineered-fluid.pdf?&fn=Novec649_6003926.pdf.

[2] Adaptive Computing. Maui. http://www.adaptivecomputing.com/products/open-source/maui/.

[3] Adaptive Computing. TORQUE Resource Manager. http://www.adaptivecomputing.com/products/open-source/torque/.

[4] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006.

[5] Norbert Attig, Florian Berberich, Ulrich Detert, Norbert Eicker, Thomas Eickermann, Paul Gibbon, Wolfgang Gürich, Wilhelm Homberg, Antonia Illich, Sebastian Rinke, Michael Stephan, Klaus Wolkersdorfer, and Thomas Lippert. Entering the Petaflop-Era - New Developments in Supercomputing. In *NIC Symposium 2010 / ed.: G. Münster, D. Wolf, M. Kremer, Jülich, Forschungszentrum Jülich, IAS Series Vol. 3. - 978-3-89336-606-4. - S. 1 - 12*, 2010. Record converted from VDB: 12.11.2012.

[6] Dirk Brömmel, Ulrich Detert, Stephan Graf, Thomas Lippert, Boris Orth, Dirk Pleiter, Michael Stephan, and Estela Suarez. Paving the Road towards Pre-Exascale Supercomputing. In *NIC Symposium 2014 - Proceedings*, volume 47 of *NIC Series*, pages 1–14, Jülich, Feb 2014. NIC Symposium 2014, Jülich (Germany), 12 Feb 2014 - 13 Feb 2014, John von Neumann Institute for Computing.

[7] Ulrich Bruening, Mondrian Nuessle, Dirk Frey, and Hans-Christian Hoppe. *An Immersive Cooled Implementation of a DEEP Booster*, pages 30–36. Intel Corporation, Munich, 2015.

[8] BSC. Reverse time migration (rtm) code website. http://www.cgg.com/en/What-We-Do/Subsurface-Imaging/Migration/Reverse-Time-Migration. Accessed: 2017-07-28.

[9] Jülich Supercomputing Centre. JUST: Jülich Storage Cluster website. http://www.fz-juelich.de/ias/jsc/EN/Expertise/Datamanagement/OnlineStorage/JUST/JUST_node.html. Accessed: 2017-07-16.

[10] ParTec Cluster Competence Centre. ParaStation V5 website. http://www.par-tec.com/products/parastationv5.html. Accessed: 2017-07-26.

[11] CERFACS. AVBP website at CERFACS. http://www.cerfacs.fr/avbp7x/. Accessed: 2017-07-28.

[12] CERN. CMS software github repository. https://github.com/cms-sw/cmssw. Accessed: 2017-07-28.

[13] Michalis Christou, Theodoros Christoudias, Julian Morillo, Damian Alvarez, and Hendrik Merx. Earth system modelling on system-level heterogeneous architectures: EMAC (version 2.42) on the Dynamical Exascale Entry Platform (DEEP). *Geoscientific model development*, 9(9):3483 – 3491, 2016.

[14] Alejandro Duran, Eduard Ayguadé, Rosa M. Badia, Jesús Labarta, Luis Martinell, Xavier Martorell, and Judit Planas. OmpSs: A proposal for programming heterogeneous multi-core architectures. *Parallel Processing Letters*, 21(02):173–193, 2011.

[15] Norbert Eicker, Andreas Galonska, Jens Hauke, and Mondrian Nüssle. *Bridging the DEEP Gap - Implementation of an Efficient Forwarding Protocol*, pages 34–41. Intel Corporation, Munich, 2014.

[16] Norbert Eicker and Thomas Lippert. An accelerated Cluster-Architecture for the Exascale. In *PARS '11, PARS-Mitteilungen, Mitteilungen - Gesellschaft für Informatik e.V., Parallel-Algorithmen und Rechnerstrukturen, ISSN 0177-0454, Nr. 28, Oktober 2011 (Workshop 2011), 110 - 119*, 2011. Record converted from VDB: 12.11.2012.

[17] Norbert Eicker, Thomas Lippert, Thomas Moschny, and Estela Suarez. The DEEP Project - An alternative approach to heterogeneous cluster-computing in the many-core era. *Concurrency and computation*, 28(8):23942411, 2016.

[18] Andrew Emerson and Fabio Affinito. Enabling a Quantum Monte Carlo application for the DEEP architecture. In *2015 International Conference on High Performance Computing Simulation (HPCS)*, pages 453–457, July 2015.

[19] EXTOLL. EXTOLL GmbH website. http://www.extoll.de. Accessed: 2017-07-23.

[20] EXTOLL. EXTOLL Tourmalet. http://www.http://extoll.de/products/tourmalet. Accessed: 2017-07-23.

[21] Partnership for Advanced Computing in Europe (PRACE). Prace website. http://www.prace-ri.eu. Accessed: 2017-08-07.

[22] Fraunhofer Gesselschaft. BeeGFS website.

[23] Fraunhofer Gesselschaft. BeeOND: BeeGFS On Demand website.

[24] Jens Freche, Wolfgang Frings, and Godehard Sutmann. High Throughput Parallel-I/O using SIONlib for Mesoscopic Particle Dynamics Simulations on Massively Parallel Computers. In *Parallel Computing: From Multicores and GPU's to Petascale, / ed.: B. Chapman, F. Desprez, G.R. Joubert, A. Lichnewsky, F. Peters and T. Priol, Amsterdam, IOS Press, 2010. Advances in Parallel Computing Volume 19. - 978-1-60750-529-7. - S. 371 - 378*, 2010. Record converted from VDB: 12.11.2012.

[25] Wolfgang Frings, Felix Wolf, and Ventsislav Petkov. Scalable Massively Parallel I/O to Task-Local Files. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, Portland, Oregon, November 14 - 20, 2009, SC'09, SESSION: Technical papers, Article No. 17, New York, ACM, 2009.ISBN 978-1-60558-744-8. - S. 1 - 11*, 2009. Record converted from VDB: 12.11.2012.

[26] Markus Götz, Christian Bodenstein, and Morris Riedel. HPDBSCAN - Highly parallel DBSCAN. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments - MLHPC '15*, page 2. Workshop Workshop on Machine Learning in High-Performance Computing Environments, subworkshop to Supercomputing 2015, Austin (Texas), 15 Nov 2015 - 15 Nov 2015, ACM Press New York, New York, USA, Nov 2015.

[27] GROMACS. GROMACS application website. http://www.gromacs.org/. Accessed: 2017-07-28.

[28] Human Brain Project (HBP). Human Brain Project pilot systems website. http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/HBPPilots/_node.html. Accessed: 2017-07-16.

[29] University Heidelberg. OpenHMC. http://www.uni-heidelberg.de/openhmc. Accessed: 2017-07-26.

[30] Helmoltz Association (HGF). Hgf website. https://www.helmholtz.de/en/. Accessed: 2017-07-09.

[31] JSC. DEEP prototype website. http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/DEEP/DEEP_node.html. Accessed: 2017-07-16.

[32] JSC. High-Q club website. http://www.fz-juelich.de/ias/jsc/EN/Expertise/High-Q-Club/_node.html. Accessed: 2017-08-10.

[33] Forschungszentrum Jülich (FZJ). Fzj website. http://www.fz-juelich.de/en. Accessed: 2017-07-09.

[34] Jülich Supercomputing Centre (JSC). JSC website. http://www.fz-juelich.de/ias/jsc/EN. Accessed: 2017-07-09.

[35] Dorian Krause and Philipp Thörnig. JURECA: General-purpose supercomputer at Jülich Supercomputing Centre. *Journal of large-scale research facilities*, 2:A62, 2016.

[36] Anke Kreuzer, Jorge Amaya, Norbert Eicker, Raphaël Léger, and Estela Suarez. The DEEP-ER project: I/O and resiliency extensions for the Cluster-Booster architecture. In *Proceedings of the 20th International Conference on High Performance Computing and Communications (HPCC)*, Exeter, UK, 2018. IEEE Computer Society Press. (accepted for publication).

[37] Anke Kreuzer, Norbert Eicker, Jorge Amaya, and Estela Suarez. Application performance on a cluster-booster system. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 69–78, May 2018.

[38] Pramod Kumbhar, Michael Hines, Aleksandr Ovcharenko, Damian Alvarez, James King, Florentino Sainz, Felix Schürmann, and Fabien Delalondre. Leveraging a Cluster-Booster Architecture for Brain-Scale Simulations. In *Proceedings of the 31st International Conference High Performance Computing*, volume 9697 of *Lecture Notes in Computer Science*, pages 363 – 380, Cham, Jun 2016. 31st International Conference High Performance Computing, Frankfurt (Germany), 19 Jun 2016 - 23 Jun 2016, Springer International Publishing.

[39] Jefferson Lab. Chroma github repository. https://jeffersonlab.github.io/chroma/. Accessed: 2017-07-28.

[40] Raphäel Léger, Damian Alvarez Mallon, Alejandro Duran, and Stephane Lanteri. Adapting a Finite-Element Type Solver for Bioelectromagnetics to the DEEP-ER Platform. In *Parallel Computing: On the Road to Exascale*, volume 27 of *Advances in Parallel Computing*, pages 349 – 359. International Conference on Parallel Computing 2015, Edinburgh (UK), 1 Sep 2015 - 4 Sep 2015, IOS Press Ebooks, Sep 2016.

[41] Thomas Lippert. *Recent Developments in Supercomputing*, volume 39 of *NIC series*, pages 1–8. John von Neumann Institute for Computing, Jülich, 2008. Record converted from VDB: 12.11.2012.

[42] NEST. NEST code website. www.nest-simulator.org. Accessed: 2017-07-28.

[43] Pogramming Models @ BSC. The OmpSs Programming Model, 2013.

[44] The DEEP-ER project. DEEP-ER project website. http://www.deep-er.eu. Accessed: 2017-07-16.

[45] The DEEP Projects. DEEP project website. http://www.deep-project.eu. Accessed: 2017-07-16.

[46] The DEEP projects. Full Wave Inversion (FWI) code in DEEP-ER. http://www.deep-projects.eu/applications/project-applications/enhancing-oil-exploration.html. Accessed: 2017-07-28.

[47] The DEEP projects. SKA data analysis pipeline in DEEP-ER. http://www.deep-projects.eu/applications/project-applications/radio-astronomy.html. Accessed: 2017-07-28.

[48] QPACE3. QPACE3 website. http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/QPACE3/_node.html. Accessed: 2017-07-16.

[49] SchedMD. SLURM website. https://slurm.schedmd.com/.

[50] SeisSol. SeisSol application website. http://www.seissol.org/. Accessed: 2017-07-28.

[51] JSC Simlabs. Website. http://www.fz-juelich.de/ias/jsc/EN/Expertise/SimLab/simlab_node.html. Accessed: 2017-08-07.

[52] Michael Stephan and Jutta Docter. JUQUEEN: IBM Blue Gene/Q Supercomputer System at the Jülich Supercomputing Centre. *Journal of large-scale research facilities*, 1:A1, 2015.

[53] Estela Suarez, Norbert Eicker, and Thomas Lippert. Supercomputer Evolution at JSC. In *Proceedings of the NIC Symposium*, volume 49 of *Publication Series of the John von Neumann Institute for Computing (NIC) NIC Series*, pages 1 – 12, Jlich, Feb 2018. NIC Symposium 2018, Jülich (Germany), 22 Feb 2018 - 23 Feb 2018, John von Neumann Institute for Computing.

[54] Top500. The list. https://www.top500.org/lists/. Accessed: 2017-06-26.

[55] Anna Wolf, Anke Zitz, Norbert Eicker, and Giovanni Lapenta. Particle-in-Cell algorithms on DEEP:The iPiC3D case study. volume 32, pages 38–48, Erlangen, May 2015. PARS 15, Potsdam (Germany), 7 May 2015 - 8 May 2015, PARS.