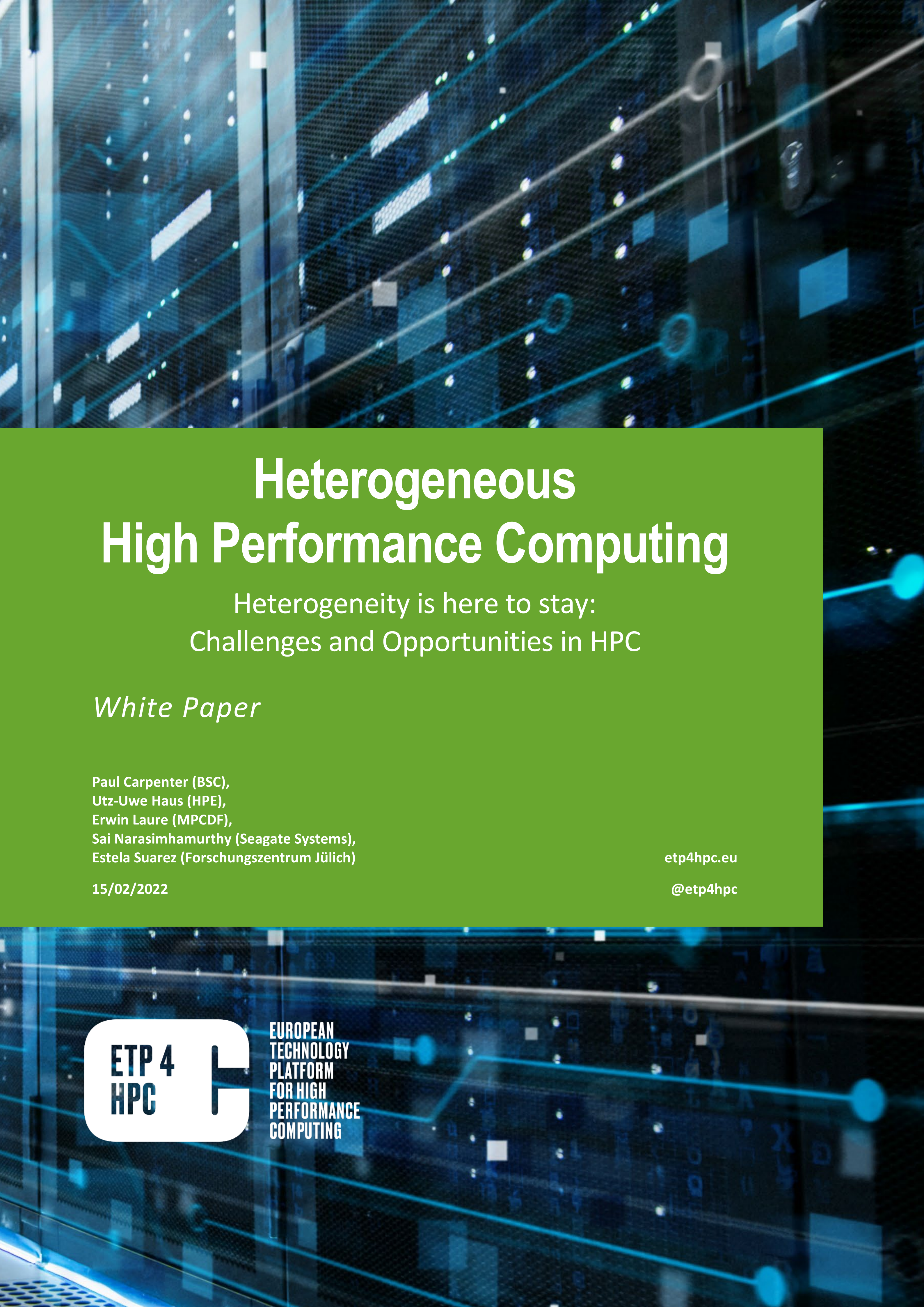


Heterogeneous High Performance Computing

Heterogeneity is here to stay:
Challenges and Opportunities in HPC

White Paper

Paul Carpenter (BSC),
Utz-Uwe Haus (HPE),
Erwin Laure (MPCDF),
Sai Narasimhamurthy (Seagate Systems),
Estela Suarez (Forschungszentrum Jülich)

etp4hpc.eu

15/02/2022

@etp4hpc

EUROPEAN TECHNOLOGY PLATFORM FOR HIGH PERFORMANCE COMPUTING

Table of Contents

Introduction

Modern HPC systems are becoming increasingly heterogeneous, affecting all components of HPC systems, from the processing units, through memory hierarchies and network components to storage systems. This trend is on the one hand due to the need to build larger, yet more energy efficient systems, and on the other hand it is caused by the need to optimise (parts of the) systems for certain workloads. In fact, it is not only the systems themselves that are becoming more heterogeneous, but also scientific and industrial applications are increasingly combining different technologies into complex workflows, including simulation, data analytics, visualisation, and artificial intelligence/machine learning. Different steps in these workflows call for different hardware and thus today's HPC systems are often composed of different modules optimised to suit certain stages in these workflows.

While the trend towards heterogeneity is certainly helpful in many aspects, it makes the task of programming these systems and using them efficiently much more complicated. Often, a combination of different programming models is required and selecting suitable technologies for certain tasks or even parts of an algorithm is difficult. Novel methods might be needed for heterogeneous components or be only facilitated by them. And this trend is continuing, with new technologies around the corner that will further increase heterogeneity, e.g. neuromorphic or quantum accelerators, in-memory-computing, and other non-von-Neumann approaches.

In this paper, we present an overview of the different levels of heterogeneity we find in HPC technologies and provide recommendations for research directions to help deal with the challenges they pose. We also point out opportunities that particularly applications can profit from by exploiting these technologies. Research efforts will be needed over the full spectrum, from system architecture, compilers and programming models/languages, to runtime systems, algorithms and novel mathematical approaches.

Key insights

- Heterogeneity is here to stay, and the upcoming disruptive technologies (e.g. neuromorphic, quantum, processing in memory) will only increase it.
- Dynamic orchestration and management of heterogeneous components are key for effective resource utilisation.
- Applications need appropriate tools (standardised programming models; smart compilers, runtime and workflow systems; debugging and performance tools) to master the complexity of heterogeneous systems.
- Novel heterogeneous systems provide opportunities for novel workflows, novel mathematical formulations and new application features.
- Heterogeneity cannot be tackled in an independent manner but coordinated efforts at different levels need to be combined (e.g. compilers and tools and runtime systems)

Key recommendations

- The challenge of heterogeneity must be faced as a community effort, sharing best practises to combat complexity.
- Heterogeneity has to be tackled at all levels of the stack: from the system architecture, (choosing an appropriate distribution of the resources) passing by the middleware and software stack (to manage and orchestrate the resources), up to the application codes (adapting them to use the various resources).
- Integration projects hardening and combining results covering all levels of the HPC ecosystem are needed
- Interoperability and exchangeability of components (hardware and software) should be improved and become part of the overall design

Today’s HPC systems exhibit heterogeneity everywhere - from the compute and storage subsystems, through integrated systems, to systems software and even application software. In this Chapter, we provide an overview of the current situation, point out some potential future directions and highlight future research directions.

1. Processing Technologies

Multiple different processing technologies are today combined in HPC systems in order to achieve the required performance and functionality at an affordable power envelope. The most frequently used components are general purpose processors (CPUs) and graphic cards (GPUs), but others are also used (e.g. field programmable gate arrays (FPGAs), domain-specific accelerators) and, on experimental basis, even more exotic ones like neuromorphic and quantum devices. This hardware diversity comes at the price of a more complex software and programming environment, and the need to adapt the applications to exploit the full computational power of heterogeneous HPC systems.

CPUs

Arguably, **CPUs** are currently the best devices in terms of programmability, with a long history of usage and very wide and stable software support that facilitates running applications across different hardware generations and even with different CPU architectures porting applications is typically quite straight-forward (e.g. from x86 to ARM CPUs). However, modern CPUs have become much more complex than they used to be in the past. The end of Dennard’s law [1] forced CPUs to become multi-core, for which cache coherency protocols and parallel programming techniques are required. Following the RISC architecture principle, the CPU pipelines are now deeper and apply multiple optimisation strategies for which specific hardware elements inside the CPU have been added. Examples include arithmetic-logic units to execute different kinds of operations, memory prefetchers, and registers for vector operations with progressively larger and even variable vector lengths. All these added capabilities bring a higher peak performance on the CPU but require specific optimisations such as data alignment and vectorisation in order to actually benefit from them. Some of these optimisations can be achieved by compilers and runtime systems, but often applications also need to be modified to best profit from them. The CPUs used in HPC today have become very heterogeneous themselves and this trend is likely to continue. A particular type of CPUs are **many-core devices**, which were designed as CPUs containing a large number of relatively weak cores, originally conceived as accelerators, i.e. PCIe-attached cards that depend on a stronger host-CPU to boot and communicate through the network. However, the distinction between multi-core CPUs and many-core devices has almost diluted, since the latter became more and more autonomous and the former today can contain over 100 cores; and with the end of Intel’s Xeon PHI [2] products, there are currently no wide-spread many-core devices on the market.

Graphics Processing Units (GPUs)

Graphics Processing Units (GPUs) were originally designed for the gaming and film industries to speed-up graphics operations, but are used today also as computing devices by arithmetic-intensive HPC and Artificial Intelligence (AI) applications. High-end GPUs contain a very large number of arithmetic-logic units supporting various precisions (double, single, half precision) and specific tensor cores used to perform very efficiently vector and matrix operations widely used in machine- and deep learning algorithms. Furthermore, they contain high-bandwidth memory (HBM) technologies that are very interesting for memory-bound applications. To use GPUs, accelerator programming models such as OpenACC [3] and OpenCL [4] are required, some of them vendor specific such as CUDA or HIP, which hampers application portability and particularly performance portability across platforms. There are ongoing efforts to include vendor-independent GPU support in other programming environments, e.g. OpenMP [5] or even C++, but so far, the best performance is still achieved using the vendor specific environments. Nevertheless, their many execution units and the high memory bandwidth make GPUs very computationally powerful and energy efficient devices, reaching peak performances an order of magnitude higher than CPUs.

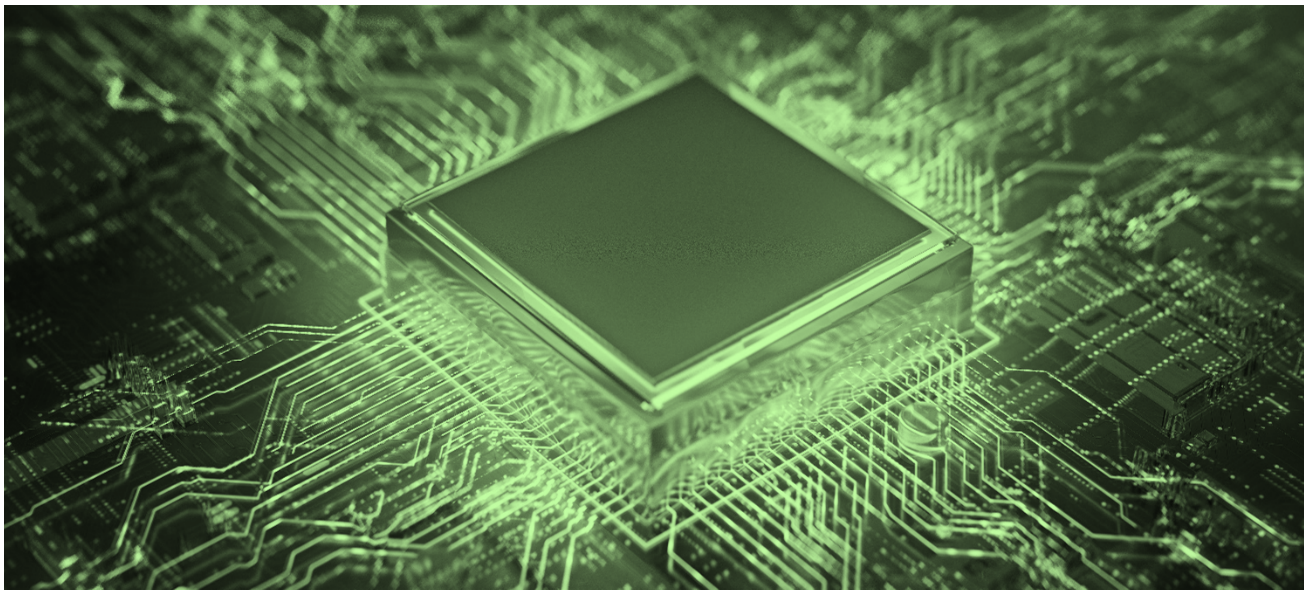
Field Programmable Gate Arrays (FPGAs)

In **Field Programmable Gate Arrays (FPGAs),** the logic gates can be set up to execute the specific operations of the problem to be solved, which allows *designing* a computing device specifically for each use case so that the maximum performance and energy efficiency can be achieved. This configuration of the system before the user code is launched induces some overhead, but it becomes negligible if the total application runtime (eventually over various executions) is large enough. The caveat is that the environment for the pre-configuration and FPGA programming using low-level approaches like VHDL or high-level synthesis (HLS) is still not familiar to application developers. OpenCL is moderately higher level, but the kernels still need to be managed from the host side. High-level programming environments are possible for a more comfortable approach, [6] and the FPGA can be used to accelerate the management of kernels in addition to the kernels themselves but achieving maximum performance still requires some low-level programming and tuning between different hardware generations, specific device configurations, and technologies. This makes achieving performance portability among different devices and generations of devices even more challenging than for different CPU/GPU architectures and generations.

Disruptive Approaches

Disruptive approaches such as neuromorphic and quantum technologies are also being explored. For example, quantum technologies have been shown to solve selected optimisation problems much faster and more efficiently than traditional von-Neumann systems. Similarly, neuromorphic computers are well suited for pattern recognition or neural networks. Until now, these technologies have proven their capabilities on individual use cases tuned to exploit them. Acceptance by the wider community still is hampered by a yet rather immature and device-specific software and programming environments, but progress is being done in all fronts to widen their use and applicability.

This diversity of processing technologies is posing new challenges to system integrators and application programmers alike. One challenge is memory management, where most approaches currently do not share the same memory and data placement needs to be managed explicitly by the application. Recently, approaches for cache coherent GPU/CPU memory systems have been announced, their efficiency needs however still to be proven. Not all accelerators or even more so disruptive approaches will allow the implementation of such coherency, though. In addition, application programmers have to deal with different, sometimes vendor-specific, programming environments (like OpenMP, MPI, Cuda, HiP, Sycl, OpenACC, OpenCL, etc.) to be able to exploit the different components. In addition, application developers need to decide, which platform or platform mix is best suited for their applications and part of an application should be executed on what platform. In turn, system integrators need to decide what mix of technologies is best suited for the expected application workload. To overcome these difficulties, more research is needed in standardised programming environments (with associated intelligent compilers and runtime systems) as well as modelling approaches to predict the potential performance of applications on certain technologies.



2. Memory Technologies

While the compute performance of CPUs, GPUs, etc. has traditionally captured most of the attention in HPC, e.g. in the TOP500 ranking, the performance of many HPC applications is actually constrained by the memory system, either due to latency/bandwidth (e.g. HPCG) or capacity (genomics, fluid dynamics). For more than five decades, HPC memory systems have followed the same basic design: a von Neumann architecture where the CPU controls a passive memory hierarchy. Nowadays, most HPC systems use a cluster architecture, in which each node has one or two sockets and its own memory hierarchy comprising L1, L2 and L3 caches and DRAM. However, the memory hierarchy is getting deeper with the introduction of further layers, particularly of High Bandwidth Memory (HBM) [7] and non-volatile memory DIMMs (NVDIMMs) [8] .

Several innovations in the memory subsystem organisation and underlying memory technologies are already in production (e.g. HBM, Non-Volatile Memory (NVM), disaggregated memories) or under active industrial and academic development (e.g. Processing in Memory (PIM) [9] [10] [11]). These systems provide opportunities to improve performance and reduce power consumption, but effectively and productively exploiting them presents many challenges:

Explicitly managed memory

Complex memory systems are generally architected in hardware not as a cache hierarchy, but on an equal ranking, e.g. Optane [12] is connected on a normal DDR interface. [13] The data placement and migration among multiple types of memories can be done in various places, for example as a transparent cache by the memory controller (DRAM is an inclusive cache for Optane in “memory mode”), transparently by the OS (at a page granularity), or by the runtime system or specific support library (at a page or object granularity, or even by the compiler. Alternatively, they may be exposed to the application either by exposing the specific memory types and topology or using a generalised interface in which memory is requested in terms of capabilities (high bandwidth, high capacity, etc.). Some APIs already exist: memkind [14], SICM (Simplified Interface to Complex Memory) [15] and some European projects are pioneering work in this direction, e.g. DEEP-SEA [16].

Processing in memory

Decades after being initially explored in the 1970s, Processing in Memory (PIM) is currently experiencing a renaissance [17]. PIM moves part of the computation to the memory devices, thereby addressing the mismatch between the von Neumann architecture and the requirements of important data-centric applications [18]. The interest in PIM has grown dramatically over the last years. The recent white paper from ETP4HPC [17] advises that wide acceptance of PIM in high-performance computing depends on our ability to create an ecosystem in which a number of PIM approaches can be designed and evaluated, leading to the selection of potential winners and adoption by system architects and end users.

Many diverse approaches are under development for PIM, differing in where the computation takes place (inside the memory array or periphery, or near the memory but outside it) [19] and in the type of operations supported, as well as in terms of cost, maturity, etc.. Any modifications required to existing codes should be as small and local as possible, and should be done in a performance-portable (if at all possible) and vendor independent way controlled in the long term by open standards. It is currently not clear what will be the eventual programming model to describe the code to run on a PIM system and how to control data placement. Overall, we can conclude that only coordinated innovations and co-design across the whole stack can make PIM a reality in production.

As with the heterogeneous computing technologies discussed above, a key challenge is to decide which mix of technologies works best for what applications and what data should be placed where and when. This requires more research to decide at what level(s) of the stack to manage data placement/migration, etc., on what basis (heuristics, historical data, sampling during the execution, user annotations etc.), subject to which metrics, and using which measurement/analysis tools—and there is still room to try various approaches. The best choice of memory for performance is not always obvious, e.g. if the memory bandwidth is low, then MCDRAM has higher latency than DDR (the bandwidth–latency curves cross), and thus worse performance. Standardisation in APIs and programming models will be needed eventually, but not too soon as the underlying technology is still evolving fast.

3. Storage Technologies

The storage system is typically attached to the supercomputer and made accessible via a file-system. The different storage technologies are organised in a hierarchical/pyramidal approach with the fastest (but smaller and more expensive) devices closest to the computer and the largest capacity (cheaper but slower) furthest away. This concept extends the principle of the memory hierarchy into the storage region, with non-volatile memories (which can be configured as memory or storage) sitting somewhere in the middle of both areas.

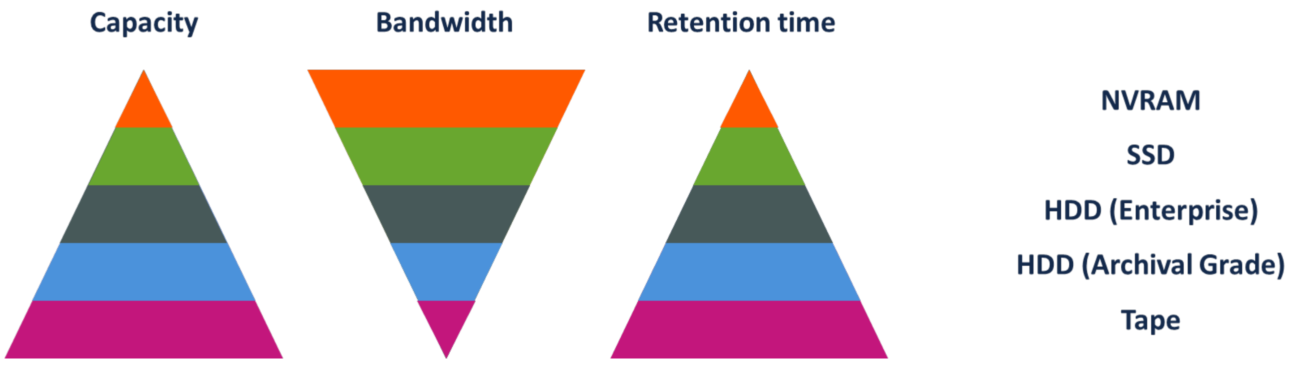


Figure 1 - Storage Hierarchy

The heterogeneous storage hierarchy-"tiers" consist of byte addressable NVRAM at the very top followed by block based Flash storage (exposed through Solid State Drives), High performance Hard Disk subsystem (eg: Serial Attached SCSI, SAS) and slower archival grade drives (eg: Shingled Magnetic Recording Drives). NVRAM components are typically present in Compute nodes and they are either exposed as byte addressable DIMMs or block addressable NVMe devices. NVRAM and Flash tiers typically could act as "Burst Buffers" hiding the latency of lower performance tiers below them. All tiers are exposed to parallel file system storage services or object based storage services. Applications can do I/O directly on the tiers or through files and objects mapped to the NVRAM tiers and memory. Parallel file systems/ object storage could also work closely with support from other software infrastructures such as Hierarchical Storage Managers (HSMs), which can help to move data to the right performance tier and make them available to workflows at the right time with appropriate performance.

One of the concepts explored for example by the SAGE/Sage2 [20] projects in the EU is "Global Memory Abstraction" where-in data objects are mapped from lower tiers to higher tiers. The objects could be mapped to NVRAM or memory - which enables applications to work with objects in memory at very high speeds and low latencies. These objects can be "drained" back to lower tiers at a later time. The concept also raises the possibilities to consolidate NVRAM pools distributed across multiple compute nodes. SAGE and Sage2 projects have explored hierarchical storage systems with all the known tiers - NVRAM, SSD, HDD & Tape - all managed as Objects by Object storage infrastructure software.

It is also to be borne in mind that tape has always provided a very low cost persistent storage resource, primarily used for long term archival. There are exabytes of data in tape across various scientific communities worldwide. However, frequently accessing data from tape multiple times, when needed by running applications, is always a challenge. Bringing tape into the fold of hierarchical tiers for running workflows is also actively being explored. In that there are software interfaces to data in tape from object stores hence providing the ability to actively work with Tape data as objects like in any higher level tier. This is explored in the IO-SEA EuroHPC project [21].

Furthermore, heterogeneous computing resources (GPUs, FPGAs, etc) very close to data can act to provide “in-storage computing” [22] capability. Whilst standard processing cores can be leveraged for in-storage computing, GPUs, FPGAs, etc. raise interesting new possibilities in storage system design. All this will help to reduce traffic from storage systems to compute nodes helping to eventually improve the time to solution. However, it needs to be noted that inclusion of some of these technologies as part of storage systems is also driven by economic considerations. In fact the role of the DPU (Data Processing Unit) [23] , pushed by some of the vendors, from the perspective of in-storage-computing is also something that needs to be explored.

In summary, heterogeneity in storage technologies exposes the same challenges as the heterogeneous memory technologies: where to place data and when to migrate data through the different tiers. More research is needed on tools that help with these decisions or are able to automatically take these decisions.

4. Full system architectures

High compute and data management performance can be achieved in a cost and energy efficient manner by wisely combining the diverse computing, memory, and storage devices mentioned above. The aim of a system architect is to choose complementary technologies, which potentiate each other’s strengths and compensate for each other's weaknesses under a given cost-target. The system-level architecture is then defined by the choice of technologies and how they are combined and connected with each other.

Cluster computers traditionally interconnected general-purpose CPUs via a high-speed network, and attached to an external disk-based storage. Accelerators, and in particular GPUs, started to play a role around 2010, due to the increasing power-needs of CPUs and the relatively high energy efficiency of accelerators. The typical integration consists of building one cluster node as a CPU with one or more PCIe-attached accelerators, and then interconnecting these “heterogeneous nodes” with each other via a high-speed network. Typically, the communication between the accelerators (within and between nodes) has to be managed by the host CPU, creating a communication bottleneck at the CPU-GPU connection. However, recent GPU technologies support (vendor specific) high performance interconnects to directly communicate between the GPUs in the node. The “GPU-islands” created in this way deliver a huge computational power per node and move the communication bottleneck to the inter-node interface, although new developments like GPU-direct communications might reduce the problem to some extent. An approach for multi-node FPGA-to-FPGA communication using shared memory, and without involving the CPU, has been developed by the EuroEXA project [24] .

A particular approach to heterogeneous system integration targeting diverse application portfolios is the so-called **Modular Supercomputer Architecture (MSA)** [25] developed in the DEEP projects [26]. It combines several clusters of potentially large size (called "modules"), each one tuned to best match the needs of a certain class of applications. For instance, a homogeneous CPU-based cluster that delivers high single-thread performance for low/medium scalable applications can be attached to an accelerator-based system that delivers energy efficient performance for highly scalable codes, and to a third CPU+accelerator cluster with large volumes of high-bandwidth memory for data-analytics codes. The architecture allows even the integration of more disruptive technologies such as neuromorphic or quantum computers. A federated network connects the module-specific interconnects, while an optimised resource manager enables assembling arbitrary combinations of these resources according to the application workload requirements. The goal of the MSA-approach is to enable an efficient orchestration of heterogeneous resources at the service of very diverse application profiles, so that each code can run on a near-optimal combination of resources and achieve excellent performance, increasing throughput and efficiency of use for the whole system.

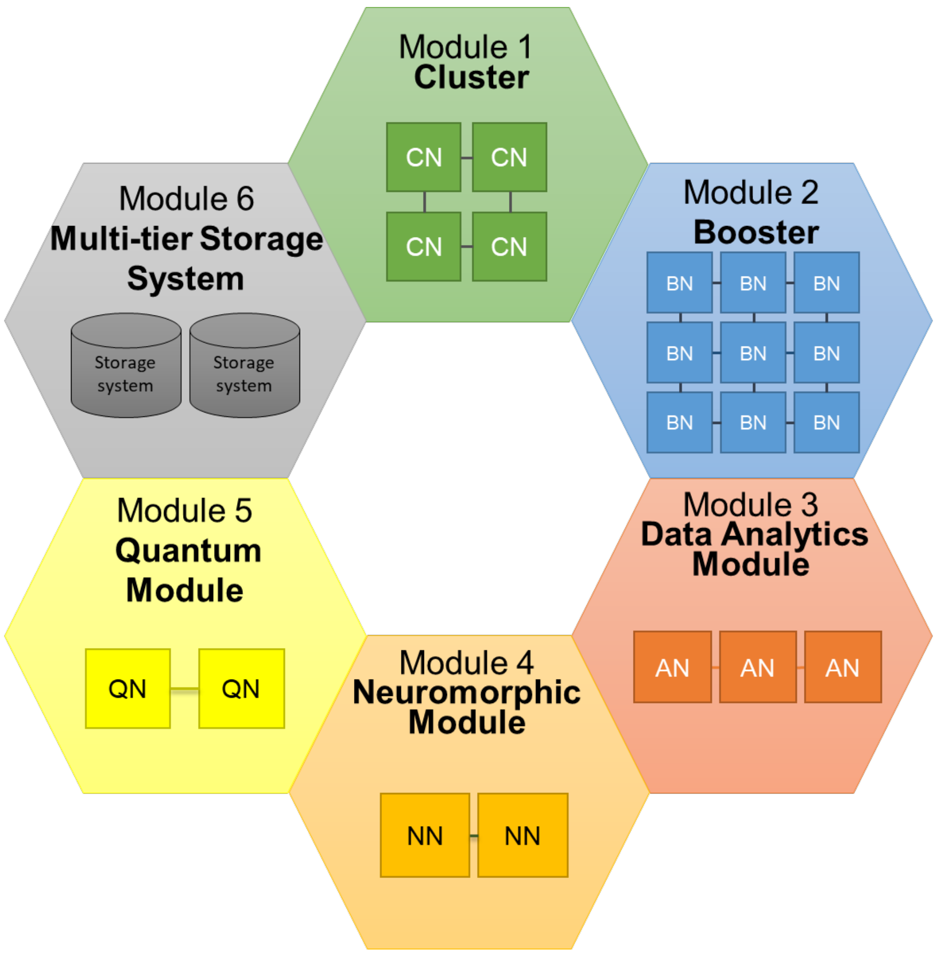


Figure 2 Modular Computing Systems [26]

Some novel architectures attempt to turn around the traditional approach that is built around the computing elements by putting a pool of globally accessible memory at the centre of the system. Such "memory-centric" architectures [27] [28] [29] promise a better energy efficiency through minimisation of data movement, but (when built at large scale) require advanced resilience mechanisms to deal with typically high failure rates on memory-hardware.

Continuing on this path of disaggregation, the network fabric can take a more central stage in not only facilitating data exchange, but also computation during data transport. While this is not a new idea -- some network technologies have been integrating in-fabric support of atomic RDMA operations and (simple) MPI collective operations for a decade -- the trend continues to gain traction, with active compute elements in switches and programmable NICs becoming a feature in multiple vendor's upcoming products[1] [30]. The challenge is accessing such features, e.g. making them available both in existing programming paradigms like MPI and through domain-specific toolkits (e.g., deep learning frameworks, workflow couplers). They also need to become accessible to high-end users ready to invest into offloading to the network, much like offloading to GPU accelerators has become commonplace.

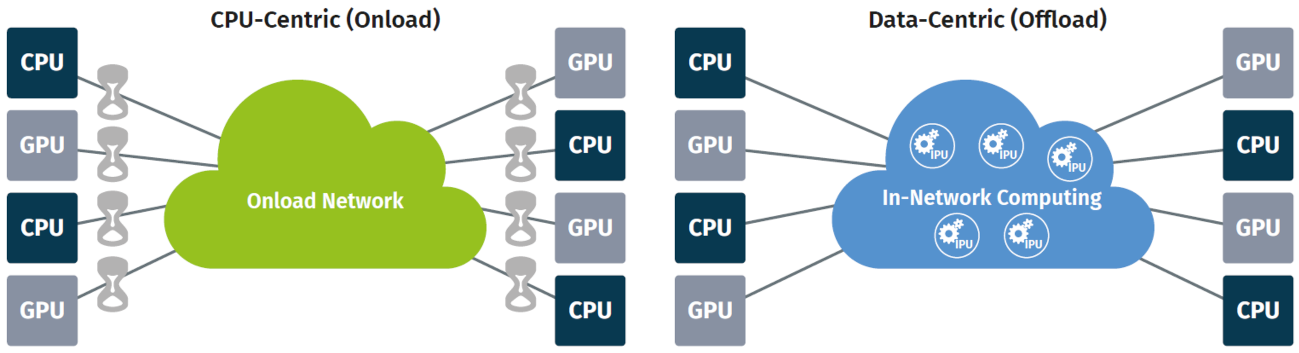


Figure 3 - CPU-centric vs. Data-centric Computing [31]

Not only resulting from heterogeneity in the HPC system, but clearly complicated by it, is the question of how to ensure increasing security requirements in such environments. While initially driven by special-purpose systems and then the need to separate potentially adversary customer's use of the same resources in a cloud computing environment, many -- but still not all -- components used in an HPC system can (or should be able to) provide some level of trusted computing capability: CPU and memory protection standards exist, some devices incorporate appropriate hardware roots-of-trust, and standards like SPIFFE are able to attest trust hierarchically [32]. However, in an HPC context there is still a lot of work left to do: UNIX style user or project separation will not be sufficient to segregate tenants on an exascale system shared by many different user communities; scheduling systems need to be aware and tied into the security design; high performance networks need to isolate traffic of jobs to both ensure guaranteed performance characteristics, but also shield users from crosstalk of other jobs, as such behaviour could otherwise permit DoS or side channel attacks. Finally, there is a trustworthy computing view expressed by HPC system operators: contrary to many cloud providers, HPC centres do care about the kind of workload run by their users, and they may want (or need) to prohibit usage of their resources for certain purposes. The question of how to attest user workloads as compliant with system operators' requirements is an entirely separate, but technologically intertwined, area of research.

Combining different heterogeneous components as discussed above into a full HPC system results in combinatorial effects in their complexity. It is a huge challenge to design systems such that they can be used efficiently by the expected workloads, particularly, when the workload is very heterogeneous. Modular systems as discussed above can help, deciding according to the user portfolio how much weight a particular module should get, and what connectivity is required within and between modules. To deal with these challenges, integrated projects that cover all levels of the HPC ecosystem are needed. *Also, interoperability and exchangeability of components, both hardware and software, should be easier to give system designers and users, alike, more flexibility.*

[1] e.g., https://www.arubanetworks.com/pensando-announcement/ and https://www.mellanox.com/products/BlueField-SmartNIC-Ethernet

5. Applications and System Software

On top of the heterogeneous hardware described above various system and application software stacks are being executed. These stacks typically have to be aware of the underlying hardware heterogeneity to various extents in order to optimally exploit the available hardware capabilities.

Scheduling and resource management

A core component of the system software is the **scheduling and resource management** system. Traditional usage of HPC systems centres around a batch scheduling system such as Slurm or PBS, which exposes the resources of the system either by partitioning them so that groups of mostly identical nodes are visible to the user, or by tagging nodes with feature labels, so that users can specify the desired kinds of features by selectors. These are feasible models if variability between nodes is small, if all resources to be allocated are encapsulated in the same nodes, and job requirements are homogenous across all components: Users request resources matching their job's needs, and the scheduler can make a decision about how to allocate resources.

Such a model is well suited to a mostly uniform set of resources, and jobs that have a constant resource requirement throughout their lifetime and across distributed parts (say, MPI ranks). It is not well suited to heterogenous jobs, or complicated workflows, where a resource matching over time may need to be performed. Solving this problem by workflow languages is not a sustainable approach: there are literally hundreds of workflow management tools, often used only by a small community, and many do not take HPC execution realms into account. Mapping scheduling decisions to one or more particular target architectures and HPC execution realms, taking into account batch schedulers with their own scheduling logic, makes holistic scheduling a far-off target. More research in these directions as well as fewer (and more broadly applicable) workflow languages will be needed.

A further complication arises if dynamically created or provisioned resources are considered. It is becoming a common requirement to allocate temporary storage space to a job (e.g., in NVRAM, as a dynamically created file system [33] [34], or by creating an object store namespace [35]), and possibly also perform stage-in and/or stage-out of data, in preparation for efficient access during the job runtime, but also possibly in order to access encrypted data temporarily in near-compute storage. While prologue and epilogue tooling can be used to perform these operations, schedulers are not advanced enough to perform this task sufficiently: Data preparation likely needs far fewer resources than a large-scale compute job, but may itself be well suited to be a distributed job. Deciding (and the accounting of) the effort spent is not a user or admin task, but should be part of the scheduling decisions, taking into account detailed knowledge about system resources, current utilisation, expected performance gains, and co-scheduling options with other jobs.

Finally, applications -- whether at user level, at middleware level, or in standard HPC libraries -- are missing system introspection infrastructure. While tools like hwloc [36] and netloc [37] often are available to provide an overview of components, there is a definite lack of higher-level portable resource description tooling that also properly reflects the restrictions placed on availability and performance characteristics by the scheduling decisions.

Compilers and runtime systems

Another important set of system software components are **compilers and runtime systems**. Despite new generations of HPC hardware becoming available at steady rates over the last decades, application developers (and so, users) could rely on the four cornerstones of compiler (Fortran, C, C++), MPI, PFS and scientific libraries (like fftw, BLAS, LAPACK) being available and tuned to the current target system. This has started to change with GPUs bringing their own programming paradigms such as CUDA [38] and HIP [39] into the mix, which cannot always be abstracted entirely by models like OpenACC, OpenMP, and OpenCL [40]. The situation worsens with novel accelerators entering the landscape, from programmable hardware like FPGAs to custom-built devices that serve best as a backend to a particular programming paradigm in a specialised application domain, such as TensorFlow. With heterogeneity becoming mainstream, relying on the compiler to make the final (and static) decisions about code transformations, pattern detection and substitution by library functions, as well as data access optimisation becomes less and less realistic

This indicates potential for the rise of additional middlewares, that is, runtimes that expose a higher level of abstraction over the actual machine architecture. It can be argued that POSIX and MPI, and standard ScaLAPACK functions are effectively such middlewares, but data centric abstractions like Kokkos [41], Raja [42], ADIOS2 [43], XIOS [44], and Maestro [45], task based execution runtimes [46], object stores[2], and deep learning frameworks are becoming equally important. The interplay (and compatibility or interoperability) of all of these is posing a major software engineering and maintenance challenge. Standardisation of the system software as well as middleware layers will be an important aspect for ensuring uptake and maintainability.

Finally, software containers like Singularity [47], Docker [48], Podman [49], and Sarus [50] cannot be ignored in this context. While they solve the dependency issue, and relieve the end user from complicated software installation burden, they create new questions for efficient system operations and use: Avoiding useless rebuilds, but rebuilding containers to best match the target platform, defining, maintaining, and mapping compatible APIs between host system and containers transparently to the user to ensure full performance, and -- in more complicated form -- providing resource awareness to the container content from the host system cannot be considered 'solved' at this time. Secure and trustworthy execution of HPC workloads is in its infancy -- while Cloud system operations often rely on virtualisation to provide this, the HPC community efficiency concerns prohibit such costly isolation procedures. Future middlewares, in concert with the scheduling system, will need to address this challenge.

Since GPUs became a wide-spread component of large HPC systems, almost all **applications** have to deal with heterogeneity in computing and memory technology. While programming environments, compilers and runtime systems discussed above are intended to help improve efficiency it is ultimately the responsibility of the application programmers to exploit the heterogeneous hardware to the best possible extent. Consequently, much effort has gone into porting and optimising applications for GPU systems in recent years - and many of these efforts show great results. With further increased heterogeneity in many system components as discussed above these efforts need to continue and even intensify. Support in performance tools and models for all forms of heterogeneity are also important to enable this. Examples of ongoing efforts include the US Exascale Computing Project (ECP) [51] and the European Centres of Excellence in High Performance Computing [52].

In addition, applications are increasingly becoming heterogeneous themselves. Instead of single, large-scale simulations in many application domains different kinds of workflows are becoming a dominant usage model. This ranges from ensemble systems where the same application is executed multiple times with different parameters to complex workflows combining simulation, data analysis, and AI methods. Indeed, the increasing use of AI methods is drastically changing the way supercomputers are being used. In tasks like pre- and post-processing, application steering and in-situ analysis, AI methods play an increasingly important role. However, AI methods have hardware requirements that are often not compatible with the requirements of the application. Therefore, heterogeneous systems, where different parts of the system are configured in a modular way, offer an excellent platform for the execution of these modern workflows. However, efficient workflow systems (including scheduling etc.) are required.

[2] for instance DAOS (https://docs.daos.io/), CORTX (https://www.seagate.com/de/de/products/storage/object-storage-software/), MinIO (https://min.io/), and to some degree Ceph (https://ceph.io/).

6. Recommended R&D in the next 2-4 years

To tackle the challenges posed by heterogeneity in HPC as discussed above, research on all levels is needed. This starts with **system architecture**, which has to deliver a concept to efficiently combine and connect the variety of resources, including also opportunities to integrate new and upcoming disruptive technologies. For an effective use and share of resources, the **resource management** software needs to organise and orchestrate heterogeneous resources taking into account side constraints like thermal management and energy efficiency, as well as interactive usage and malleability of applications. More research is needed in these areas to provide the scheduler and resource manager with the necessary flexibility and dynamic functionality, potentially also applying AI methods to extract insight from both the system monitoring and application performance data. Also, efficient execution of complex workflows and large ensembles requires more attention, including analysis tools with introspection capabilities to better match application tasks to adequate hardware components.

The drastic changes in **storage** technologies ask for better support for the continuum from on-node to siloed storage, disaggregated storage, object store, and in-storage compute.

Security and trustworthy aspects are becoming critical also in HPC: Confidential Computing in HPC has traditionally been implemented via on-premises single-tenant systems. The on-going democratisation of HPC through cloud-like multi-tenant systems necessitates a secure solution to off-site job execution on “foreign”-hosted high performance resources. Cloud vendors currently utilise hypervisor-based virtualisation, however this is expected to be a performance inhibitor for HPC. Instead, we believe an end-to-end protocol for remote workload execution protected from in-storage, in-flight, and in-execution access by third parties through encryption and trust and identity federation is becoming important, and will also become a requirement to integrate with Gaia-X.

On the **compiler** side, more efficient vectorisation and parallelisation, potentially exploiting polyhedral compilation and just-in-time code generation are needed. In addition, compiler support, possibly in conjunction with appropriate runtime systems, for managing heterogeneous memory systems will be needed. Advanced **runtime systems** will need to support the placement of tasks on processing and memory technologies, explicit management of heterogeneous memories (not necessarily in a hierarchy), data transfers, data locality aware abstractions, malleability, and coupling of application tasks.

More research is also needed on **programming models** such that common models for GPUs, FPGAs, etc. can be provided. To increase the efficiency of programmers, high-level parallel programming models and DSLs can provide a more intuitive way of programming. In addition, compilers and runtime systems need to ensure interoperability/composability of different components (e.g. in-situ tasks) and ideally performance portability to the largest possible extent.

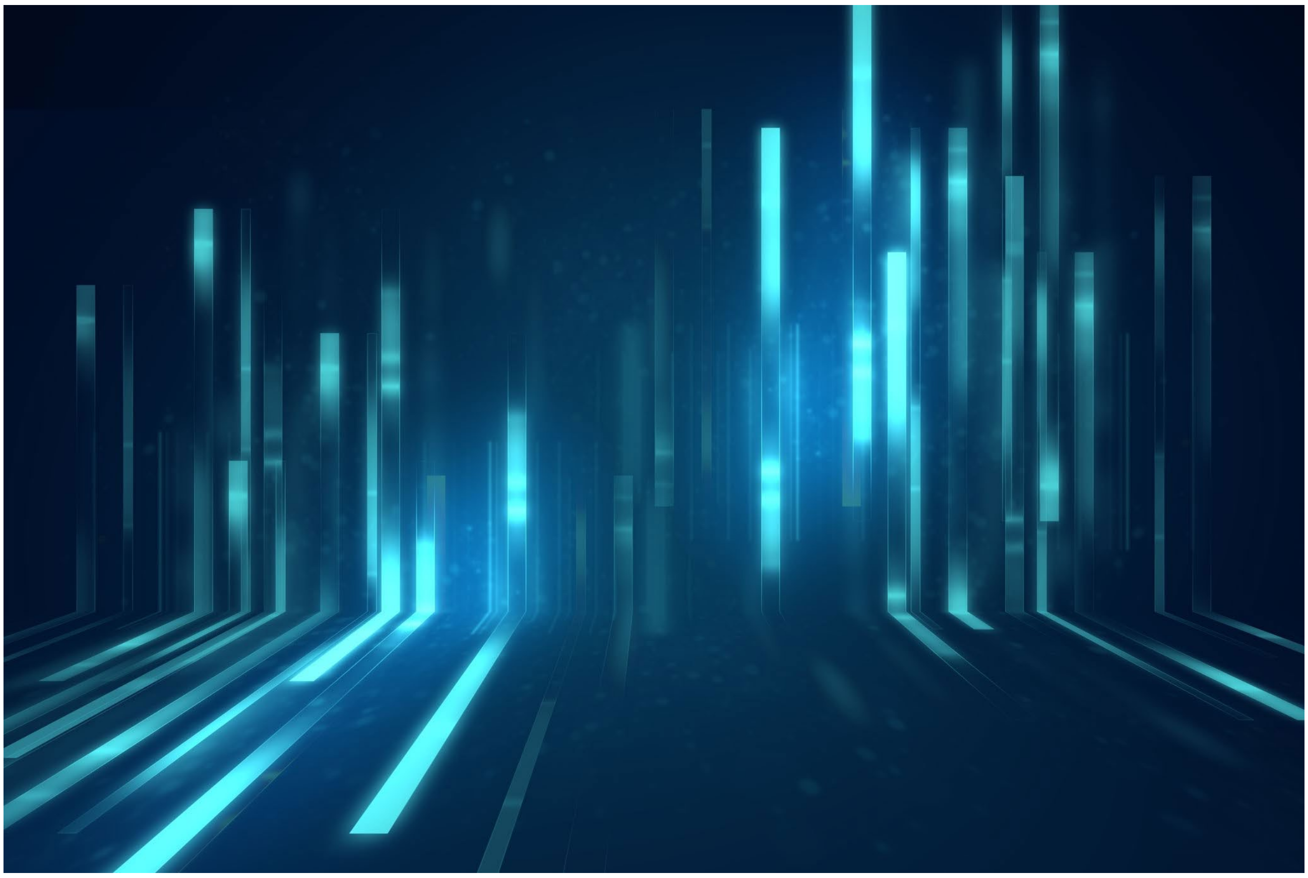
Tools also need to be adapted for heterogeneity. This includes particularly performance analysis tools, monitoring tools, and tools to identify the best resources for each step of the workflow and accordingly execute them.

Finally, **applications** have to exploit novel methods and algorithms enabled by heterogeneous components. This also includes novel formulations of existing models, using e.g. mixed precision arithmetic or non-IEEE data formats. Complex application workflows will be enabled via system software and tools, but the last-mile optimisations are likely to require intervention of the application developers.

All the above research areas must be well aligned and coordinated, in a concerted and coherent approach towards integration, support, management, and use of heterogeneous compute resources. This could be achieved by dedicated coordinated integration projects **hardening and combining results covering all levels of the HPC ecosystem**.

Conclusions

Heterogeneity in HPC started with the introductions of GPUs more than a decade ago. Since then, the trend towards heterogeneous hardware has continued and increased. Heterogeneity is now present in all system components, from computing elements, memory and storage technologies, to full system designs. This heterogeneity offers great opportunities for a more efficient use of HPC resources, but also poses severe challenges. To overcome these challenges, increased research efforts are needed on all levels of the HPC ecosystem. It is crucial that these efforts are perfectly coordinated, addressing hardware heterogeneity in a holistic and coherent manner.



References

[1] R. H. Dennard, F. H. Gaensslen, H.-N. Yu, V. L. Rideout, E. Bassous and A. R. LeBlanc, “Design of ion-implanted MOSFET's with very small physical dimensions,” *IEEE Journal of Solid-State Circuits,* vol. 9, no. 5, pp. 256-268, October 1974.

[2] “Intel discontinues Xeon Phi 7200 series Knights Landing coprocessor cards,” [Online]. Available: https://www.anandtech.com/show/11769/intel-discontinues-xeon-phi-7200-series-knights-landing-coprocessor-cards.

[3] OpenACC, [Online]. Available: https://www.openacc.org/.

[4] OpenCL, [Online]. Available: https://www.khronos.org/registry/cl/specs/opencl-1.1.pdf.

[5] OpenMP, [Online]. Available: https://www.openmp.org/.

[6] “OmpSs@FPGA,” [Online]. Available: https://pm.bsc.es/ompss-at-fpga.

[7] “High Bandwidth Memory (HBM) DRAM, White Paper,” JEDEC Solid State Technology Association, 2013.

[8] “JEDEC Publishes DDR4 NVDIMM-P Bus Protocol Standard,” February 2021. [Online]. Available: https://www.jedec.org/news/pressreleases/jedec-publishes-ddr4-nvdimm-p-bus-protocol-standard.

[9] K. Wang, K. Angstadt, C. Bo, N. Brunelle, E. Sadredini, T. Tracy, J. Wadden, M. Stan and K. Skadron, “An Overview of Micron's Automata Processor,” in *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES)*, 2016.

[10] Y.-C. Kwon, S. H. Lee, J. Lee, S.-H. Kwon, J. M. Ryu, J.-P. Son, O. Seongil, H.-S. Yu, H. Lee, S. Y. Kim, Y. Cho, J. G. Kim, J. Choi, H.-S. Shin, J. Kim, B. Phuah, H. Kim, M. J. Song, A. Choi, D. Kim, S. Kim, E.-B. Kim, D. Wang, S. Kang, Y. Ro, S. Seo, J. Song, J. Youn, K. Sohn and N. S. Kim, “A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications,” in *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC)*, 2021.

[11] F. Devaux, “The True Processing in Memory Accelerator,” in *IEEE Hot Chips Symposium (HCS)*, 2019.

[12] “Intel Optane Memory - Responsive Memory, Accelerated Performance,” [Online]. Available: https://www.intel.com/content/www/us/en/products/details/memory-storage/optane-memory.html.

[13] “Intel at last announces Optane memory: DDR4 that never forgets.,” May 2018. [Online]. Available: https://arstechnica.com/gadgets/2018/05/intel-finally-announces-ddr4-memory-made-from-persistent-3d-xpoint.

[14] “memkind GitHub,” [Online]. Available: http://memkind.github.io/memkind/.

[15] “SICM GitHub,” [Online]. Available: https://github.com/lanl/SICM.

[16] DEEP-SEA Project. Horizon 2020 grant agreement 955606, [Online]. Available: https://www.deep-projects.eu/.

[17] P. Radojković, P. Carpenter, P. Esmaili-Dokht, R. Cimadomo, H.-P. Charles, A. Sebastian and P. Amato, “Processing in Memory: The Tipping Point,” ETP4HPC, 2021.

[18] S. Ghose, A. Boroumand, J. S. Kim, J. Gomez-Luna and O. Mutlu, “Processing-in-memory: A workload-driven perspective,” *IBM Journal of Research and Development,* vol. 63, no. 6, pp. 3:1-3:19, November-December 2019.

[19] P. Siegl, R. Buchty and M. Berekovic, “Data-centric computing frontiers: A survey on processing-in-memory,” in *Proceedings of the Second International Symposium on Memory Systems*, 2016.

[20] S. Narasimhamurthy, N. Danilov, S. Wu, G. Umanesan, S. Markidis, S. Rivas-Gomez, I. B. Peng and S. De Witt, “Sage: percipient storage for exascale data centric computing,” *Parallel Computing,* no. 81, 2019.

[21] IO-SEA project, [Online]. Available: https://iosea-project.eu.

[22] Z. Ruan, T. He and J. Cong, “Designing In-Storage Computing System for Emerging High-Performance Drive,” in *Proceedings of the 2019 USENIX Annual Technical Conference*, Renton, WA, USA, 2019.

[23] “What's a DPU data processing unit,” [Online]. Available: https://blogs.nvidia.com/blog/2020/05/20/whats-a-dpu-data-processing-unit/.

[24] EuroEXA project. Horizon 2020 grant agreement number 754337, [Online]. Available: https://euroexa.eu/.

[25] E. Suarez and T. Lippert, “Modular Supercomputing Architecture: from idea to production,” in *Contemporary High Performance Computing: from Petascale toward Exascale*, vol. 3, Ed. Jeffrey S. Vetter, CRC Press, 2019, pp. 223-251.

[26] DEEP projects. Horizon 2020 grant agreements 287530, 610476, 754303 and 955606., [Online]. Available: https://www.deep-projects.eu/.

[27] J. Schmidt, “Network Attached Memory, Chapter 4 of the PhD Thesis: "Accelerating Checkpoint/Restart Application Performance in Large-Scale Systems with Network Attached Memory", Ruprecht-Karls University Heidelberg - Fakultaet fuer Mathematik und Informatik,” [Online]. Available: http://archiv.ub.uni-heidelberg.de/volltextserver/23800/1/dissertation_juri_schmidt_publish.pdf.

[28] P. Faraboschi, K. Keeton, T. Marsland and D. Milojicic, “Beyond Processor-centric Operating Systems,” in *15th Workshop on Hot Topics in Operating Systems (HotOS XV)*, 2015.

[29] Rigo et al., “Paving the way towards a highly energy-efficient and highly integrated compute node for the Exascale revolution: the ExaNoDe approach,” in *Euromicro Symposium on Digital System Design, DSD 2017*, 2017.

[30] S. Schweitzer, “Panel: SmartNIC or DPU, Who Wins?,” in *2020 IEEE Symposium on High-Performance Interconnects (HOTI)*, see https://technologyevangelist.co/2020/08/25/smartnics-vs-dpus/, 2020.

[31] M. Malms, M. Ostasz, M. Gilliot, P. Bernier-Bruna, L. Cargemel, E. Suarez, H. Cornelius, M. Duranton, B. Koren, P. Rosse-Laurent, M. S. Pérez-Hernández, M. Marazakis, G. Lonsdale, P. Carpenter, G. Antoniu , S. Narasimhamurthy, A. Brinkman, D. Pleiter, A. Tate, A. Wierse, J. Krueger, H.-C. Hoppe and E. Laure, “ETP4HPC's Strategic Research Agenda for High-Performance Computing in Europe 4,” Zenodo, 2020.

[32] SPIFFE, [Online]. Available: https://spiffe.io/.

[33] F. Tessier, M. Martinasso, M. Chesi, M. Klein and M. Gila, “Dynamic Provisioning of Storage Resources: A Case Study with Burst Buffers,” in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2020.

[34] “BeeOND™: BeeGFS On Demand,” [Online]. Available: https://www.beegfs.io/wiki/BeeOND.

[35] DAOS, [Online]. Available: https://docs.daos.io/.

[36] “Portable Hardware Locality (hwloc),” [Online]. Available: https://www.open-mpi.org/projects/hwloc/.

[37] “Portable Network Locality (netloc),” [Online]. Available: https://www.open-mpi.org/projects/netloc/.

[38] “CUDA Zone,” [Online]. Available: https://developer.nvidia.com/cuda-zone.

[39] “HIP Guide,” [Online]. Available: https://rocmdocs.amd.com/en/latest/Programming_Guides/HIP-GUIDE.html.

[40] S. Memeti, L. Li, S. Pllana, J. Kołodziej and C. Kessler, “Benchmarking OpenCL, OpenACC, OpenMP, and CUDA: Programming Productivity, Performance, and Energy Consumption,” in *ARMS-CC '17. Proceedings of the 2017 Workshop on Adaptive Resource Management and Scheduling for Cloud Computing*, July 2017.

[41] Kokkos, [Online]. Available: https://kokkos.org/.

[42] “RAJA,” [Online]. Available: https://computing.llnl.gov/projects/raja-managing-application-portability-next-generation-platforms.

[43] ADIOS2, [Online]. Available: https://csmd.ornl.gov/software/adios2.

[44] “XIOS wiki page,” [Online]. Available: http://forge.ipsl.jussieu.fr/ioserver/wiki.

[45] Maestro project, [Online]. Available: https://www.maestro-data.eu/.

[46] O. Aumage, P. Carpenter and S. Benkner, “Task-Based Performance Portability in HPC,” Zenodo, 2021.

[47] SINGULARITY, [Online]. Available: https://singularity.hpcng.org/.

[48] Docker, [Online]. Available: https://www.docker.com/.

[49] Podman, [Online]. Available: https://podman.io/.

[50] Sarus, [Online]. Available: https://sarus.readthedocs.io/.

[51] ECP, [Online]. Available: https://www.exascaleproject.org/.

[52] European Centres of Excellence in High Performance Computing, [Online]. Available: https://www.hpccoe.eu/eu-hpc-centres-of-excellence2/.

Authors:

Paul Carpenter is leader of the Microservers and System Software team at the Barcelona Supercomputing Center (BSC).

Utz-Uwe Haus is Head of HPE HPC/AI EMEA Research Lab, and technical coordinator of the MAESTRO project

Erwin Laure is Director of the Max Planck Computing and Data Facility (MPCDF)

Sai Narasimhamurthy leads European R&D for Seagate Systems and leads the SAGE & Sage2 storage projects.

Estela Suarez is Senior Scientist at the Jülich Supercomputing Centre from Forschungszentrum Jülich, and leads the DEEP projects series.

This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreements No. 955606 (DEEP-SEA), No.801101 (MAESTRO), No. 823964 (FocusCoE), No. 754337 (EuroEXA).

Paul Carpenter holds the Ramon y Cajal fellowship under contract RYC2018-025628-I of the Ministry of Economy and Competitiveness of Spain.

Cite as: P. Carpenter et al., « Heterogeneous High-Performance Computing », ETP4HPC White Paper, 2022, doi 10.5281/zenodo.6090425.

DOI: 10.5281/zenodo.6090425