Modular Supercomputing Architecture

A success story of European R&D

White Paper

Estela Suarez, JSC / University of Bonn,
Norbert Eicker, JSC / Bergische Universität Wuppertal
Thomas Moschny, ParTec AG,
Simon Pickartz, ParTec AG,
Carsten Clauss, ParTec AG
Valentin Plugaru, LuxProvide,
Andreas Herten, JSC,
Kristel Michielsen, JSC / RWTH Aachen University,
Thomas Lippert, JSC / Goethe-Universität Frankfurt

11/05/2022

etp4hpc.eu

@etp4hpc

ETP 4 HPC

EUROPEAN TECHNOLOGY PLATFORM FOR HIGH PERFORMANCE COMPUTING

Table of Contents

Table of Figures

Introduction

The European Community and its member states regularly invest large volumes of funding and effort in the development of HPC technologies in Europe. However, some observers express the criticism that these investments are either unfocused, lack long-term perspectives, or that their results are not mature enough to be adopted by the mainstream developments, which limits their benefit for the European HPC community, industry and society. This paper is intended as a counterexample to this pessimistic view. It describes the success story of Modular Supercomputing Architecture, which started in 2011 with the EU-funded R&D project "DEEP", and is now being adopted by large-scale supercomputing centres across the old continent and worldwide. Main hardware and software characteristics of the architecture and some of the systems using it are described, complemented by a historical view of its development, the lessons learned in the process and future prospects.

Key insights

- Modular Supercomputing Architecture (MSA) has been developed through the EU-funded DEEP projects over the last 10 years. MSA is now being adopted across Europe and world-wide, with pre-Exascale production systems already based on this concept.
- MSA realises the concept of functional parallelism at the highest organisational level of a supercomputing system.
- MSA orchestrates hardware heterogeneity at the system level, organising the resources in various modules, each one a cluster computer in its own right.
- It builds systems that can dynamically adapt to the needs of a wide range of applications and strives for serving both HPC and HPDA/AI applications.
- MSA enables the integration of very diverse hardware technologies, including disruptive approaches such as quantum or neuromorphic computers.
- For any new architecture or hardware approach, a reliable and mature software stack is essential for the HPC users to embrace the whole concept.
- HPC technology development is a marathon not a sprint. It requires years of investment before it reaches a high level of maturity. It is crucial to keep up the engagement in order to achieve a final success.

Key recommendations

- Co-design: bring together hardware, system-software, and application developers and develop a mutual understanding for the needs, priorities, and framework conditions of each of these HPC-players.
- Deploy software-development vehicles and demonstrate your developments through hardware and software prototypes, before implementing the solutions in a production environment.
- Reduce dependencies on specific hardware developments, by creating software layers that are agnostic to the underlying platform, and provide a unified interface to the applications.
- Invest at least as much effort and funding on the software developments, as you do on the hardware side, and give the former enough visibility.
- Build up a team of engaged individuals, build up trust between your partners, encouraging all to discuss problems openly and focusing on finding solutions together.
- Be aware that difficulties will arise. Identify your risks and monitor them closely. Be prepared with alternative plans to go around any roadblock.

1. Architecture overview

The Modular Supercomputer Architecture (MSA) [1] interconnects a variety of compute modules to address general and diverse user requirements, ranging from computationally intensive high-scaling simulation codes to data-intensive artificial intelligence workflows. The diversity of application profiles is further increasing, as does the specificity of their requirements, thus serving them all with a homogeneous, monolithic system becomes impossible.

Within MSA, several compute modules of potentially large size – each one tuned to best match the needs of a certain class of algorithms – are connected to each other at the system level to create a single heterogeneous system (see Figure 1). Power consumption is kept at bay by only using power-hungry, general purpose processing components where these are absolutely necessary (in the cluster module). Energy efficient, highly parallel technologies (e.g. GPUs) are scaled up and employed for high-throughput and capability-computing codes (in the booster module). Disruptive technologies (e.g., neuromorphic or quantum) can be integrated through dedicated modules.

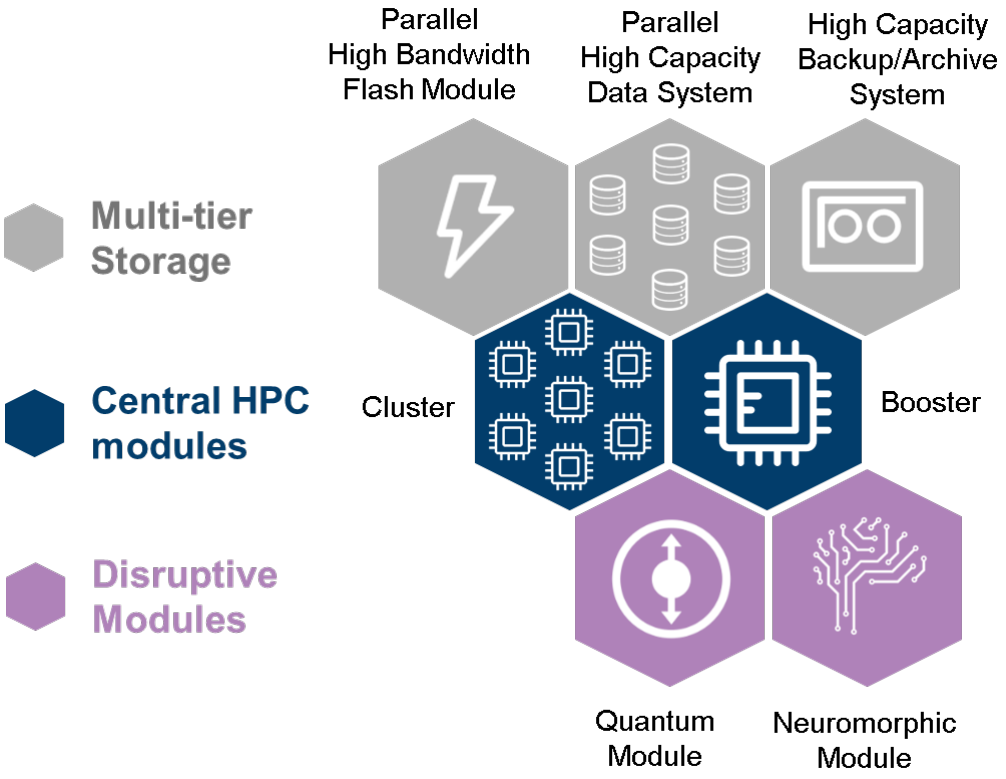


Figure 1 - Modular Supercomputing Architecture (MSA)

A common software stack enables users to map the intrinsic application requirements (e.g., their need for different acceleration technologies and varying memory types or capacities) onto the hardware; highly scalable code parts run on the energy efficient booster (a cluster of accelerators), while less scalable code parts profit from single-thread performance of the general purpose cluster. Likewise, new types of hybrid application workflows will be possible by exploiting the capabilities provided by modules implementing disruptive technologies, all under the umbrella of the common software stack. Optimisations for the hardware and middleware are implemented within the stack, enabling both developers and end-users of production codes to take advantage of the modular system. A unified software stack is key for easing application coupling and improved agility, as more and more workflows require a complex execution pipeline to produce final results.

2. Origins: The DEEP projects

Modular Supercomputer Architecture (MSA) is the result of 10 years research on computer architectures at the Jülich Supercomputing Centre (JSC), primarily through the EU-funded DEEP project series [1]. It all started in 2010 when the team at JSC debated whether it would be possible to build an HPC system that combined the best characteristics of the two production machines that were running on the site at that time: the general purpose cluster JUROPA (based on Intel CPUs), and the massively parallel system JUGENE (an IBM BlueGene/P machine) [2].

The goal was to design a system-level architecture that offers the flexibility and high single-thread performance of a general purpose cluster, while achieving the energy efficiency of a massively parallel platform. To increase energy efficiency, the general trend at that time was to build heterogeneous compute-nodes by attaching accelerators (typically general purpose graphic cards, GPGPUs) to the CPU. The JSC-team, however, was looking for a less static system design in which compute resources could be easily shared between applications in order to enable a more dynamic allocation. This would be possible if one could segregate the accelerators and create a pure accelerator-based cluster -- which received the name *booster* -- and couple it to a standard, CPU-based system. This *cluster-booster concept* was the seed of what we now call *Modular Supercomputing*.

The idea was promising but, first of all, it had to be demonstrated. Therefore, detailed technical discussions started with leading industrial and academic players in the European HPC landscape. The meetings were so fruitful that they led to the set up of a consortium with 16 partners and the submission of a proposal for the DEEP project (standing for *Dynamical Exascale Entry Platform*). The DEEP project was selected within the first Exascale R&D calls in the FP7 program from the European Commission, and it started in November 2011 [3]. Therein, the first cluster-booster prototype was built (see section DEEP p.11), its software stack was developed (see section 3. Software environment), and applications were analysed and tested. DEEP ran for 45 months and was followed by DEEP-ER [4] (2013-2017) and DEEP-EST (2017-2021) [5].

Each of these project phases addressed specific challenges: building the first cluster-booster prototype, improving the I/O and resiliency capabilities, and generalising the idea toward the Modular Supercomputing Architecture. But there is one characteristic that is shared by all of them and is key to their success: a very tight cooperation between developers from hardware, system software, and applications in a holistic co-design endeavour.

Keeping this philosophy, the R&D work around MSA continues within the SEA projects (2021-2024): DEEP-SEA (enhances the software stack for modular and heterogeneous HPC systems), IO-SEA[2] (improves the I/O and data management capabilities at scale), and RED-SEA[3] (develops next-generation interconnect technologies).

[1] DEEP projects website: https://www.deep-projects.eu/

[2] IO-SEA project website: https://iosea-project.eu

[3] RED-SEA project website: https://redsea-project.eu

3. Software environment

MSA adds a third level to the topology of a supercomputer by introducing the notion of modules. This concept brings benefits to both the users and the system operators, but at the same time it adds to the complexity of the overall software stack.

Each module can be considered as a traditional, monolithic parallel computer, suggesting the Multiple Program Multiple Data (MPMD) programming model for the utilisation of multiple modules. However, this approach would force the application developers to handle the inter-module communication explicitly and independently of the intra-module traffic. This, in turn, would prevent possible optimisations regarding the communication. Therefore, it is indispensable to embed the concept of modularity within the MPI programming model while providing a transparent view on the underlying architecture. Nevertheless, completely hiding the module level within MPI would prevent further optimisations on the application level.

Therefore, the information on system topology should be exploited by the runtime and likewise be made available to the applications in a generic way. For one thing, this enables architecture-specific optimisations such as optimised collective communication. This approach also supports modular application designs to evolve with the MSA system. These considerations lead to two main requirements that must be fulfilled by the programming environment of an MSA system:

1. The supply of a transparent view on the modular architecture possibly comprising different types of high-speed interconnects, and
2. the exploitation of the information on system topology within the runtime as well as its provision to the application layer.

Network Federation

The MSA concept incorporates support for different network architectures within distinct modules. The runtime-system of the programming environment is in charge of transparently connecting these networks to meet the first requirement listed above. This calls for a generic solution bridging between any pair of interconnects in favour of solutions being tailored to specific hardware [6].

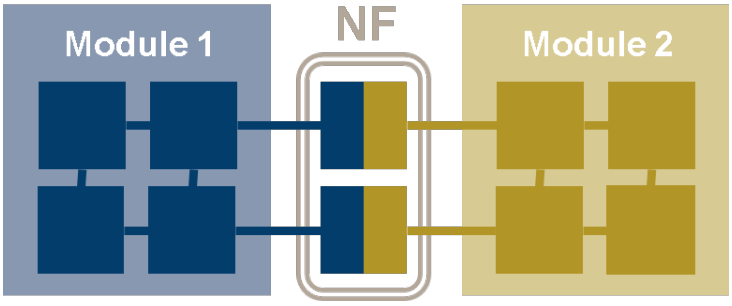


Figure 2 - Gateway-based approach: Network Federation (NF) connecting two different MSA modules by means of multiple gateway nodes

A gateway-based solution (see Figure 2) promises good performance characteristics by enabling a tight coupling of different modules and hence better fits the MSA context. Here, dedicated nodes are part of at least two high-speed fabrics within the system and act as gateways forwarding network traffic from one module to another. One could argue that this approach is likely to introduce communication bottlenecks at the gateways. However, the MSA concept requires a coarse-grained partitioning of applications reducing the inter-module communication to a minimum to avoid such a behaviour [7].

ParaStation MPI

The ParaStation MPI communication stack is a central pillar of ParaStation Modulo and is developed by ParTec under the umbrella of the ParaStation Consortium. ParaStation Modulo is an MSA-enabled software suite that powers the DEEP projects and is also extensively used in production environments such as JURECA, JUWELS, and MeluXina (see section 5. Production systems). ParaStation MPI is an MPICH [8] derivative relying on the pscom library [9] for point-to-point communication. This is a high-performance communication library especially designed for HPC, supporting different communication transports concurrently, and used to implement an ADI3 device within MPICH. The pscom library likewise exhibits a layered structure including a plugin layer to support different communication interfaces and protocols (e. g., InfiniBand (IB) [10], Omni-Path [11], and EXTOLL [12]) in parallel.

ParaStation MPI enables transparent inter-module communication in heterogeneous network landscapes by implementing a gateway-based approach. For this purpose, the pscom library has been extended by two components: the pscom gateway daemon (*psgwd*) and the gateway plugin supporting the communication between regular MPI processes and the gateway daemons. The gateway plugin enables message-forwarding between different network transports, relying on the native plugins for the actual data transfer within each fabric. This mechanism is used by both the MPI processes and the daemons.

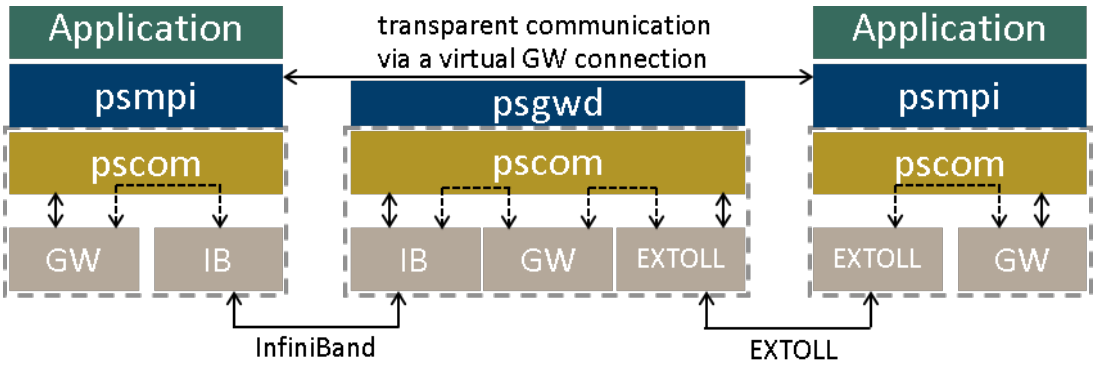


Figure 3 - Overview of the network bridging within ParaStation MPI. This relies on a gateway-base approach and is implemented as part of the pscom library

Figure 3 visualises the gateway solution as implemented in the first DEEP prototype to bridge the MPI traffic between IB and EXTOLL. The gateway daemons are transparently started by the resource management as part of ParaStation Modulo during the initialisation of the communication session of a modular job. If an MPI process residing within the IB network sends a message to an MPI process located on a node connected to the EXTOLL network, it uses the gateway plugin to transmit the message to the connecting gateway daemon which, in turn, forwards it to the destination process.

This mechanism is transparent to the communicating processes, i.e. they see a direct gateway connection for inter-module communication when using the gateway plugin, and supports the bridging between any pair of interconnects and communication protocols supported by the pscom. This way, further network technologies (such as Slingshot and BXI) could be easily integrated into an existing modular communication stack, with pscom serving as the bridge enabled by new pscom plugins. At this point it has to be noted that inter-module communication may require additional copy operations at the psgwd depending on the low-level communication interfaces. The additional hop on the communication path introduced by the gateway-based NF impacts communication latency and throughput. Therefore, optimisations are in place to reduce the discrepancy between the intra-module and the inter-module communication performance, such as fragmentation of large messages to benefit from pipelining effects and avoiding intermediate copy operations.

MSA awareness at MPI level

The distinct nature of MSA requires the MPI library to address modularity beyond the capability of forwarding messages between modules. This comprises transparent application of modularity-aware optimisations to avoid communication bottlenecks between the modules, e. g., when employing collective communication patterns. While this transparency facilitates the porting of applications to the MSA, it may be desirable to provide information on its heterogeneity and topology at the application level as well.

If the interface to this information provides MPI compliance, an incremental and backward compatible adaptation of applications to the MSA becomes possible. For this purpose, ParaStation MPI provides each process with the information about its own module affiliation within the MSA via the standardised `MPI_INFO_ENV` object. By leveraging this information, each process is able to create new MPI communicators reflecting the overall topology of the MSA.

The information on the module affiliation is not only available to the application alone but also to ParaStation MPI itself: it enables the utilisation of adapted patterns to optimise collective communication operations with respect to the system's MSA topology. This approach is not limited to the module affiliation but can be extended, for example, also to the node affiliation. Through a recursive use of the topology information, adaptations of collective communication patterns can be achieved even to multi-tier topologies by ParaStation MPI for optimisation.

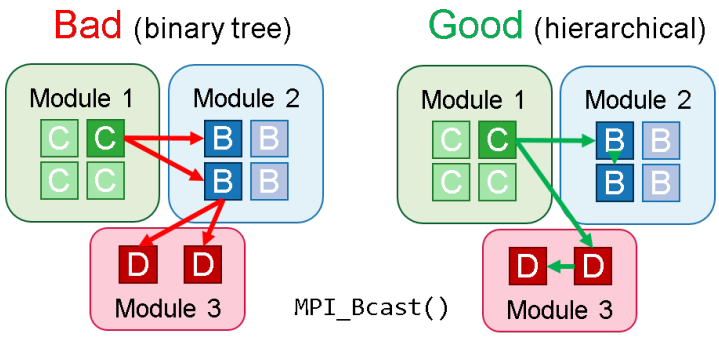


Figure 4 - Principle behind the MSA-aware optimisation of MPI collectives

Figure 4 shows an example for the realisation of a simple broadcast pattern on an MSA: While on the left-hand side ("Bad") a simple binary tree is transparently applied, on the right-hand side ("Good") a hierarchical pattern is used, which minimises the inter-module messages by channelling such traffic through specific processes.

ParaStation Management

The runtime for parallel applications in general, and modular applications in particular, is provided by ParaStation Management, the process management facility of ParaStation Modulo. It provides all elements required to form the lifecycle of an application, including remote process creation and termination, the control of their I/O channels, the management of signals across node and module boundaries, as well as process separation and accounting features.

ParaStation Management (see Figure 5) treats all processes and threads started as part of a parallel application as a single entity. This way, it is able to perform job control, allocation-internal scheduling, and resource assignment on a very fine-granular level. In doing so, it considers features of the underlying, possibly heterogeneous hardware, and tightly integrates with a batch queuing system and scheduler such as Slurm[4]. The runtime elements of ParaStation Management,

[4] Slurm website: https://slurm.schedmd.com/

i.e. the so-called ParaStation Daemons (*psid*), are communicating with each other by means of a very efficient and scalable, datagram-based communication subsystem.

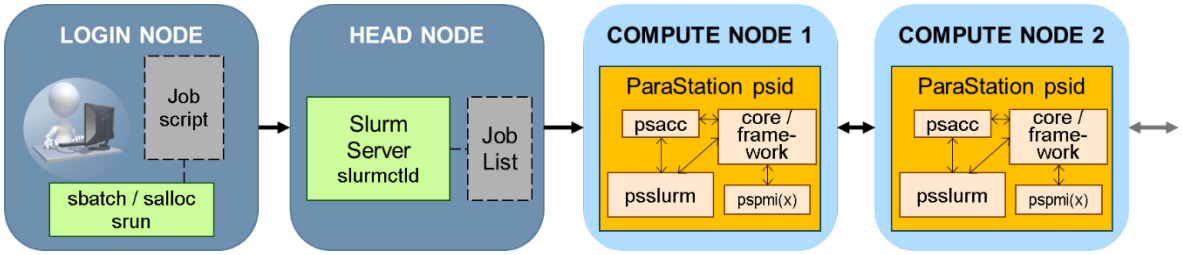


Figure 5 - Control flow for submitting a job to the Slurm scheduler when using ParaStation Management. The ParaStation Daemons (psid) completely replace the node-local Slurm daemons and directly interact with the Slurm control daemon.

ParaStation Management can be integrated with batch queuing systems supporting efficient scheduling in modular systems. In particular, *psid*'s *psslurm* plugin allows using ParaStation Management in conjunction with Slurm. In doing so, it completely replaces the *slurmd* processes running on the computing nodes and directly talks to the Slurm control daemon *(slurmctld)* running on the master node. This way, ParaStation Management enables implementing MSA-aware features that are not part of the standard Slurm implementation. For example, Slurm's lack of support for non-compute resource management is compensated by a corresponding mechanism provided by ParaStation Management, which is used amongst others to manage the resources on the gateway nodes and to realise network federations in modular systems. Users are now able to use those resources as part of their (heterogeneous) jobs including the execution of additional user processes (i.e., these are the *psgwd* daemons of the *pscom* in case of gateway communication between different high-performance interconnects).

Architecture-Specific Software Environment

MSA revolves around providing a heterogeneous set of computers as an integrated system. The architecture of these individual computers might be entirely different, even with different instruction set architectures (ISA). But different micro-architectures might exhibit different performance characteristics, too. HPC aims to provide the most efficient implementations and configuration of software, which is tightly integrated with specific architectures, and as such typically provides individual software and environment configuration for each module. Examples include compilers and libraries optimised for different micro-architectures (AVX512 or RISC-V, for example), or MPI tuning options (like protocol size thresholds). In many HPC centres, environment modules (*Lmod*[5]) are used to introduce software and configuration to the environment; using different environment modules for different hardware modules. To enable the architecture-specific software through environment modules within batch jobs, different approaches can be taken. The easiest – albeit most cumbersome – is the addition of wrapper scripts, wrapping the execution of the application with environment module loading. To facilitate this task, an advanced, Python-based tool is available at JSC: *xenv* – inspired by Unix's *env* – enables the introduction of different environment modules to processes by prepending the application call[6]. An example:

```
srun xenv -L GCC/avx512 ./myapp_a : xenv -L GCC/risv ./myapp_b
```

It uses Slurm to launch a heterogeneous job of two parts (separated by the colon) and each part using different environment modules through `xenv`.

[5] Lmod github repository: https://github.com/TACC/Lmod
[6] See also: https://apps.fz-juelich.de/jsc/hps/jureca/modular-jobs.html

Scheduler

To make efficient use of MSA, the scheduling system has to take the system-level heterogeneity into account. This is not limited to the distinction between accelerated and non-accelerated computing nodes, but extends to different accelerator types, heterogeneous memory hierarchies, and hierarchical storage systems. This variety of resources, even including global, non-compute resources (e.g. gateway resources or I/O nodes) has to be made available to application scenarios ranging from coupled co-simulations to application workflows executing each step on the most suitable platform one after another. As Slurm–the most widely used open source scheduling system in HPC as of today–lacks support for the management of global resources, ParaStation Management takes over the task of managing the gateway nodes in MSA systems with network federation.

To make efficient use of MSA, the scheduling system has to take the system-level heterogeneity into account

To address the applications' need for more flexibility in terms of required and available resources, the resource management and scheduling facilities will have to provide support for malleability in the future. This requires the negotiation of resource adjustments between applications and/or workflow engines and the resource manager. That way, large-scale application workflows can be started without the necessity for all resources being available from the beginning. This kind of customisation, in which the ratio and the type of resources can be defined during runtime to match each application, is inherent to MSA. It is realised with a dynamic scheduling and resource management, which has to be modular itself, to match the demands of system operators for standardised interfaces between the different components of the system software. The Process Management Interface PMIx is such an open standard designed to act as the interface between runtime systems and resource managers and is (in its current form) already supported by several MPI implementations (e.g., ParaStation MPI) and several resource managers (e.g., Slurm).

4. Hardware prototypes

The best way to demonstrate a new system-architecture is to build hardware prototypes and use them as proof-of-concept. They are used to test a specific hardware implementation, develop and implement the necessary system software, and test and validate the principles using applications. For MSA, this work was done within the DEEP projects.

DEEP

The first MSA-computer was the DEEP prototype built in 2013 by the integrator Eurotech. The DEEP cluster was an off-the-shelf system with 128 Intel Xeon Haswell CPUs in an InfiniBand FDR network. The DEEP booster hosted 384 Intel Xeon Phi (Knights Corner, KNC) many-core processors in an Extoll interconnect. The main challenge was building the booster in such a manner that the KNCs could be operated as autonomous entities. This meant also running them out of their specifications, which made it crucial to have Intel's technical expertise in the project. The root-complex capabilities of the Extoll network and a clever low-level software enabled the booster operation. Specially designed interface cards were responsible to boot the KNC nodes and bridge between cluster and booster, acting as gateways to translate messages from IB to Extoll, and vice versa. The construction of this first booster was a high-risk endeavour and its operation suffered from hardware failures typical of its prototype quality, but it paved the first stone in the path which ultimately brought high-gain returns. In fact, it was interesting to see how soon it became much easier to build a booster, with the Intel Xeon Phi Knights Landing (KNL) generation, which was designed as an autonomous accelerator. It is not too far-fetched to believe that the vision of an autonomous cluster of accelerators and the DEEP experience with KNC might have had some influence in the KNL development.

Figure 6 - DEEP prototype (cluster and booster)

DEEP-EST

The arguments that led to the cluster-booster architecture were based on energy efficiency and the existence of different intrinsic scalability patterns in the applications. However, not only scalability-concerns are at play when a user chooses one computing platform against another. General requirements such as amount of memory, memory bandwidth, and vectorisation capabilities are as important. HPC centres observe how these requirements are becoming more diverse, since new use-cases from the high-performance data-analytics and machine learning domains are now also part of the user portfolio. For this reason, it became natural to generalise the idea of attaching various differently configured computer clusters with each other and build a modular supercomputer.

At about the same time (2018) the discontinuation of the Intel Xeon Phi line was announced, reopening the question of how to build a booster. The only remaining accelerators widely-available and well-adopted by the HPC-user communities were then general purpose graphic processing units (GPGPUs). This was the choice made for the DEEP-EST prototype, built in 2019 by Megware. It contains three modules: a 50-node cluster (using Intel Xeon Skylake CPUs), a 75-node booster (with lower-end Intel Cascade Lake CPUs and one NVIDIA V100 GPU per node), and a 16-node data analytics module (with an Intel Cascade Lake CPU, one NVIDIA V100 GPU, and one Intel Stratix-10 FPGA per node).

The result is a platform that achieves very high energy efficiency, reliability and operational stability by taking advantage of commercial-off-the-shelf technology in a very robust system integration. Since the bulk of the computation is executed on the GPUs, a lower skew was chosen for the host-CPU in the Booster, so that its power consumption has a lower impact on the final cost envelope. Cluster and booster are direct-liquid cooled. The data analytics module, on the other hand, offers maximum flexibility by providing different kinds of accelerators (GPUs and FPGAs) in an air-cooled chassis.

The DEEP-EST platform has been evaluated by the project partners and also by the wider community through an early-access program. The experiences from application developers porting and testing their codes on this MSA prototype have been collected and documented for future users [13]. After the end of the DEEP-EST project, the system continues in operation at JSC and is the main software development platform for the DEEP-SEA project. It is integrated in the login and user-management infrastructure of JSC and available for both in-house, and external interested users.

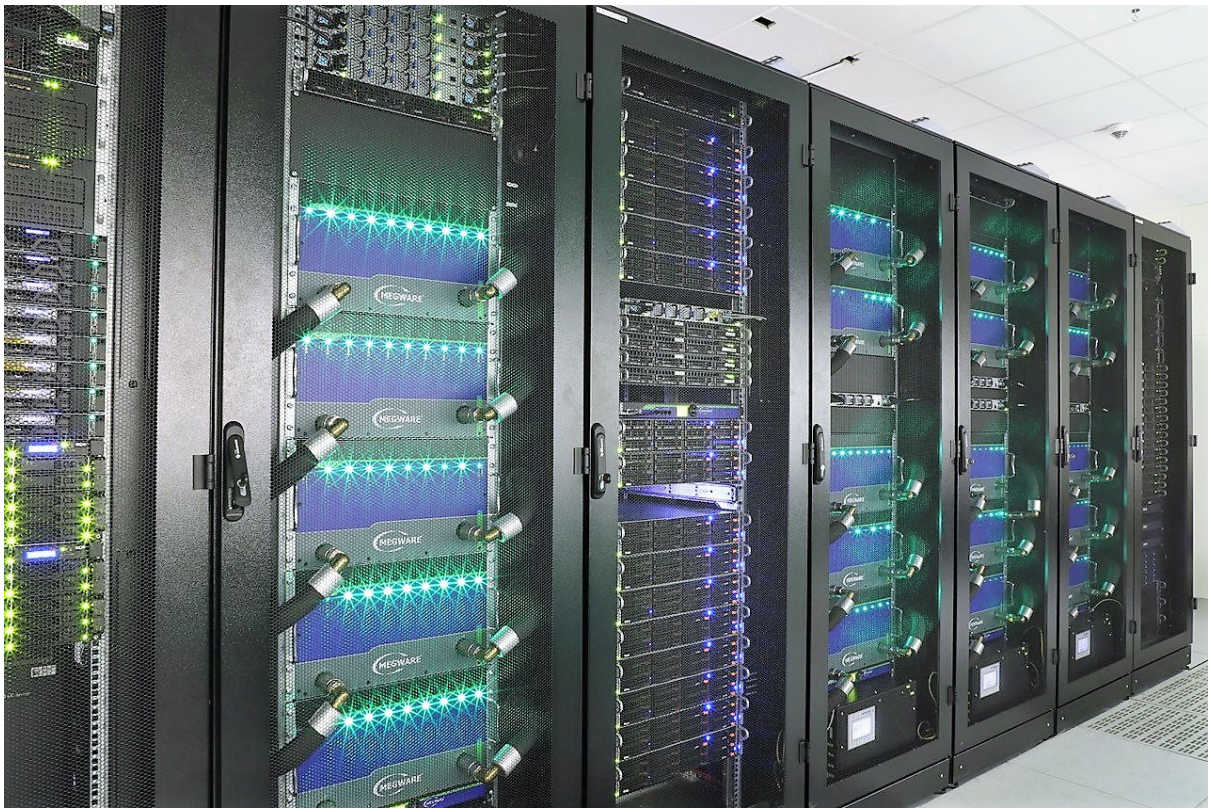


Figure 7 - DEEP-EST prototype (cluster, booster, and data analytics module)

5. Production systems

In November 2017, JSC's JURECA became the first modular (cluster-booster) system worldwide in the Top 500, reaching position 29 with the Linpack benchmark running over both partitions. Today, the MSA concept is well established on the HPC landscape, with both the Petascale systems MeluXina and JUWELS based on this system architecture.

JURECA

Given the experience collected with the DEEP system, the cluster-booster idea was brought to production for the first time in JSC's JURECA system. JURECA originally started as what Jülich calls the "general purpose HPC system" [14] (i.e., a classical HPC-Cluster based on Intel Xeon E5-2680v3 processors interconnected by InfiniBand EDR), in contrast to the series of BlueGene capability systems. It served as the successor of the successful JUROPA systems. In 2017 the JURECA cluster was accompanied by a booster system utilising Intel Xeon Phi 7250-F many-core processors of the KNL (Knights Landing) generation employing an OmniPath fabric for communication [15]. The OmniPath network was interconnected to the InfiniBand EDR network of the JURECA cluster via 198 gateway nodes, i.e., nodes equipped with one InfiniBand HCA and one OmniPath HFI each running the ParaStation *psgwd* providing data-forwarding services for the ParaStation Gateway protocol. For this, ParaStation Modulo was extended in order to provide bridging services between InfiniBand and OmniPath, too. As with all JSC's production systems, JURECA is embedded in JSC's JUST storage infrastructure.

The purpose of the JURECA booster was threefold: First of all it enabled the JURECA system to also serve highly scalable applications that were reserved for the JUQUEEN system before. Secondly, the booster operated as a bridge system between the to be decommissioned JUQUEEN systems and the JUWELS booster, which would come a few years later (see section JUWELS p.14). Last but not least, for the first time it brought the cluster-booster concept to a large-scale production system enabling heterogeneous applications to benefit from its enhanced capabilities.

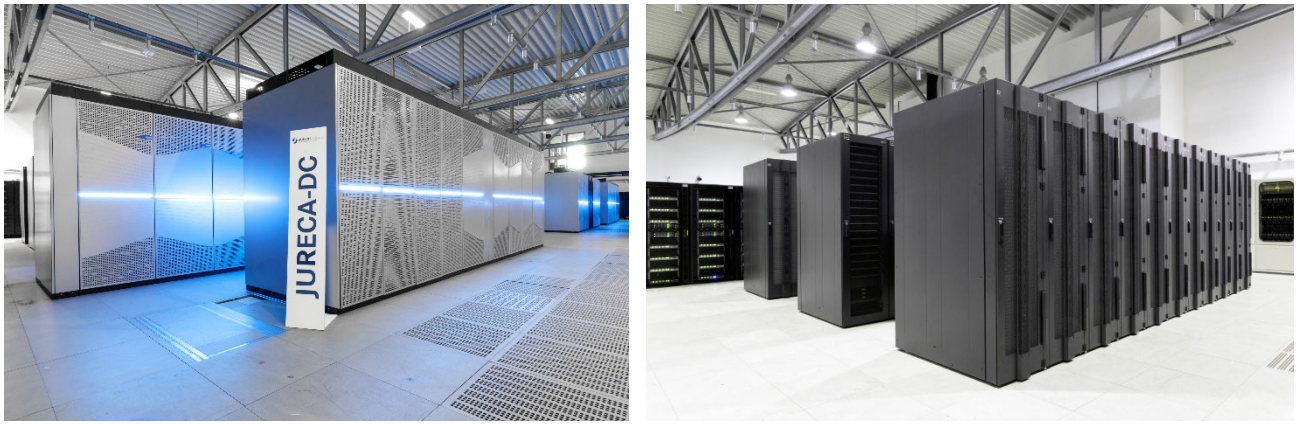


Figure 8 - JURECA system. Left: JURECA-DC cluster. Right: JURECA booster.
© Forschungszentrum Jülich GmbH, Ralf-Uwe Limbach

In the meantime, the cluster part of JURECA has been updated to JURECA-DC [16], an Atos BullSequana cluster employing AMD EPYC 7742 processors and InfiniBand HDR100. The 198 gateway nodes are still in place to bridge between the cluster and the OmniPath-based booster. Thus, JURECA is a perfect example that MSA enables the opportunity to extend and update a production system in the most flexible way.

JUWELS

JUWELS [17] is a modular supercomputer operated by JSC as a European and German computing resource. JUWELS currently consists of two modules, a cluster and a booster. The cluster – installed in 2018 – is an Atos BullSequana X1000 system with about 2500 nodes, equipped with Intel Skylake CPUs and Mellanox EDR InfiniBand (100 Gbit/s). In 2020 it was accompanied by the booster, a 936 node Atos BullSequana XH2000 system, each node equipped with four NVIDIA A100 GPUs and four Mellanox HDR InfiniBand adapters (200 Gbit/s each) managed by two AMD EPYC 7402 processors. The combined theoretical peak performance is about 80 PFLOP/s, which ranked it 23rd (Cluster in November 2018 with 6.2 PFLOP/s HPL performance) and 7th (Booster in November 2020 with 44 PFLOP/s HPL performance), respectively. Both modules feature an InfiniBand-based interconnect, lowering complexity compared to previous modular installations at JSC, and are operated using ParTec's ParaStation Modulo software suite.

Figure 9 - JUWELS system, with its cluster (left) and booster (right) modules.
© Forschungszentrum Jülich GmbH, Ralf-Uwe Limbach

JUWELS is catering to users needing the highest computer performance. It is used by various scientific domains from different national and European communities. Examples include applications in particle physics (plasma, Lattice QCD), medicine, biology and soft-matter, computational fluid dynamics, quantum computer simulation, and weather and climate. Both JUWELS modules are heavily used and their compute time is regularly over-requested. JUWELS booster was already busy from the first day of production, also because of intensive preparation through the JUWELS Booster Early Access Program [18]. The 3600 GPUs of the system, offering accelerated tensor operations and mixed precision, are especially suited for AI research [19].

MeluXina

All the systems above have been installed and operated at JSC, but MSA is also adopted in other HPC centres. Through flagship national digital programmes, the Grand Duchy of Luxembourg is prioritising initiatives on HPC, AI and Big Data analytics, key building blocks for digital competitiveness. The vision that emerged from a national reflection on requirements led to the creation of LuxProvide, Luxembourg's uniquely differentiated HPC Center, and the development of its MeluXina supercomputer concept into a robust platform for science and industry. MeluXina became live in 2021 as one of the first operational systems in the EuroHPC network and one of the most powerful and energy-efficient supercomputers in Europe[7].

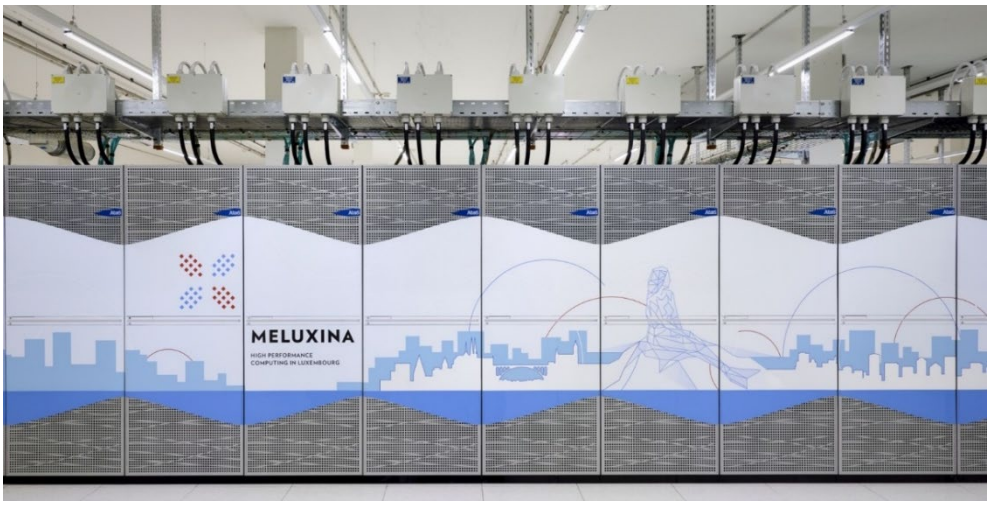


Figure 10 - MeluXina system: cluster and accelerator modules.
© LuxProvide S.A.

MeluXina's forward-looking design responds to the convergence of simulation, modelling, data analytics and AI, and enables simulation driven by predictive analytics[8]. Its main computing modules are integrated in Atos BullSequana racks and contain AMD EPYC CPUs with high core count and at least half a TB of DDR4 memory, future-proofing the system for memory (and memory-bandwidth) intensive applications, while providing high network performance through the use of the Infiniband HDR interconnect. The largest compute node partition (573 nodes / 73344 cores) is meant to offer outstanding performance for most workloads and compatibility for the broadest range of applications. The fastest compute partition is accelerated (200 nodes / 800 NVIDIA Ampere GPUs) and designed for workloads exhibiting strong data parallelism, thus perfectly suited for AI, ML/DL and data analytics work. The GPU nodes have dual-rail connectivity to the fabric, driving improved performance for tightly coupled applications. Linked to the same fabric is MeluXina's tiered data platform, based on multiple Lustre parallel file systems. The fastest data tier is built on half a PB of NVMe drives with over 500 GB/s bandwidth, very high relative to the number of compute nodes within the system and thus ideal for accelerating data-driven applications. A larger data tier with over 12 PB of NVMe/HDD storage provides the main capacity for high volume projects and datasets while still being high-performing, delivering high bandwidth at over 230 GB/s. The operational management and monitoring of core system services is performed through a custom tailored software stack, while the ParaStation Modulo software suite facilitates the deployment and management of the MeluXina compute nodes.

[7] MeluXina inauguration - eurohpc-ju.europa.eu/press-release/meluxina-live-eurohpc-ju-supercomputer-luxembourg-operational

[8] MeluXina architecture details, computing, data and software technologies - docs.lxp.lu

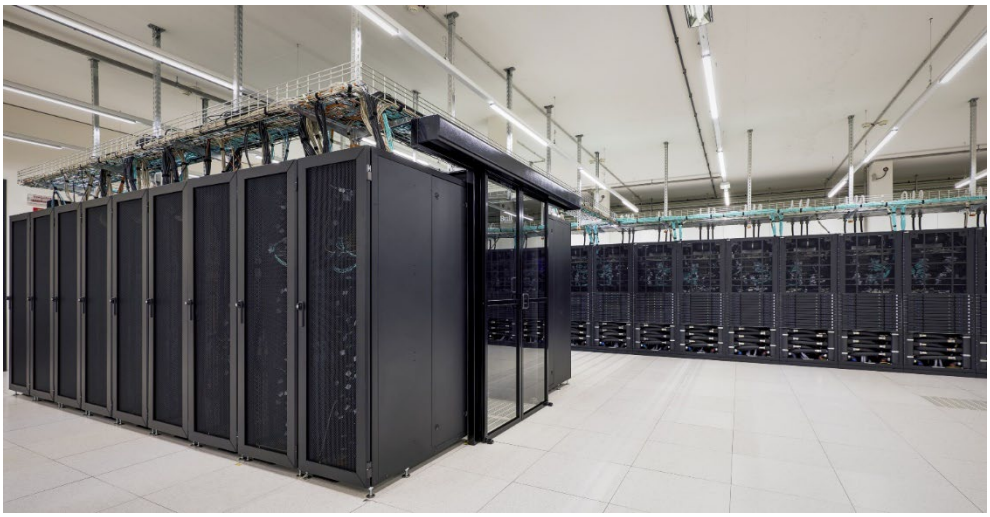


Figure 11 - MeluXina system: storage, system and cloud modules.
© LuxProvide S.A.

A private Cloud module based on the OpenStack platform provides facilities for running APIs, dashboards, workflow orchestration, tools for data intake from sensor platforms and interactive data exploration applications, complementing the core supercomputing capabilities of MeluXina and meant to drive computational workloads on its other modules.

A rich software environment enables users to take advantage of the systems' computational and data capabilities, with hardware-optimised software built using the EasyBuild[9] platform. From 2021 over 300 packages are provided to users, comprising compilers, programming languages, performance engineering tools, a variety of numerical, acceleration and data libraries, frameworks (especially for ML/DL - PyTorch, TensorFlow, Horovod, Keras), and also end user applications (such as GROMACS, OpenFOAM, CP2k, QuantumESPRESSO, HOOMD-blue, NAMD, NWChem) covering a variety of domains.

Counting on over 18 PFLOP/s of aggregated raw FP64 performance (the GPUs themselves capable of delivering 500 PFLOP/s of AI performance) and over 20 PBytes of data, MeluXina is meant to tackle workloads coming from a wide variety of domains, from space to health technologies, environment, manufacturing, engineering and finance. The modular system is ideally suited to accelerate simulations for building highly accurate digital twins, driving added value for societal applications. Truly bringing the Modular concept to life, the hybrid technologies part of MeluXina provide a cutting-edge base for excellence in science and innovation, able to cater to the needs of a wide range of users from industry, the public sector and the scientific community.

MeluXina is meant to tackle workloads coming from a wide variety of domains, from space to health technologies, environment, manufacturing, engineering and finance

[9] EasyBuild website: https://easybuild.io/

6. Lessons learned

When developing a new system architecture, challenges manifest at all levels, starting at the pure technical aspects related to the use of prototype components, and ending at the organisational and personnel-related challenges inherent to every large collaborative effort.

On the technical side, one major risk in hardware-related research is the dependency on external product-roadmaps, on which the DEEP consortium had very limited or no direct control. Here, it was of utmost importance to clearly identify all these dependencies and closely monitor the vendor roadmaps. For each dependency a mitigation plan was elaborated, including the identification of alternative technologies and clearly defining and communicating deadlines for the activation of an alternative plan. In the particular, case of prototype or pre-production hardware, it was crucial to have vendor expertise on board, and to clearly define beforehand the contributions and responsibilities from each partner, which must match their own product roadmaps and commercial interests, as well as the expectations of the overall consortium. Impact of hardware dependencies on software and application-related work was mitigated by providing alternative software development vehicles (SDV) at a very early stage. These SDVs were built using standard off-the-shelf components and enabled starting software-related work immediately while the prototype hardware was still in development.

Working with application developers, in general, it was hard to motivate them to adopt new architectures or programming paradigms. This resistance exists for a very good reason: hardware technologies have a life-cycle of about five years, while application codes grow over decades. Therefore, far-reaching modifications of the applications must be carefully considered. In this context, three elements are critical: (i) getting a very good understanding of the application characteristics and requirements and ensuring these are addressed in the hardware and software implementations; (ii), hiding most of the hardware complexity from the user behind the software stack and middleware, sticking to standard programming models (i.e. MPI, OpenMP), and reducing as much as possible the amount of changes required to the application codes themselves; and (iii), providing the application development teams with enough personnel resources to dedicate to the code adaptations, experiments, and benchmark campaigns.

Making a new system architecture usable is a task that relies mainly on the software environment. Hardware-specific and architecture-aware optimisations are implemented in the lower layers of the software stack and runtime environment (see section 3. Software environment), while the upper layers and programming interfaces towards the application are kept as unmodified as possible. In this context, it was found that some widely used middleware components lack important features necessary to fully exploit modularity. For example, heterogeneous jobs were only supported by Slurm starting from the 2019-release, and even now they are still treated as second-class jobs, insofar as they are only considered by the backfill phase of the scheduler, i.e., in the gaps left after scheduling non-heterogeneous jobs. This can lead to long waiting times for modular jobs. Better software strategies are also needed to improve the support for malleability and to facilitate a more dynamic sharing of compute resources. In conclusion, even if a good basis has been set to enable the utilisation of modular systems in particular, still a lot of work remains to fully exploit the architecture's potential. For this reason, software development for MSA continues and is intensified in the currently running DEEP-SEA and IO-SEA projects, focusing on improving support for heterogeneous resources and data management, respectively.

Making a new system architecture usable is a task that relies mainly on the software environment.

But even with good software support, application developers should be aware of the architecture characteristics to reach maximum performance. MSA requires them to consider a new layer in the system topology, between the node and the system level, namely the module level. In MSA-systems with different interconnects in each module, applications using two or more modules should be partitioned in a coarse-grained manner, in which most communications occur intra-module (instead of between modules). The reason is that the use of network gateways penalises performance. One important observation in this context was that application codes are rarely manufactured in such a manner, and reorganising them can be cumbersome. In response to this co-design insight, the most recent MSA systems (e.g., JUWELS, MeluXina) use the same interconnect technology for all modules, without gateways, which gives about the same latency and bandwidth for communication between any pair of nodes, independently of their position within the same or different modules. It should be noted, however, that this trend towards uniform networks does not render obsolete the low-level protocols developed to enable gateway-connections. On the contrary, this protocol sets the basis to enable the

adoption of new network technologies (e.g., next generations of Slingshot, or BXI) in an existing system, to evolve it over time in a sustainable manner, in which individual components can be exchanged without affecting the rest of the infrastructure.

Probably the most important lesson learned in the past years is the importance of building bridges between the different stakeholders in the HPC landscape. The multiple steps of integration for a highly complex set of hardware platforms, software layers, system stabilisation, and extensive benchmarking to ensure usability and performance have shown the critical importance of project planning, trust and good collaboration between the involved partners. The point of view, working philosophy, interests, and even the language used by hardware, software, and application developers are traditionally very different. Establishing a tight co-design strategy means finding a common language between all players and understanding and respecting the different perspectives and constraints, looking for an equilibrium between the occasionally conflicting expectations. It is crucial to establish trust and to strengthen the team-spirit between all partners, creating an atmosphere in which issues, difficulties and delays can be openly communicated and discussed. Only then a constructive, solution-targeted approach can take place. It is equally important to engage with the funding entities in the same open manner. Informing the funding authorities and taking them on-board immediately when risks manifest and affect the project (regardless if it is an R&D project or the deployment of a production system) is absolutely mandatory to build a trustful relationship between the project and the funding agency.

7. Future prospects

After one decade, the grounds of MSA are well established. A bright future is ahead with R&D projects, large-scale production systems, and the international community taking on the MSA principle.

Integration of European technology

The European HPC community, and the EuroHPC Joint Undertaking (EuroHPC JU) in particular, strongly invests in the promotion of HPC technology made in Europe, and as good example of this approach is the EU chips act[10,11]. The European Processor Initiative (EPI)[12] and the related pilot projects EUPilot[13] and EUPEX[14] are the spearhead in the design and development of processors in Europe. The EUPEX project, in particular, will build a hardware prototype using the Rhea chip: an Arm-based general purpose processor developed within EPI. The architecture of the EUPEX pilot follows the MSA principle. It will consist of two modules, a cluster and a booster. The cluster will employ Rhea as the main processing engine, while the booster will use Rhea as the host for some acceleration device, which could be GPUs or the RISC-V based accelerator also developed in EPI. Furthermore, EUPEX will employ the Bull eXascale Interconnect (BXI). Its third generation is being developed to include advanced features for dynamic routing and gateway mechanisms within the RED-SEA project[15]. With this design, the EUPEX pilot system will be the first hardware platform employing European IP in all its layers: system architecture, hardware components, software stack, and user-applications. It is therefore a very strong statement on the importance of European technology and a proof-of-concept for future pre-Exascale and Exascale platforms.

Quantum Computing

A prerequisite for the practical application of quantum computing in scientific computing is the integration of quantum computers (QC) into existing HPC infrastructures in order to enable the execution of quantum-classical hybrid computing models on the integrated HPC-QC infrastructure. In addition, various user communities of classical HPC (*e.g.*, in condensed matter physics, high-energy and plasma physics, materials science, biomolecular and bioinformatics researchers, among others) are expert in classical quantum simulations. Leveraging such expertise is essential to integrate user communities in the creation of application codes for practical quantum computing.

In December 2021, the EuroHPC JU “Pilot on quantum simulator” project HPCQS[16] started. The aim of this project is to prepare European research, industry and society for the use and federal operation of QCs and quantum simulators (QS). In HPCQS, a twin pilot system, consisting of two identical QSs, will be implemented and integrated at GENCI/CEA and FZJ/JSC, both hosts of European Tier-0 HPC systems. The programming and access platform for the QS is based on the European QLM system from Atos and uses the European MSA concept for the deep, low-latency integration with the HPC system. In this context, ParaStation Modulo will provide the runtime environment supporting the seamless execution of quantum-hybrid workflows. Therefore, it will be tightly integrated with the QLM on all levels of the system software stack, with a special focus on integrated resource management and scheduling. The HPCQS technology will be developed in a co-design process together with selected exemplary use cases from chemistry, physics, optimisation and machine learning suitable for quantum HPC hybrid calculations. HPCQS will provide cloud access and middleware for programming and execution of applications on the QS through the QLM. A Jupyter-Hub platform will guarantee safe access through the European UNICORE system to its ecosystem of quantum programming facilities and application libraries.

[10] https://ec.europa.eu/commission/presscorner/detail/en/IP_22_729
[11] https://www.kdt-ju.europa.eu/news/kdt-ju-become-chips-joint-undertaking
[12] EPI project website: https://www.european-processor-initiative.eu/
[13] EUPilot project website: https://www.eupilot.eu
[14] EUPEX project website: https://eupex.eu
[15] RED-SEA project website: https://redsea-project.eu/
[16] HPCQS project website: https://www.hpcqs.eu/

International impact

MSA is spreading across Europe, not only with the above described JUWELS and MeluXina: the pre-Exascale system LUMI[17], soon to be operational in Finland, couples various compute clusters in a modular fashion, and the upcoming Italian pre-Exascale platform Leonardo[18] is expected to be an MSA-platform, too.

Worldwide, HPC-centres are adopting MSA or very similar concepts. In the USA, the Perlmutter[19] system from NERSC (when completed) will present a CPU partition using AMD EPYC and a GPU partition using NVIDIA A100 devices, plus a small partition with large-memory nodes. All of these are interconnected in a Slingshot-10 network, making it possible to operate it as a modular system. Also, the Lawrence Livermore National Lab is following the concept of disaggregation, with subsets of the HPC system showing different node configurations, and enabling any mix of devices to be selected to best serve a given workload[20,21]. In Japan, the Wisteria/BDEC-01 platform[22] consists of two partitions, a simulation node group called Odyssey (with Fujitsu A64Fx CPUs) and a data analysis node group (aka Aquarius, with NVIDIA A100 GPUs). Although one cannot run single MPI jobs over the two Wisteria partitions, it is possible to run different workloads using the same job script, thanks to a new software platform called h3-Open-BDEC that facilitates the integration of simulation, data analysis and machine/deep learning. The so-called *heterogeneous flexible architecture*[23] foreseen for the Chinese Tianhe-3 system also strongly resembles a cluster-booster system as in an MSA.

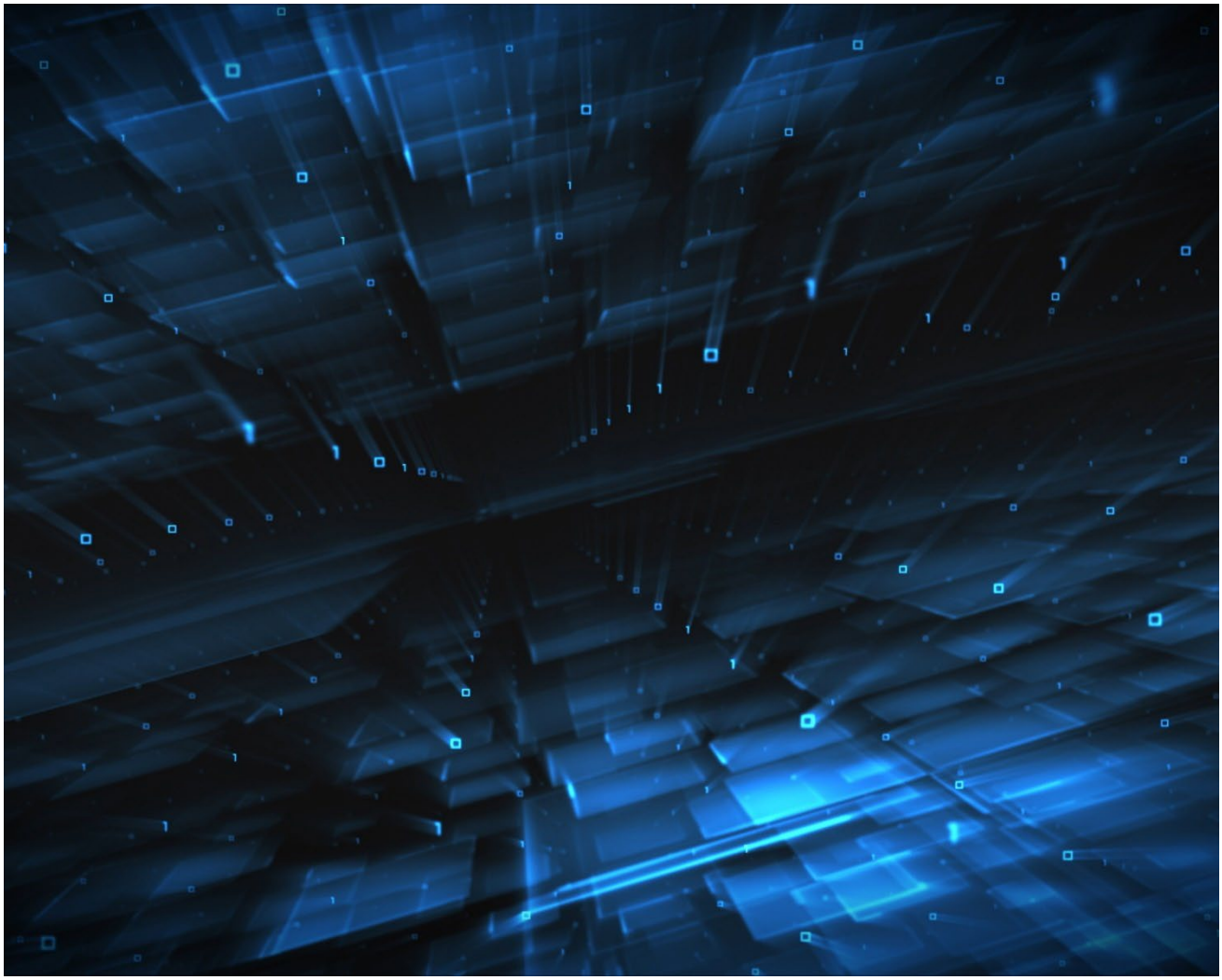
Exascale

The capabilities of MSA have been demonstrated in prototypes and are displayed in Petascale systems. The next step is now Exascale. Co-financed by the EuroHPC JU, two Exascale systems are expected to be deployed in Europe by 2024. Supercomputing centres currently apply as hosting sites; JSC is the representative of the German candidature from the Gauss Centre for Supercomputing (GCS) and brings MSA as the basis-concept for its candidature. If the proposal succeeds, at least one of the first European Exascale systems will be a modular platform. The system is expected to comprise at least a CPU-based cluster module and a GPU-based booster module. The cluster shall deliver a high memory bandwidth, while the booster provides at least 1 ExaFLOP/s computing performance in a very small energy envelope (under 20 MW). The system and its network topology will be conceived to enable the subsequent integration of future technology modules, including one based on quantum technology. With its modular design based on MSA, the Exascale system envisioned by JSC will put its compute resources at the service of a very wide range of European HPC user communities.

[17] LUMI system website: https://www.lumi-supercomputer.eu/
[18] Leonardo system website: https://www.cineca.it/index.php/en/hot-topics/Leonardo
[19] Perlmutter system website: https://docs.nersc.gov/systems/perlmutter/system_details/
[20] Interview with Bronis R. Supinski, LLNL CTO, in https://www.hpcwire.com/people-to-watch-2021/
[21] https://www.llnl.gov/news/ai-gets-boost-llnl-sambanova-collaboration
[22] https://www.hpcwire.com/2021/02/25/japan-to-debut-integrated-fujitsu-hpc-ai-supercomputer-this-spring
[23] Slide 32 in https://indico.cern.ch/event/764552/contributions/3432628/attachments/1869989/3076661/Chinas_Efforts_on_Supercomputing.pdf

Conclusions

The DEEP projects, with strong and continuous support from the European Commission, started ten years ago (in 2011) a research and development program in which over thirty European partners joined efforts to bring forward Modular Supercomputing Architecture. This concept, first received with scepticism by many, has in the meantime reached a high level of maturity, and is now taken over as a mainstream concept by HPC-centres across Europe and worldwide, with many large-scale production HPC-systems certifying it. At the same time, and to address the changing requirements of HPC users and take benefit of the latest technology trends, its further development continues within a series of initiatives, among which the SEA projects family (DEEP-SEA, IO-SEA, RED-SEA) and the pilot projects (EUPEX, HPCQS) shall be highlighted. The already achieved milestones and the bright future ahead, allows acknowledging the Modular Supercomputing Architecture as one of the major success stories of research and development in Europe.



Acknowledgements

- E. Suarez would like to acknowledge the contributions of the partners and individuals involved in the DEEP project series, ultimately responsible for the success history described in this white paper.
- The DEEP Projects have received funding from the European Commission's FP7, H2020, and EuroHPC JU Programmes, under Grant Agreements n° 287530 (DEEP), 610476 (DEEP-ER), 754304 (DEEP-EST), and 955606 (DEEP-SEA). The EuroHPC Joint Undertaking (JU) receives support from the European Union's Horizon 2020 research and innovation programme and Germany, France, Spain, Greece, Belgium, Sweden, United Kingdom, Switzerland.
- HPCQS has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No. 101018180. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, France, Italy, Ireland, Austria, Spain.
- The authors gratefully acknowledge the funding provided by the Helmholtz Programme Supercomputing & Big Data to realise the JURECA Cluster and Booster, as well as the project funding provided by the Ministry of Education and Research (BMBF) and the State of North Rhine-Westphalia for the procurement of the JUWELS Cluster and Booster (SiVeGCS).
- MeluXina: The acquisition and operation of the EuroHPC supercomputer is funded jointly by the EuroHPC Joint Undertaking, through the European Union's Connecting Europe Facility and the Horizon 2020 research and innovation program, as well as the Grand Duché du Luxembourg. This publication only reflects the authors' view and the EuroHPC Joint Undertaking is not responsible for any use that may be made of the information it contains.

References

[1] E. Suarez, N. Eicker and T. Lippert, Modular Supercomputing Architecture: from idea to production, Chapter 9 in Contemporary High Performance Computing: from Petascale toward Exascale, vol. 3, J. S. Vetter, Ed., CRC Press, 2019, pp. 223-251.

[2] E. Suarez, N. Eicker and T. Lippert, "Supercomputer Evolution at JSC," in *Proceedings of the 2018 NIC Symposium*, 2018.

[3] N. Eicker, T. Lippert, T. Moschny and E. Suarez, "The DEEP Project - An alternative approach to heterogeneous cluster-computing in the many-core era," *Concurrency and computation: Practice and Experience,* vol. 28, pp. 2394–-2411, 2016.

[4] A. Kreuzer, J. Amaya, N. Eicker, R. Léger and E. Suarez, "The DEEP-ER project: I/O and resiliency extensions for the Cluster-Booster architecture," in *Proceedings of 2018 IEEE 20th International Conference on High Performance Computing and Communications (HPCC)*, Exeter, United Kingdom, 2018.

[5] E. Suarez, A. Kreuzer, N. Eicker and T. Lippert, The DEEP-EST project, Chapter 1 in Porting applications to a Modular Supercomputer - Experiences from the DEEP-EST project, Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag, Schriften des Forschungszentrums Jülich IAS Series 48, 2021, pp. 9-25.

[6] N. Eicker, A. Galonska and M. N. J. Hauke, Bridging the DEEP Gap – Implementation of an Efficient Forwarding Protocol. In: Intel European Exascale Labs - Report 2013, 2014, p. 34–41.

[7] E. Suarez, N. Eicker, T. Moschny and T. Lippert, Critical Analysis of the Modular Supercomputing Architecture. Chapter 9 in Porting applications to a Modular Supercomputer - Experiences from the DEEP-EST project, Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag Jülich, IAS Series 48, 2021, pp. 233-245.

[8] W. Gropp, "MPICH2: a new start for MPI implementations," *Recent Advances in Parallel Virtual Machine and Message Passing Interface,* vol. 2474, p. 7, 2002.

[9] S. Pickartz, C. Clauss, S. Lankes, S. Krempel, T. Moschny and A. Monti, "Non- Intrusive Migration of MPI Processes in OS-bypass Networks," in *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2016.

[10] "InfiniBand (IB) architecture specification. Tech. rep., ITA," 2016. [Online]. Available: https://www.infinibandta.org/ibta-specification/.

[11] M. S. Birrittella, M. Debbage, R. Huggahalli, J. Kunz, T. Lovett, T. Rimmer, K. D. Underwood and R. C. Zak, "Intel® Omni-path Architecture: Enabling Scalable, High Performance Fabrics," in *2015 IEEE 23rd Annual Symposium on High-Performance Interconnects*, Santa Clara, CA, USA, 2015.

[12] H. Fröning, M. Nüssle, H. Litz, C. Leber and U. Brüning, "On achieving high message rates," in *13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, 2013.

[13] A. Kreuzer, E. Suarez, N. Eicker and T. Lippert, Porting applications to a Modular Supercomputer - Experiences from the DEEP-EST project, Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag Jülich, IAS Series 48, 2021, p. 254.

[14] Jülich Supercomputing Centre, «JURECA: General-purpose supercomputer at Jülich Supercomputing Centre,» *Journal of large-scale research facilities,* vol. 2, p. A62, 2016.

[15] Jülich Supercomputing Centre, "JURECA: Modular supercomputer at Jülich Supercomputing Centre," *Journal of large-scale research facilities,* vol. 4, p. A132, 2018.

[16] Jülich Supercomputing Centre, "JURECA: Data Centric and Booster Modules implementing the Modular Supercomputing Architecture at Jülich Supercomputing Centre," *Journal of large-scale research facilities,* vol. 7, p. A182, 2021.

[17] Jülich Supercomputing Centre, "JUWELS Cluster and Booster: Exascale Pathfinder with Modular Supercomputing Architecture at Jülich Supercomputing Centre," *Journal of large-scale research facilities,* vol. 7, p. A183, 2021.

[18] A. Herten, "JUWELS Booster - Early User Experiences," in *The 30th International Symposium on High-Performance Parallel and Distributed Computing, PERMAVOST Workshop, HPDC21*, Virtual, Sweden, 2021.

[19] S. Kesselheim, A. Herten and K. Krajsek, "JUWELS Booster – A Supercomputer for Large-Scale AI Research, High Performance Computing," in *ISC High Performance 2021*, Digital, Germany, 2021.

Authors:

Estela Suarez is Senior Scientist at the JSC, leader of the DEEP project series, and Professor for High Performance Computing at the Rhenish Friedrich Wilhelm University of Bonn.

Norbert Eicker leads the research group on cluster computing at JSC, and is Professor for Parallel Hard- und Software-Systems at the Bergische Universität Wuppertal.

Thomas Moschny is CTO of ParTec AG, and lead-architect of the ParaStation Modulo software stack.

Simon Pickartz works as a Research Engineer at ParTec AG and is mainly involved in the development of ParaStation MPI, the communication stack of ParaStation Modulo.

Carsten Clauss is an MPI Developer and Research Engineer at ParTec AG. From 2018 to 2020, he was Professor for Computer Science at the IUBH Internationale Hochschule.

Valentin Plugaru is the Chief Technology Officer of LuxProvide, home of the MeluXina Luxembourg - EuroHPC supercomputer, where he leads strategic and operational planning.

Andreas Herten is Senior Scientist at the JSC, where he leads the research group 'Accelerating Devices'. He is also the JSC-lead of the NVIDIA Application Lab at Jülich, a long-term collaboration between JSC and NVIDIA.

Kristel Michielsen leads the research group Quantum Information Processing (QIP) at the JSC and is Professor for QIP at RWTH Aachen University.

Thomas Lippert is Director of the Jülich Supercomputing Centre (JSC) from Forschungszentrum Jülich, and Professor for Modular Supercomputing und Quantum Computing at the Goethe-Universität Frankfurt.

Cite as: E. Suarez et al., Modular Supercomputing Architecture, ETP4HPC White Paper, 2022, doi 10.5281/zenodo.6508394

DOI: 10.5281/zenodo.6508394