The DEEP-SEA project: a software stack for heterogeneous and modular supercomputers

Estela Suarez [1,2], Norbert Eicker [1,3], and Hans-Christian Hoppe [1,4]

Abstract: Today's most powerful supercomputers achieve their performance through heterogeneous system architectures that integrate CPUs with accelerators, especially GPUs, and advanced multi-level memory systems. This hardware diversity challenges application developers to adapt legacy code, requiring significant efforts in code evolution and optimisation. The European DEEP-SEA project has developed an integrated software stack for heterogeneous HPC systems, including kernel modules, libraries, management systems and programming abstractions. It supports heterogeneous hardware configurations including modular supercomputers, enabling optimal resource allocation, application of malleability and programming model composability. Enhanced tools and data placement policies improved performance on DRAM and fast memory. Results were made publicly available, ensuring sustainability through integration with upstream open source projects and extension of HPC standards. This paper summarises the DEEP-SEA project's contributions to a wide variety of software packages and developments.

Keywords: High Performance Computing, Modular Supercomputing Architecture, Software Stack, DEEP-SEA, codesign

1 Introduction

High Performance Computing (HPC) infrastructures deliver increasing computing performance by steadily growing the size and capabilities of supercomputers [Pr24c, Ca22b]. First HPC systems exceeding one Exaflop/s performance are running in the USA [Oa24, Ar24] and Europe has committed to deploy two Exascale systems [Ju24, Eu24]. All disclosed Exascale architectures combine CPUs with accelerators (particularly GPUs) into heterogeneous platforms. Memory systems also seem to evolve towards multi-level memory hierarchies, which include DRAM, non-volatile and high-bandwidth memory layers [Si24, As21a, As21b]. This compute and memory heterogeneity puts a burden on application developers, who struggle adapting their legacy codes to these rapid hardware changes [Lü21]. Significant effort is needed to evolve and port application codes to heterogeneous HPC systems and optimise their performance [Su21]. An strategy to address this challenge and mitigate its impact is to adapt the system software so that its lower layers deal with the hardware heterogeneity, while its upper layers provide standardized and system-agnostic interfaces to the applications running on top.

1 Jülich Supercomputing Centre, Institute for Advanced Simulation, Forschungszentrum Jülich GmbH, Germany,
2 University of Bonn, Department of Computer Science, Germany,
3 University of Wuppertal, Faculty of Mathematics and Natural Sciences, Germany,
4 ParTec AG, Possartstr. 20, Munich, Germany,

For this purpose, in the USA, the Exascale Computing Project [Pr24a] included a concerted and cohesive effort to develop an optimised and efficient system software stack suitable for the full range of applications is required. Following the same objective, in Europe the DEEP-SEA project (*DEEP – Software for Exascale Architectures*) was set up to develop a software stack for heterogeneous HPC systems, especially those based on the Modular Supercomputing Architecture (MSA) [Pr24b, SEL19, Su22], such as the upcoming Exascale system JUPITER [Ju24].

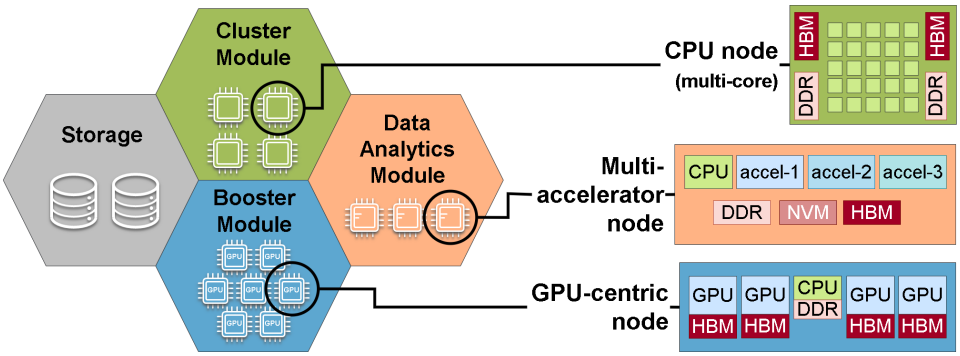

Fig. 1: Schematic of the Modular Supercomputing Architecture (MSA)

The MSA couples computer modules with different hardware configurations (CPU-only, GPU-accelerated, etc.), each designed to address the requirements of specific kinds or parts of applications (see Fig. 1). Thanks to an unified software stack, applications can run on any mix of resources, with communication capabilities within and across modules. Goal of the MSA system design is to efficiently organize hardware heterogeneity, achieving high performance for a large diversity of HPC applications.

The DEEP-SEA project was executed by a consortium of 14 European partners, under the lead of the Jülich Supercomputing Centre (JSC) from Forschungszentrum Jülich (FZJ). During its three years duration (2021 to 2024), the project developed an integrated software stack for European HPC systems. This stack supports heterogeneous compute, network, and memory configurations, enabling applications to run on diverse resources and optimizing code across various architectures.

2 Addressed Challenges

The DEEP-SEA project addresses the following challenges in the context of large-scale heterogeneous MSA systems:

- **Compute heterogeneity**: Integrating general-purpose CPUs with accelerators (GPUs or FPGAs) into compute modules within the MSA was already possible thanks to the work done within the DEEP project series [Co24]. DEEP-SEA further improved

the system software and programming tools for easier management and use of MSA systems.

- **Memory heterogeneity**: Processors combining DRAM with non-volatile memory and High-Bandwidth Memory (HBM) require intelligent data management to achieve maximum application performance. DEEP-SEA has developed data-placement tools and policies, and validated them with new, cutting edge systems. Furthermore, prototypical, innovative SW layers and tools for processing-in-memory were created.
- **Performance analysis and modelling**: Application performance and scalability on HPC systems with heterogeneous computing and memory configurations are difficult to achieve and predict. DEEP-SEA has enhanced best-of-breed tools to give application developers insights into how their codes run on different MSA modules, and advice on how to map their codes and workflows to achieve highest performance and efficiency. The novel *Optimisation Cycles* (OCs) guide SW developers and users through the steps needed to achieve their porting and optimisation goals.
- **Job scheduling and resource management**: Static scheduling policies tend to reserve resources longer than necessary, especially on heterogeneous platforms where applications do not always use all the resources of each node during their complete runtime. DEEP-SEA has enhanced scheduling and runtime software to enable dynamic allocation of execution threads and processes, and developed the application programming interfaces (APIs) required to achieve malleability.
- **Portability**: Hardware changes much faster than SW. Applications need to be able to run on different kinds of platforms without creating a multitude of diverging code branches that become very difficult to maintain. DEEP-SEA has developed and validated high-level programming models supporting a range of diverse processors/accelerators and providing standard APIs on which application developers can rely.
- **Composability**: Different programming models must be combined to effectively and efficiently use the diverse compute, memory and storage devices in a heterogeneous HPC system. DEEP-SEA has implemented interfaces that allow the collaboration of diverse programming models.

These issues have been addressed in DEEP-SEA through an integrated HPC and AI SW stack (see Fig. 2) that contains performance analysis, monitoring, and modelling tools (WP2), programming models at the node (WP4) and system (WP5) level, and management and system software (WP3). Ten co-design applications from seven key fields of science and research informed the SW design and implementation, were substantially enhanced during the project to be fit for Exascale systems, and served to test and validate the results.

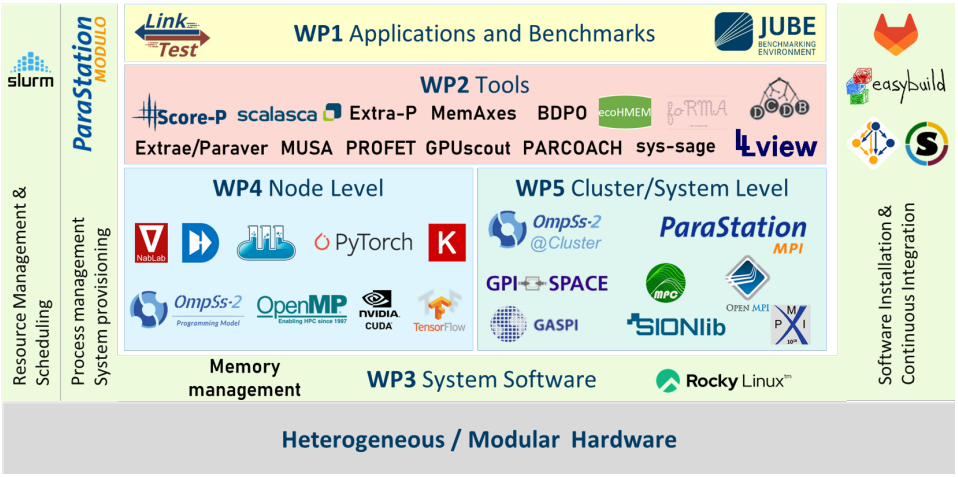

Fig. 2: Software Stack from the DEEP-SEA project

3 System Software and Programming Environments

The DEEP-SEA project has developed an integrated SW stack (see Fig. 2) consisting of low-level components, system and resource management software, plus runtime and programming environments at the node and system level. The DEEP-SEA software stack is documented in detail in the DEEP-SEA *Software specification* [SP21], in which each software component, their respective relations, interfaces, and dependencies are described. Use of a CI/CD framework ensures compatibility of the SW elements. Applications use the SW components, which provide the specific functions they need for a given objective.

To assist application developers and users in selecting and handling these, **Optimisation Cycles (OCs)** combine SW elements and tools. They define the process to analyse, modify and improve a given application to achieve a specific goal. For example, the *Memory System Performance Analysis OC* (see Fig. 3) uses an analysis tool to profile the memory allocations of an application, which provides input to a human developer to improve the data distribution across multiple levels of a memory hierarchy or to move to a different, more scalable or portable programming model. Such models might come with a run-time system capable of optimising the memory usage during execution.

From the system software perspective, the OCs are implemented by creating multi-step workflows that sequence use of SW tools and environments with complementary functionality. This requires the definition and implementation of interfaces between the tools and environments. To name some examples [SP21]: LLView and DCDB are combined in the *Monitoring OC*; Score-P and Scalasca are used in conjunction with ParaStation MPI [CME16] and Slurm [YJG03] in the *MSA-related OC*; and Extrae, HMem Advisor [Se17], FlexMalloc [Se], and MPC [PJN08] and OpenMP together form the *Memory Allocation OC*. Some

OCs are also combined to provide more advanced functionalities, e.g., the *MSA-related OC* and the *Monitoring OC* in combination with the Extra-P modelling tool [Wo16] create the *Mapping OC*.

Low-level communication: In DEEP-SEA, full MSA gateway support for the Multi-Processor Computing (MPC) environment [PJN08] has been implemented. This involves adapting the low-level communication layer from MPC and creating a new Point-to-point Management Layer (PML) for integration with Open MPI [GWS06]. Open MPI itself has also been extended with better support for collective operations when communicating across modules of an MSA system, as well as for varying core counts per node. ParaStation MPI's MSA awareness and its collective communication operations, which resulted from the previous DEEP projects [Be19a], have been further optimised in DEEP-SEA, and its low-level ParaStation Communication library was integrated with the GPI PGAS programming model [GS13].

Dynamic load balance: Mechanisms and policies for dynamic load balancing on a node have been implemented for OmpSs-2 [Ce24] and OpenMP via the Dynamic Load Balancing (DLB) library [Ga17], which optimises the use of compute cores by OmpSs or OpenMP tasks/threads within a node and improves throughput and, consequently, energy efficiency. As a result of the DEEP-SEA project, the upcoming OpenMP 6.0 standard will include provision for *free-agent threads*, which ensures integration of DLB with OpenMP implementations.

Performance portability: The OmpSs-2 tasking system has been extended to include a generic interface between its runtime and an LLVM-based compiler [Fo] developed in DEEP-SEA to support various hardware accelerators (GPUs, FPGAs, etc.). The two high-level programming models DaCe [Be19b] and NabLab [LOC18] were extended to support additional accelerators, for instance FPGAs, via a new backend. NabLab now uses DaCe as the backend for code generation, enabling straightforward use of all processors and accelerators supported by DaCe.

Composability and interoperability of programming models were improved by defining and implementing interfaces between different systems [Ja23]. This enables application developers to combine different approaches without having to entirely rewrite their codes to use just one of the programming models. Examples include enabling the combination of the PGAS model GPI-2 [IT] with ParaStation MPI with a shared, low-level communication layer and interoperability between the OmpSs-2@Cluster [Ag22] multi-node tasking system and MPI. ParaStation MPI was extended by support for the standardised PMIx process management layer [Ca18], which provides a clear boundary between communication middleware and resource managers.

Malleability: A prototype implementation was developed which supports dynamic shrinking and expanding of resources allocated to MPI or GPI-2 applications. Such resource changes can be triggered by the application (active malleability) or the runtime (passive malleability). For this purpose, ParaStation MPI was enhanced to work with PMIx and include support for MPI sessions [Hu22]. A MPI-level API for applications to handle malleability was defined with an immediate focus on active malleability (i.e., changes in resources triggered by the application). Using the SPANK plugin interface, Slurm was extended to support PMIx in a way sufficient for both active and passive malleability and for gracefully handling changes in nodes allocated to running jobs.

Resiliency: The trace-based Simgrid simulation framework [Ca01] was enhanced with a consistent failure model to study workload management for large systems in the presence of faults. Improvements include generating fault profiles beyond simple probabilistic laws, which is the first step towards creating more precise models, including locality, variability, ageing; such models will improve the prediction of failure probability and allow to adapt checkpointing intervals and reduce checkpointing overhead.

4 Performance Analysis and Modelling Tools

Existing SW tools for performance measurement, modelling and monitoring were improved and extended by new functionalities; examples are the MUSA tool [Gr16], which can now model multi-level memory hierarchies and support explicit OpenMP memory allocation routines, or the Extra-P modelling tool [Wo16], which can now compare the measured performance of each application function on traditional CPU clusters with their expected performance on GPUs, enabling users to decide which parts of their code are worth the porting effort. Once the code is ported, it can also automatically validate if the measured performance of each application function matches the expected/predicted performance on the GPU.

While this work on enhancing individual components is important, the most significant effect was achieved by combining them into OCs. Highlights include:

- **Mapping OC**: combines Scalasca [Ge10], Score-P [Kn12], and Extra-P [Wo16] to help application developers in deciding how to distribute their code over a heterogeneous MSA system [SP21]. Performance models of GPU- and CPU-based Booster and Cluster nodes were developed using hardware performance counter data and are used to easily compare application performance on these different modules and derive optimal application mappings. A new visual model comparison tool displays performance differences for parametrised models and applications.
- **Monitoring OC**: provides detailed measurements of the resource- and energy use of jobs running on an HPC system. These tools enable quantifying the impact of

different application mappings by monitoring application execution. Metrics measured by the DCDB framework [Ne19] were incorporated into LLview [Ced], which was completely reshaped to use a more generalised database. Also, DCDB now includes a new plugin for LIKWID [THW10] to access performance counters in a hardware-agnostic manner.

- **Energy OC**: relies on the BDPO tool [SM18] that transparently runs in the background of the allocated compute nodes and automatically optimises energy consumption by using hardware-exposed power control *knobs* to save energy. Applying this OC to several DEEP-SEA applications significantly reduced energy consumption with only minor impact on time-to-solution, and thus improved energy efficiency. Typical *reductions of energy-to-solution were in the range of 10% - 15%*.

5 Memory Management

Support for optimising use and management of multi-level memory hierarchies was significantly improved in the DEEP-SEA software stack. Several DEEP-SEA Optimisation Cycles (OCs) address this topic: the *Memory Management OC*, the *Memory System Performance Analysis OC*, the *Multi-level Simulation OC*, the *Analysis of Data Transfers and Memory Behaviour OC*, and the *Analysis and Debugging of MPI-RMA Communications OC* [SP21, Pi23, Ca22a].

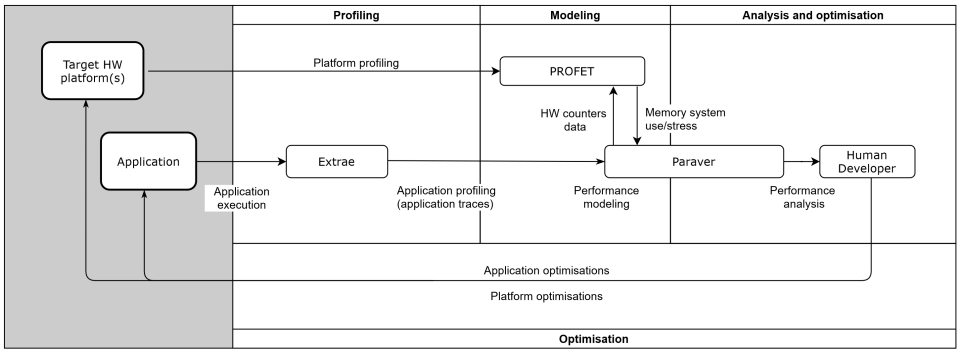

Fig. 3: Heterogenous Memory OC schematic

Performance analysis and modelling on hierarchical memory systems: Support for heterogeneous and deep memory hierarchies was implemented in multiple analysis and modelling tools (e.g., MUSA, PROFET [Ra19], ecoHMEM [Jo22], MemAxes [Gi18], Extrae, Paraver). As part of the *Memory System Performance Analysis OC* (see Fig. 3), the analytical model PROFET was integrated with the Extrae and Paraver performance analysis tools. PROFET can now process Extrae traces, and Paraver menus include the option to seamlessly launch a PROFET analysis. This enables users to understand and quantify the

impact of the use of different memory layers on the application performance, efficiency, and power consumption more easily. With this information, application developers can make better-informed decisions on how to distribute their data across heterogeneous memory technologies and deep memory hierarchies.

Low-level memory management: A sampling-based mechanism was implemented in the Linux kernel to track memory accesses over the lifetime of applications; an online controller is used to re-balance pages across memory tiers (e.g., DRAM and Non-Volatile Memory (NVM) or HBM) transparently to the application. In addition, the SHAMBLES memory profiler has been improved to simplify porting to more recent versions of the Linux kernel and other architectures, including ARM CPU architectures. The ecoHMEM framework optimises data placement across memory levels based on execution traces; it was refactored to become more modular and extensible, and now supports scaling up of measured object sizes, thus enabling application of ecoHMEM to large use cases.

Runtime: Multiple APIs and runtime extensions to exploit NUMA systems were published and included in the latest OmpSs-2 release, which can now automatically manage and optimise data transfers between host and accelerator memory spaces.

Programming environment: MPC was updated by support for the OpenMP 5.0 memory management interface, and the MPC-Allocator has been extended with support for HBM and NVM. The same interface is also used by GPI-2 to address NVM, which is an initial step towards persistence for GPI-2. Also, the high-level programming model DaCe now supports GPU memory spaces through CUDA's mallocAsync API.

Processing in Memory (PIM): Appropriate PIM-related data structures and APIs were implemented in the ZSim+DRAMsim3 simulation infrastructure [SK13, Li20, Ceb]. The PIM simulator was released as open source [Es24], and the performance, energy and behaviour of the PIM functions was evaluated. Also, the multi-level modelling tool MUSA has been extended with the ability to analyse PIM functions. Multiple target application domains, PIM functionalities, corresponding data structures, and APIs were identified and will inform future work on creating PIM systems.

6 Application Development and Co-design

Seven European HPC application fields (see Fig. 4) participated in DEEP-SEA, providing a total of 10 codes:

- **Space weather**: xPic [Kr18], AIDApy [LT19]

- **Weather forecast and climate**: IFS [Ba95], ecTrans [EC24]
- **Seismic imaging**: FRTM [Gr14], BSIT [Ha14]
- **Molecular dynamics**: GROMACS [Va05]
- **Computational fluid dynamics**: Neko [Ja24]
- **Neutron Monte-Carlo transport for nuclear energy**: PATMOS [CBC20]
- **Earth system modelling**: TSMP [Fu19]

These applications were selected to represent the typical multi-disciplinary usage of European HPC systems. They also cover different programming models and approaches and thus contributed to co-design, testing and validating the different areas of system software, programming models, and tools developed within DEEP-SEA.

Co-design: Characteristics and requirements of all applications were analysed at the beginning of the project and included in the initial co-design input [MK21]. The results were also used to identify specific application use cases to test the DEEP-SEA tools and programming models [MH22]. Once the requirements of a given application were known, it was necessary to identify the group of DEEP-SEA tools, programming environments and run-time systems that should be used to address them. To do this, we have defined the **Optimisation Cycles (OCs)**, which describe the sets of software components and steps for using them to achieve a specific analysis or optimisation objective [SP21]. As experience was gained in using the OCs, feedback was provided to their developers regarding user experience and suggestions for improvements.

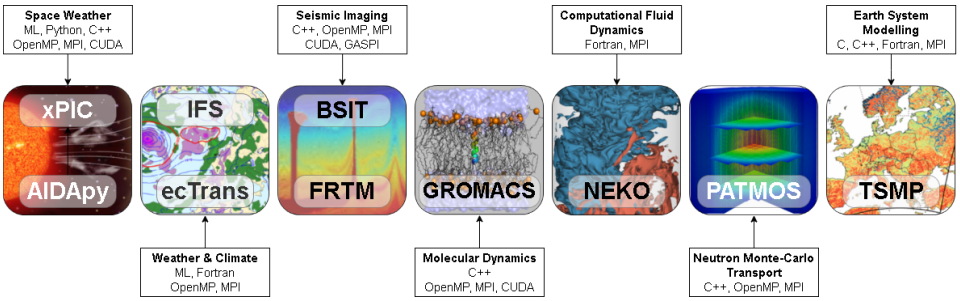

Fig. 4: DEEP-SEA is based on ten co-design applications from seven scientific fields

Application code adaptations and improvements: To identify performance limitations and bottlenecks, all application codes have been analysed in depth using *performance analysis and monitoring OCs* based on the Score-P/Scalasca [Ge10, Kn12], Extrae/Paraver [Cea], and LLview/DCDB [Ced, Ne19] tools. Performance portability of application codes was then improved by use on intra- and inter-node APIs for GPUs and CPUs (xPic, FRTM, PATMOS), by applying advanced dynamic load balancing (GROMACS, TSMP, xPic), or

by using high-level programming models such as DaCE (GROMACS). The *Energy OC* enabled several applications (xPic, AidaPy, IFS, GROMACS) to improve energy efficiency by optimising the processor operating points. Codes also explored the use of heterogeneous memory systems with the *Memory OCs* (BSIT, PATMOS) and malleability features (FRTM, TSMP).

Benchmarking: A benchmark suite composed of a wide range of synthetic benchmarks and selected co-design application use cases was integrated with the JUBE benchmarking environment [Fr10] and periodically run on the DEEP system [Cec] to measure its performance and identify potential variations. A visualisation environment for the results was put in place to simplify the interpretation of data.

7 Software Release

The DEEP-SEA project adopted a professional software development approach, relying on near-production software development platforms, continuous integration and deployment, as well as judicious validation and benchmarking. The DEEP system [Cec] was used as the central software development and integration platform in the project. It was upgraded with a uniform EDR InfiniBand network (100 Gbit/s bandwidth) and kept up-to-date with the latest software required for the development activities. Additional hardware added to the DEEP system included two Intel Xeon MAX servers that combine DRAM and HBM memory for validation of DEEP-SEA results on heterogeneous/multi-level memory hierarchies.

Continuous Integration and Continuous Deployment (CI/CD): The DEEP-SEA CI/CD infrastructure uses Gitlab repositories, the EasyBuild software deployment utility [cob] (for installation on the DEEP system), and JUBE for automatic benchmark-based verification of system performance. The generic GitLab Runner was replaced by Jacamar [Prd], which enables running CI jobs with user rights instead of requiring higher privileges. The CI infrastructure is set up and has been used to prepare the interim and final SW releases. With very few exceptions, all SW components are available as open source in EasyBuild or Spack [Prf] packages.

Software containers: Containerisation of HPC workloads has a high potential to simplify software installation and application execution. Security features to digitally sign Singularity containers were added to the DEEP-SEA container infrastructure to defend HPC systems and their users from unauthorised accesses and attacks. In DEEP-SEA, ParaStation Modulo has added support for the Slurm container start-up mechanism.

Public release: The final version of the DEEP-SEA software stack was released in two different ways: EasyBuild configuration files on a public FZJ Gitlab [pra], and Spack installation files at the LRZ/TUM repository [prb]. The EasyBuild configuration files list the dependencies and software modules required to install a specific SW package, the address of the repository where the latest release from the DEEP-SEA project was done, and the upstream repository in which the given SW component will continue to evolve. In this manner, interested users can take the version developed, tested and released by DEEP-SEA, or choose the latest generation of the given package, which contains the DEEP-SEA results and further improvements made beyond the DEEP-SEA lifetime. This shall ensure the future sustained use of the DEEP-SEA results across the European (and international) HPC landscape.

Sustainability of developments: The DEEP-SEA software stack relies on established European (or in a few cases, international) software packages that existed before the project started. They were enhanced with new features and interfaces that allow them to exploit the hardware heterogeneity of leading-edge HPC systems and cooperate with other European development activities. These improvements were pushed upstream and are now part of the official releases of each SW package. As owners of these products, the project partners will continue their future development through their own efforts and in future projects, ensuring that the work performed in DEEP-SEA and its results are sustained as part of the European Software Environment.

8 Summary and Future Prospects

In its three year term, the European project DEEP-SEA has made important contributions to the software stack of European HPC systems. These include low-level kernel modules, computation and communication libraries, system and resource management, and programming abstractions with associated runtime systems and tools. The DEEP-SEA software stack supports highly heterogeneous compute, network, and memory configurations, allowing applications to utilize any combination of resources according to their needs by mapping them to the Modular Supercomputing Architecture (MSA). The DEEP-SEA software stack supports reservation, allocation, and combination of heterogeneous resources, the application of malleability at the thread and MPI process level, and the composability of programming models. Data placement policies for emerging deep memory hierarchies were implemented on existing memory devices, and have been tested and validated on combinations of DRAM and fast memory to improve application performance on future systems like the SiPearl's line of ARM-based CPUs developed in the European Processor Initiative (EPI) [Si24]. A novel approach for the organization and use of the system software stack was introduced, which combines different sets of software in so-called *Optimisation Cycles (OCs)*, each one tailored towards a specific application or system performance target.

The project adopted a collaborative approach, working with ten real European applications from seven domain areas, driving the co-design and evaluation of the SW stack and demonstrating the benefits for European computing centres. DEEP-SEA managed dependencies between levels of the SW stack and introduced continuous integration/continuous deployment (CI/CD) techniques. At the end of the project, the DEEP-SEA software results were released in a public repository with all necessary installation packages. Care was also taken to include the developed code in the upstream branches of the open source projects used as the starting point, thereby ensuring the long-term sustainability of the project's contributions.

The R&D work around MSA continues beyond DEEP-SEA through the EuroHPC JU pilots EUPEX [prc] and HPCQS [pre], as well as through future initiatives at the European and national levels, such as the third generation of the PoP Center of Excellence [Coa]. The development roadmap initiated with the DEEP project in 2011 and continued until DEEP-SEA has led to the deployment of MSA pre-Exascale systems such as JUWELS, MeluXina, and Leonardo. It will culminate in the first European Exascale system, JUPITER, soon to be deployed at the Jülich Supercomputing Centre.

9 Acknowledgements

The work summarised in this paper is the result of the efforts of the many people who have been involved in the DEEP-SEA project. The authors would like to thank all members of DEEP-SEA for their contributions and commitment to the project objectives.

The DEEP Projects have received funding from the European Commission's FP7, H2020, and EuroHPC JU Programmes, under Grant Agreements nr. 287530, 610476, 754304, and 955606. The DEEP-SEA project receives also support from Belgium, France, Germany, Greece, Spain, Sweden, and Switzerland.

Bibliography

[Ag22] Aguilar Mena, Jimmy; Shaaban, Omar; Beltran, Vicenç; Carpenter, Paul; Ayguade, Eduard; Labarta Mancho, Jesus: OmpSs-2@Cluster: Distributed Memory Execution of Nested OpenMP-style Tasks. In: Euro-Par 2022: Parallel Processing: 28th International Conference on Parallel and Distributed Computing, Glasgow, UK, August 22–26, 2022, Proceedings. Springer-Verlag, Berlin, Heidelberg, p. 319–334, 2022.

[Ar24] Argonne National Laboratory: `https://www.alcf.anl.gov/aurora`, 2024. Accessed: 2024-06-16.

[As21a] Association, JEDEC Solid State Technology: DDR4 NVDIMM-P Bus Protocol Standard. `https://www.jedec.org/news/pressreleases/jedec-publishes-ddr4-nvdimm-p-bus-protocol-standard`, 2021.

[As21b] Association, JEDEC Solid State Technology: High Bandwidth Memory (HBM) DRAM. https://www.jedec.org/standards-documents/docs/jesd235a, 2021.

[Ba95] Barros, S.R.M.; Dent, D.; Isaksen, L.; Robinson, G.; Mozdzynski, G.; Wollenweber, F.: The IFS model: A parallel production weather code. Parallel Computing, 21(10):1621–1638, 1995. Climate and weather modeling.

[Be19a] Beltran, V.; Martinez, P.; Clauß, C.; Keller, K.; Bautista, L.; Roob, J.; Reh, P.; Montigny, L.; Steinbusch, B.: DEEP-EST deliverable D6.3: Complete Programming Environment Implementation. https://deep-projects.eu/wp-content/uploads/2023/09/DEEP-EST_D63_Complete_Programming_Environment_Impl_v10.pdf, 2019. Accessed: 2024-07-16.

[Be19b] Ben-Nun, Tal; Matteis, Tiziano De; Rausch, Oliver; Johnsen, Carl; Raje, Saurabh; Kuster, Andreas; Schaad, Philipp; Burger, Manuel; Walo, Neville; Lavarini, Luca; Scholbe, Stefan; Hofer, Dominic; Trümper, Lukas; Ivanov, Andrei; Gavrilas, Gabriel; Baumann, Thomas; Ates, Berke; Simmonds, Benjamin; Huetter, Noah; Kleine, Jan; Widmer, Marc; Schneider, Timo; Hu, Tom; Deconinck, Florian; Thaler, Felix; Dahm, Johann; Ratsimbazafy, Mamy; Jacob, Simon; Thierry, Backes; Ehrengruber, Till; Anklin, Valentin; USDOE: DaCe - Data Centric Parallel Programming, 5 2019.

[Ca01] Casanova, Henri: Simgrid: A toolkit for the simulation of application scheduling. In: Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid. IEEE, pp. 430–437, 2001.

[Ca18] Castain, Ralph H.; Hursey, Joshua; Bouteiller, Aurelien; Solt, David: PMIx: Process management for exascale environments. Parallel Computing, 79:9–29, 2018.

[Ca22a] Carpenter, P.; Marazakis, M.; Chazapis, A.; Malliaroudakis, E.; Asiminakis, M.; Mavridis, S.; Pavlidakis, M.; Peña, A.J.; Elshazly, H.; Jaeger, J.; Meynard, R.; Roussel, A.; Lemarinier, P.; Radojkovic, P.; Ridley, T.; Beltran, V.; Lopez, V.; Chasapis, D.; Schneider, T.; Andersson, M. I.; Karp, M.: DEEP-SEA deliverable D4.1: Initial node-level programming environment. https://deep-projects.eu/wp-content/uploads/2023/09/DEEP-SEA_D4.1_Initial-node-level-programming-environment_Published.pdf, 2022.

[Ca22b] Carpenter, Paul; Haus, Utz-Uwe; Laure, Erwin; Narasimhamurthy, Sai; Suarez, Estela: Heterogeneous High-Performance Computing. ETP4HPC White Paper, 2022.

[CBC20] Chang, Tao; Brun, Emeric; Calvin, Christophe: Portable Monte Carlo Transport Performance Evaluation in the PATMOS Prototype. In (Wyrzykowski, Roman; Deelman, Ewa; Dongarra, Jack; Karczewski, Konrad, eds): Parallel Processing and Applied Mathematics. Springer International Publishing, Cham, pp. 528–539, 2020.

[Cea] Center, Barcelona Supercomputing: BSC Performancea analsysis tools Extrae/Paraver. https://tools.bsc.es. Accessed: 2024-07-16.

[Ceb] Center, Barcelona Supercomputing: zsim+DRAMsim3 simulation infrastructure. https://github.com/bsc-mem/zsim/tree/zsim+DRAMsim3+ACM. Accessed: 2024-07-17.

[Cec] Centre, Juelich Supercomputing: DEEP System. https://www.fz-juelich.de/de/ias/jsc/systeme/prototypsysteme/deep-system. Accessed: 2024-07-16.

[Ced] Centre, Juelich Supercomputing: LLview monitoring. https://apps.fz-juelich.de/jsc/llview/docu. Accessed: 2024-07-16.

[Ce24] Center, Barcelona Supercomputing: OmpSs-2 programming model. https://pm.bsc.es/ompss-2, 2024. Accessed: 2024-07-16.

[CME16] Clauss, Carsten; Moschny, Thomas; Eicker, Norbert: Dynamic process management with allocation-internal co-scheduling towards interactive supercomputing. In: Proceedings of the 1st COSH Workshop on Co-Scheduling of HPC Applications. p. 13, 2016.

[Coa] CoE, PoP: The Performance Optimisation and Productivity Center of Excellence. https://pop-coe.eu. Accessed: 2024-07-17.

[cob] community, EasyBuild: EasyBuild software build and installation framework. https://easybuild.io/. Accessed: 2024-07-17.

[Co24] Consortium, DEEP-SEA: https://deep-projects.eu, 2024. Accessed: 2024-06-16.

[EC24] ECMWF: https://github.com/ecmwf-ifs/ectrans, 2024. Accessed: 2024-07-22.

[Es24] Esmaili-Dokht, Pouya; Guiot, Miquel; Radojković, Petar; Martorell, Xavier; Ayguadé, Eduard; Labarta, Jesus; Adlard, Jason; Amato, Paolo; Sforzin, Marco: $O(n)$O(n) Key–Value Sort With Active Compute Memory. IEEE Transactions on Computers, 73(5):1341–1356, 2024.

[Eu24] EuroHPC Joint Undertaking: https://eurohpc-ju.europa.eu/jules-verne-consortium-will-host-new-eurohpc-exascale-supercomputer-france-2023-06-20_en, 2024. Accessed: 2024-06-16.

[Fo] Foundation, LLVM: LLVM Compiler Infrastructure. https://llvm.org. Accessed: 2024-07-16.

[Fr10] Frings, W.; Schnurpfeil, A.; Meier, S.; Janetzko, F.; Arnold, L.: A Flexible, Application- and Platform-Independent Environment for Benchmarking. 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. 423 - 430. 2010. Record converted from VDB: 12.11.2012.

[Fu19] Furusho-Percot, Carina; Görgen, Klaus; Hartick, Carl; Kulkarni, Ketan; Keune, Jessica; Kollet, Stefan: Pan-European groundwater to atmosphere terrestrial systems climatology from a physically consistent simulation. Scientific data, 6(1):320, 2019.

[Ga17] Gasulla, Marta Garcia: Dynamic load balancing for hybrid applications. PhD thesis, Universitat Politècnica de Catalunya. Departament d'Arquitectura de Computadors, 2017.

[Ge10] Geimer, Markus; Wolf, Felix; Wylie, Brian J. N.; Ábrahám, Erika; Becker, Daniel; Mohr, Bernd: The Scalasca performance toolset architecture. Concurrency and Computation: Practice and Experience, 22(6):702–719, 2010.

[Gi18] Giménez, Alfredo; Gamblin, Todd; Jusufi, Ilir; Bhatele, Abhinav; Schulz, Martin; Bremer, Peer-Timo; Hamann, Bernd: MemAxes: Visualization and Analytics for Characterizing Complex Memory Performance Behaviors. IEEE Transactions on Visualization and Computer Graphics, 24(7):2180–2193, 2018.

[Gr14] Gruenewald, D.; Ettrich, N.; Rahn, M.; Pfreundt, F.J.: FRTM - A Productive Framework for Reverse Time Migration. 2014.

[Gr16] Grass, Thomas; Allande, Cesar; Armejach, Adria; Rico Carro, Alejandro; Ayguade, Eduard; Labarta, Jesús; Valero, Mateo; Casas, Marc; Moreto, Miquel: MUSA: A Multi-level Simulation Approach for Next-Generation HPC Machines. In: SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 526–537, 11 2016.

[GS13] Grünewald, Daniel; Simmendinger, Christian: The GASPI API specification and its implementation GPI 2.0. In: 7th International Conference on PGAS Programming Models. volume 243, p. 52, 2013.

[GWS06] Graham, Richard L.; Woodall, Timothy S.; Squyres, Jeffrey M.: Open MPI: A Flexible High Performance MPI. In (Wyrzykowski, Roman; Dongarra, Jack; Meyer, Norbert; Waśniewski, Jerzy, eds): Parallel Processing and Applied Mathematics. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 228–239, 2006.

[Ha14] Hanzich, M.; Kormann, J.; Gutierrez, N.; Rodriguez, J.E.; de la Puente, J.; Cela, J.M.: Developing Full Waveform Inversion Using HPC Frameworks: BSIT. 2014.

[Hu22] Huber, Dominik; Streubel, Maximilian; Comprés, Isaías; Schulz, Martin; Schreiber, Martin; Pritchard, Howard: Towards Dynamic Resource Management with MPI Sessions and PMIx. In: Proceedings of the 29th European MPI Users' Group Meeting. EuroMPI/USA '22, Association for Computing Machinery, New York, NY, USA, p. 57–67, 2022.

[IT] ITWM, Fraunhoger: GPI-2 Programming Next Generation Supercomputers. http://www.gpi-site.com/. Accessed: 2024-07-16.

[Ja23] Jaeger, J.; Clauss, C.; Dehenne, R.; Happ, S.; Katevenis, G.; Mavridis, S.; Hidri, K.; Marazakis, M.; Lemarinier, P.; Comprés, I.: DEEP-SEA deliverable D5.5: Resource management and tool interface. `https://deep-projects.eu/resources/deliverables/#project_9`, 2023.

[Ja24] Jansson, Niclas; Karp, Martin; Podobas, Artur; Markidis, Stefano; Schlatter, Philipp: Neko: A modern, portable, and scalable framework for high-fidelity computational fluid dynamics. Computers & Fluids, 275:106243, 2024.

[Jo22] Jordà, Marc; Rai, Siddharth; Ayguadé, Eduard; Labarta, Jesús; Peña, Antonio J.: ecoHMEM: Improving Object Placement Methodology for Hybrid Memory Systems in HPC. In: 2022 IEEE International Conference on Cluster Computing (CLUSTER). pp. 278–288, 2022.

[Ju24] Juelich Supercomputing Centre: `https://www.fz-juelich.de/en/ias/jsc/jupiter`, 2024. Accessed: 2024-06-16.

[Kn12] Knüpfer, Andreas; Rössel, Christian; Mey, Dieter an; Biersdorff, Scott; Diethelm, Kai; Eschweiler, Dominic; Geimer, Markus; Gerndt, Michael; Lorenz, Daniel; Malony, Allen; Nagel, Wolfgang E.; Oleynik, Yury; Philippen, Peter; Saviankou, Pavel; Schmidl, Dirk; Shende, Sameer; Tschüter, Ronny; Wagner, Michael; Wesarg, Bert; Wolf, Felix: Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope,Scalasca, TAU, and Vampir. In (Brunst, Holger; Müller, Matthias S.; Nagel, Wolfgang E.; Resch, Michael M., eds): Tools for High Performance Computing 2011. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 79–91, 2012.

[Kr18] Kreuzer, Anke; Eicker, Norbert; Amaya, Jorge; Suarez, Estela: Application Performance on a Cluster-Booster System. In: 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). pp. 69–78, 2018.

[Li20] Li, Shang; Yang, Zhiyuan; Reddy, Dhiraj; Srivastava, Ankur; Jacob, Bruce: DRAMsim3: A Cycle-Accurate, Thermal-Capable DRAM Simulator. IEEE Computer Architecture Letters, 19(2):106–109, 2020.

[LOC18] Lelandais, Benoît; Oudot, Marie-Pierre; Combemale, Benoit: Fostering metamodels and grammars within a dedicated environment for HPC: the NabLab environment (tool demo). In: Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering. SLE 2018, Association for Computing Machinery, New York, NY, USA, p. 200–204, 2018.

[LT19] Lapenta, Giovanni; Team, AIDA H2020: Processing big data from space missions and massively parallel simulations within the Horizon 2020 Project AIDA, 2019.

[Lü21] Lührs, Sebastian; Dick, Björn; Johnson, Chris; Marsella, Luca; Mathias, Gerald; Morales, Cristian; Axner, Lilit; Shamakina, Anastasiia; Shoukourian, Hayk: Best Practice Guide - Application porting and code-optimization activities for European HPC systems, April 2021.

[MH22] Meinke, J.H.; Holicki, M.E.: DEEP-SEA deliverable D1.2: Application use cases and traces. `https://deep-projects.eu/wp-content/uploads/2023/09/DEEP-SEA_D1.2_Application-use-cases-and-traces_Published.pdf`, 2022. Accessed: 2024-07-16.

[MK21] Meinke, J.H.; Kreuzer, A.: DEEP-SEA deliverable D1.1: Initial application co-design input. `https://deep-projects.eu/wp-content/uploads/2023/09/DEEP-SEA_D1.1_Initial-application-co-design-input_Published.pdf`, 2021. Accessed: 2024-07-16.

[Ne19] Netti, Alessio; Müller, Micha; Auweter, Axel; Guillen, Carla; Ott, Michael; Tafani, Daniele; Schulz, Martin: From facility to application sensor data: modular, continuous and holistic monitoring with DCDB. In: SC '19: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '19, Association for Computing Machinery, New York, NY, USA, 2019.

[Oa24] Oak Ridge National Laboratory: `https://www.olcf.ornl.gov/frontier`, 2024. Accessed: 2024-06-16.

[Pi23] Pickartz, S.; Marazakis, M.; Bishnoi, A.; na, A. J. Pe Roussel, A.; Clauss, C.; Kanellou, E.; na, I. A. Comprés Ure Jaeger, J.; Hidri, K.; Asiminakis, M.; Ploumidis, M.; Rauh, M.; Kossyfidis, N.; Lemarinier, P.; Petrakis, P.; Happ, S.; Iserte, S.; Krempel, S.; Robert, R.; Rotaru, T.; Huang, X.: DEEP-SEA deliverable D3.2: Complete system software implementation. `https://deep-projects.eu/resources/deliverables/#project_9`, 2023.

[PJN08] Pérache, Marc; Jourdren, Hervé; Namyst, Raymond: MPC: A unified parallel runtime for clusters of NUMA machines. In: Euro-Par 2008–Parallel Processing: 14th International Euro-Par Conference, Las Palmas de Gran Canaria, Spain, August 26-29, 2008. Proceedings 14. Springer, pp. 78–88, 2008.

[pra] project, DEEP-SEA: EasyBuild recipes for public release of DEEP-SEA Software. `https://gitlab.jsc.fz-juelich.de/deep-sea/easybuild-repository-public-release`. Accessed: 2024-07-17.

[prb] project, DEEP-SEA: Spack recipes for public release of DEEP-SEA Software. `https://gitlab.lrz.de/deep-sea/spack`. Accessed: 2024-07-17.

[prc] project, EUPEX: The EUPEX project. https://eupex.eu. 2024-07-17.

[Prd] Project, Exascale Computing: Jacamar CI/CD driver. https://gitlab.com/ecp-ci/jacamar-ci. Accessed: 2024-07-17.

[pre] project, HPCQS: The HPCQS project. https://www.hpcqs.eu. Accessed: 2024-07-17.

[Prf] Project, Spack: Spack package manager. https://spack.io. Accessed: 2024-07-17.

[Pr24a] Project, Exascale Computing: https://www.exascaleproject.org, 2024. Accessed: 2024-06-16.

[Pr24b] Projects, SEA: https://sea-projects.eu, 2024.

[Pr24c] Prometeus GmbH: http://www.top500.org, 2024. Accessed: 2024-06-16.

[Ra19] Radulovic, Milan; Sánchez Verdejo, Rommel; Carpenter, Paul; Radojković, Petar; Jacob, Bruce; Ayguadé, Eduard: PROFET: Modeling System Performance and Energy Without Simulating the CPU. 3(2), 2019.

[Se] Servat, Harold: Flexible memory allocation tool for multi-tiered memory systems. https://github.com/intel/flexmalloc. Accessed: 2024-07-16.

[Se17] Servat, Harald; Peña, Antonio J.; Llort, Germán; Mercadal, Estanislao; Hoppe, Hans-Christian; Labarta, Jesús: Automating the Application Data Placement in Hybrid Memory Systems. In: 2017 IEEE International Conference on Cluster Computing (CLUSTER). pp. 126–136, 2017.

[SEL19] Suarez, Estela; Eicker, Norbert; Lippert, Thomas: Modular Supercomputing Architecture: from Idea to Production; 3rd. In: Contemporary High Performance Computing: From Petascale toward Exascale, Volume 3, volume 3. CRC Press, FL, USA, pp. 223–251, 2019.

[Si24] SiPEARL: https://sipearl.com/wp-content/uploads/2024/05/PR_SiPearl_Rhea1_EN.pdf, 2024. Accessed: 2024-06-16.

[SK13] Sanchez, Daniel; Kozyrakis, Christos: ZSim: fast and accurate microarchitectural simulation of thousand-core systems. In: Proceedings of the 40th Annual International Symposium on Computer Architecture. ISCA '13, Association for Computing Machinery, New York, NY, USA, p. 475–486, 2013.

[SM18] Stoffel, Mathieu; Mazouz, Abdelhafid: Improving Power Efficiency Through Fine-Grain Performance Monitoring in HPC Clusters. In: 2018 IEEE International Conference on Cluster Computing (CLUSTER). pp. 552–561, 2018.

[SP21] Simon Pickartz, Manolis Marazakis, Norbert Eicker: DEEP-SEA deliverable D3.1: Software specification. https://deep-projects.eu/wp-content/uploads/2023/09/DEEP-SEA_D3.1_Software-specification_Published.pdf, 2021. Accessed: 2024-07-16.

[Su21] Suarez, Estela; Kreuzer, Anke; Eicker, Norbert; Lippert, Thomas: The DEEP-EST project. In: Porting applications to a Modular Supercomputer - Experiences from the DEEP-EST project, volume 48 of Schriften des Forschungszentrums Jülich IAS Series. Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag, Jülich, pp. 9–25, 2021.

[Su22] Suarez, Estela; Eicker, Norbert; Moschny, Thomas; Pickartz, Simon; Clauss, Carsten; Plugaru, Valentin; Herten, Andreas; Michielsen, Kristel; Lippert, Thomas: Modular Supercomputing Architecture – A Success Story of European R&D. ETP4HPC White Paper, 2022.

[THW10] Treibig, Jan; Hager, Georg; Wellein, Gerhard: LIKWID: A Lightweight Performance-Oriented Tool Suite for x86 Multicore Environments. In: 2010 39th International Conference on Parallel Processing Workshops. pp. 207–216, 2010.

[Va05] Van Der Spoel, David; Lindahl, Erik; Hess, Berk; Groenhof, Gerrit; Mark, Alan E.; Berendsen, Herman J. C.: GROMACS: Fast, flexible, and free. Journal of Computational Chemistry, 26(16):1701–1718, 2005.

[Wo16] Wolf, Felix; Bischof, Christian; Calotoiu, Alexandru; Hoefler, Torsten; Iwainsky, Christian; Kwasniewski, Grzegorz; Mohr, Bernd; Shudler, Sergei; Strube, Alexandre; Vogel, Andreas; Wittum, Gabriel: Automatic Performance Modeling of HPC Applications. In (Bungartz, Hans-Joachim; Neumann, Philipp; Nagel, Wolfgang E., eds): Software for Exascale Computing - SPPEXA 2013-2015. Springer International Publishing, Cham, pp. 445–465, 2016.

[YJG03] Yoo, Andy B.; Jette, Morris A.; Grondona, Mark: SLURM: Simple Linux Utility for Resource Management. In (Feitelson, Dror; Rudolph, Larry; Schwiegelshohn, Uwe, eds): Job Scheduling Strategies for Parallel Processing. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 44–60, 2003.