


# Challenges in High-Performance Computing

Philippe Olivier Alexandre Navaux  [Federal University of Rio Grande do Sul | [navaux@inf.ufrgs.br](mailto:navaux@inf.ufrgs.br)]

Arthur Francisco Lorenzon   [Federal University of Rio Grande do Sul | [aflorenzon@inf.ufrgs.br](mailto:aflorenzon@inf.ufrgs.br)]

Matheus da Silva Serpa  [Federal University of Rio Grande do Sul | [msserpa@inf.ufrgs.br](mailto:msserpa@inf.ufrgs.br)]

 *Institute of Informatics, Universidade Federal do Rio Grande do Sul - PO Box 15064, Av. Bento Gonçalves, 9500, 91501-970, Porto Alegre, RS, Brazil*

**Received:** 15 September 2021 • **Accepted:** 30 March 2023 • **Published:** 01 August 2023

**Abstract** High-Performance Computing, HPC, has become one of the most active computer science fields. Driven mainly by the need for high processing capabilities required by algorithms from many areas, such as Big Data, Artificial Intelligence, Data Science, and subjects related to chemistry, physics, and biology, the state-of-art algorithms from these fields are notoriously demanding computer resources. Therefore, choosing the right computer system to optimize their performance is paramount. This article presents the main challenges of future supercomputer systems, highlighting the areas that demand the most of HPC servers; the new architectures, including heterogeneous processors composed of artificial intelligence chips, quantum processors, the adoption of HPC on cloud servers; and the challenges of software developers when facing parallelizing applications. We also discuss challenges regarding non-functional requirements, such as energy consumption and resilience.

**Keywords:** High-Performance Computing, Supercomputers, Exascale, Computer Architecture, Parallel Programming

## 1 Introduction

High-Performance Computing, HPC, has been settled as the area of specialists concerned with machines with the greatest processing power of a given time, represented by supercomputers. Traditionally, two selections occur annually to indicate which machines in the world have reached the top of this processing power disclosed in the TOP500 [Dongarra and Strohmaier, 2020].

The HPC scenario has been changing in recent years, pressured by the processing power requirements from Artificial Intelligence, AI, Machine Learning, ML, and Deep Learning, DL, algorithms, and the need for this power to matter learning data in Big Data environments [Verbraeken *et al.*, 2020]. In summary, HPC is not only an area that meets a niche of needs of researchers in the fields of physics, chemistry, and biology, among other specific ones, but an area that needs to meet the demands of society as a whole.

Another deeper change has occurred. Cloud vendors are investing in global networks of massive-scale systems for HPC systems [Reed *et al.*, 2022]. Simultaneously, one can observe a massive migration from parallel application executions on dedicated HPC servers to cloud environments due to many factors, such as the facility to access them over the internet and for provisioning high-performance architectures on demand while keeping operating costs low. In this scenario, hardware and software technologies have been proposed over the years to allow a sustainable execution of HPC applications in cloud environments.

On top of that, HPC environments are becoming increasingly heterogeneous to support applications' performance and energy demands in scientific areas. In this scenario, most supercomputers are equipped with hardware accelerators and techniques with distinct computing power capabilities, such as graphical processing units, GPUs, field-programmable

gate arrays, FPGAs, processing in memory, PIM, and recently, the adoption of Quantum processors. Hence, several parallel programming libraries and patterns are proposed yearly to get the most performance from such architectures.

Hence, we present in this article a comprehensive discussion regarding: (i) the demands for HPC servers from different fields of science and society; (ii) the main challenges in the HPC area that researchers will face shortly in hardware and software to further optimize the execution of applications in terms of performance, energy consumption, and resilience; (iii) the trends related to the increasing availability of heterogeneous architectures in HPC systems and how to get the most out of each device.

## 2 The Need for HPC

In this section, we discuss how HPC systems can be employed to optimize the execution of applications widely known in scientific areas.

### 2.1 Demands from Big Data area

The term Big Data was used for the first time in 1997, referring to the growing number of data generated every second in the world in a structured or unstructured way. Michael Cox and David Ellsworth, both working at NASA, wrote the article "Managing Big Data for Scientific Visualization" for the 1997 Conference on Visualization, introducing the concept of Big Data to the academic community [Cox and Ellsworth, 1997].

The amount of data generated per year doubles every two years and grows exponentially [Desjardins, 2019]. With this evolution, we are reaching the level of Yotta data, which corresponds to 1024 bytes or 10008 zettabytes. This amount of

data, also known as Big Data, comes from different sources (e.g., sensors on the Internet of Things) and needs treatment by non-traditional systems for manipulating, analyzing, and extracting information from the dataset. However, only around 20% of this extracted information would be helpful.

Therefore, Big Data is a collection of large and complex datasets that becomes difficult to process using database management tools. It is often a collection of legacy data. In addition to dealing with new ways of managing data, the Big Data area needs great computing power to process this data. Furthermore, we must remember that data are numbers and codes without treatment. Information, on the other hand, is processed data. It is the processing of data that will create meaning. Finally, knowledge is knowing about a specific subject; it is having an application for information.

It is also important to mention that knowledge about information is power nowadays. It always has been, but with the advent of large amounts of data and the ability to process them, it became possible to conduct analysis and decision-making. Information has become the most critical asset. Companies that hold data and have the power to process and analyze it hold today a capacity many times greater than the governments. Furthermore, there are no frontiers for information, and these companies are capturing data from almost all over the world.

In summary, machines with high processing power, such as supercomputers, are needed to assist in data storage and extraction in the Big Data era.

## 2.2 Demands from the Artificial Intelligence Area

According to the US Department of Energy's "AI for Science" report [Stevens *et al.*, 2020], new Artificial Intelligence techniques will be indispensable to support the continued growth and expansion of Science infrastructure through Exascale systems. The experience of the scientific community, the use of Machine Learning, the simulation with HPC, and the data analysis methods allowed a unique and new growth of opportunities for Science, discoveries, and more robust approaches to accelerated Science and its applications for the benefit of humanity.

The convergence of HPC with AI allows simulation environments to employ deep reinforcement learning in various problems, such as simulating robots, aircraft, autonomous vehicles, etc. DL techniques accelerate simulations by replacing models in climate prediction, geoscience, pharmaceuticals, etc. New frontiers in physics are being reached by increasing the application of Partial Differential Equations (PDE) with DL for simulations.

On the other hand, one major bottleneck in using ML and DL algorithms is the learning phase before their use. Depending on the available computing infrastructure, this activity can take weeks and even months. That is where high-performance processing accelerates this step.

In the end, due to the HPC demands from the AI area, companies are investing in parallel frameworks to optimize the execution of AI software. As an example, NVIDIA provides different solutions so that users can take advantage of GPUs optimized for AI algorithms: TensorFlow, an open-source

platform for Machine Learning that provides tools and libraries to allow easy deployment of codes across heterogeneous devices; PyTorch, a GPU-accelerated tensor computational framework; and TorchANI, an implementation of Accurate Neural Network Engine for Molecular Energies. Another example is the Intel Neural Compressor tool that helps software developers to easily and quickly deploy inference solutions on popular deep learning frameworks, including TensorFlow and PyTorch.

Therefore, one of the challenges software developers will face in the convergence of HPC with AI is how to efficiently use the available hardware devices to get the most out of all the provided frameworks. Simultaneously, hardware vendors have the challenging task of designing optimized processors for AI computing, as we discuss in Section 3.3.

## 2.3 Demands from Data Science and Related Areas

In conjunction with advancements in supercomputing, the rise of data generation technologies has resulted in the convergence of HPC and data science applications. The widespread adoption of high-throughput data generation technologies, scalable algorithms and analytics, and efficient methods for managing large-scale data has established scalable Data Science as a central pillar for accelerating scientific discovery and innovation [Xenopoulos *et al.*, 2016].

Data science can be defined as a field that combines different areas (e.g., math, statistics, artificial intelligence, machine learning, and specialized programming) with a subject to uncover insights that are hidden in an organization's data. The lifecycle of data science involves tools, roles, and processes that usually undergo the following steps: data collection, data storage and processing, data analysis, and presentation of reports and data visualizations that make the insights [Kelleher and Tierney, 2018].

All steps are accelerated with the employment of HPC systems and can be used to provide insights beneficial to society: fraud and risk detection, search engines, advanced image recognition, speech recognition, and airline route planning. In summary, even with the use of HPC to optimize the execution of data science algorithms, one of the main challenges faced by the area is to use hardware resources better to reduce the training cost of learning algorithms.

## 3 HPC Architectures and Processors Challenges

The last section (section 2) showed how advances in scientific areas will demand more processing power to run the models, manage the files, and extract the data, among other demands. Hence, the main challenges HPC needs to overcome to get results on time in terms of hardware for its employment in such areas will be presented in this section.



Figure 1. Fugaku [Fujitsu, 2021], Frontier [2021], Frontier [2021], and El Capitan supercomputers[LLNL, 2021].

System attributes	ALCF Now	NERSC Now	OLCF Now	NERSC Pre-Exascale	ALCF Pre-Exascale	OLCF Exascale	ALCF Exascale
<b>Name (Planned) Installation</b>	Theta 2016	Cori 2016	Summit 2017-2018	Perlmutter (2020-2021)	Polaris (2021)	Frontier (2021-2022)	Aurora (2022-2023)
<b>System peak</b>	> 15.6 PF	> 30 PF	200 PF	> 120PF	35 – 45PF	>1.5 EF	≥ 1 EF DP sustained
<b>Peak Power (MW)</b>	< 2.1	< 3.7	10	6	< 2	29	≤ 60
<b>Total system memory</b>	847 TB DDR4 + 70 TB HBM + 7.5 TB GPU memory	~1 PB DDR4 + High Bandwidth Memory (HBM) + 1.5PB persistent memory	2.4 PB DDR4 + 0.4 PB HBM + 7.4 PB persistent memory	1.92 PB DDR4 + 240TB HBM	> 250 TB	4.6 PB DDR4 + 4.6 PB HBM2e + 36 PB persistent memory	> 10 PB
<b>Node performance (TF)</b>	2.7 TF (KNL node) and 166.4 TF (GPU node)	> 3	43	> 70 (GPU) > 4 (CPU)	> 70 TF	TBD	> 130
<b>Node processors</b>	Intel Xeon Phi 7320 64-core CPUs (KNL) and GPU nodes with 8 NVIDIA A100 GPUs coupled with 2 AMD EPYC 64-core CPUs	Intel Knights Landing many core CPUs Intel Haswell CPU in data partition	2 IBM Power9 CPUs + 6 Nvidia Volta GPUs	CPU only nodes: AMD EPYC Milan CPUS; CPU-GPU nodes: AMD EPYC Milan with NVIDIA A100 GPUs	1 CPU; 4 GPUs	1 HPC and AI optimized AMD EPYC CPU and 4 AMD Radeon Instinct GPUs	2 Intel Xeon Sapphire Rapids and 6 Xe Ponte Vecchio GPUs
<b>System size (nodes)</b>	4,392 KNL nodes and 24 DGX-A100 nodes	9,300 nodes 1,900 nodes in data partition	4608 nodes	> 1,500(GPU) > 3,000 (CPU)	> 500	> 9,000 nodes	> 9,000 nodes
<b>CPU-GPU Interconnect</b>	NVLINK on GPU nodes	N/A	NVLINK Coherent memory across node	PCIe		AMD Infinity Fabric Coherent memory across the node	Unified memory architecture, RAMBO
<b>Node-to-node interconnect</b>	Aries (KNL nodes) and HDR200 (GPU nodes)	Aries	Dual Rail EDR-IB	HPE Slingshot NIC	HPE Slingshot NIC	HPE Slingshot	HPE Slingshot
<b>File System</b>	200 PB, 1.3 TB/s Lustre 10 PB, 210 GB/s Lustre	28 PB, 744 GB/s Lustre	250 PB, 2.5 TB/s GPFS	35 PB All Flash, Lustre	N/A	695 PB + 10 PB Flash performance tier, Lustre	≥ 230 PB, ≥ 25 TB/s DAOS



ASCR Computing Upgrades At-a-Glance  
November 24, 2020

Figure 2. Evolution of supercomputers

### 3.1 New Generation of Processors and Accelerators

Several proposals for parallel architectures have emerged recently as depicted in Figures 1 and 2, which should shake and change the processor market. Next we highlight the four supercomputers with the highest computing power currently.

The ARM company designed the A64FX processor that allowed Fujitsu’s FUGAKU computer with NVIDIA GPUs [Fujitsu, 2021], Japan, to remain for two years (2020 and 2021) as the fastest computer in the TOP500 list, reaching a performance of 442 Petaflops (Figure 1).

The first supercomputer to deliver performance greater than 1 Exaflop was the Frontier (Figure 1) from the Oak Ridge laboratory, USA. It was designed by Cray-HPE with AMD Epyc processors and Radeon accelerators and reached the performance of 1.1 Exaflops [Frontier, 2021], according to the TOP500 list.

After breaking the exaflop performance barrier, it is natural that the number of supercomputers achieving such a performance level increases. In this scenario, Aurora (Figure 1) is a supercomputer from Argonne’s laboratory that is being developed with Intel processors adopting the Ponte Vecchio architecture that integrates the Xeon processor and the Xe accelerator on the same chip [Aurora, 2021].

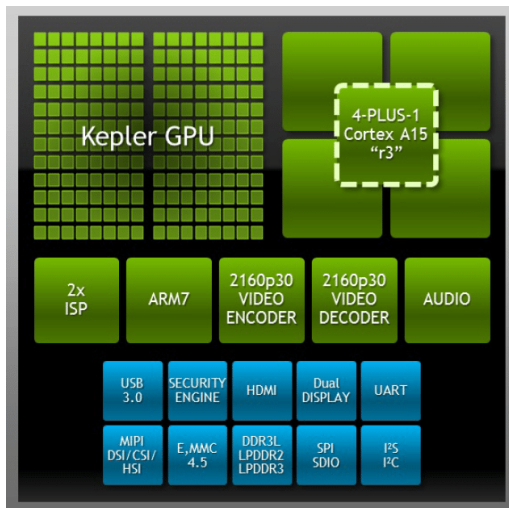
Also, the El Capitan supercomputer (Figure 1) from the Lawrence Livermore National Laboratory, LLNL, designed with AMD EPYC processors, code-named Genoa and Zen 4 processor core, and AMD Radeon Instinct GPUs, is expected to exceed 2 Exaflops [LLNL, 2021].

In addition to these two machines, other Exascale Systems are planned to be delivered in the next few years. The DoE, Department of Energy from the USA, financed national computing facilities centers to receive new systems with distinct architectures from different companies. Figure 2 summarizes the evolution of the subsequent machine installations, starting with Perlmutter in NERSC - Berkeley, Polaris in ALCF - Argonne, Frontier in OLCF - Oak Ridge, and Aurora in ALCF - Argonne. In other countries, there are also Exascale Machines to be delivered like China, with the evolution of the Sunway and the Tianhe, and in Japan, with the new Fugaku.

It is worth mentioning that the architectures of these processors and machines are increasingly heterogeneous, allowing the execution of applications to be processed in the device with the best performance.

### 3.2 Heterogeneous Architectures

The new processors, which are already appearing in the market, have more and more heterogeneous architectures. In HPC systems, processing units with different computing capabilities are combined to provide better performance and energy efficiency when compared to homogeneous systems (e.g., CPU-Only).



**Figure 3.** NVIDIA Tegra K1 chip integrating processors with GPUs in an SoC

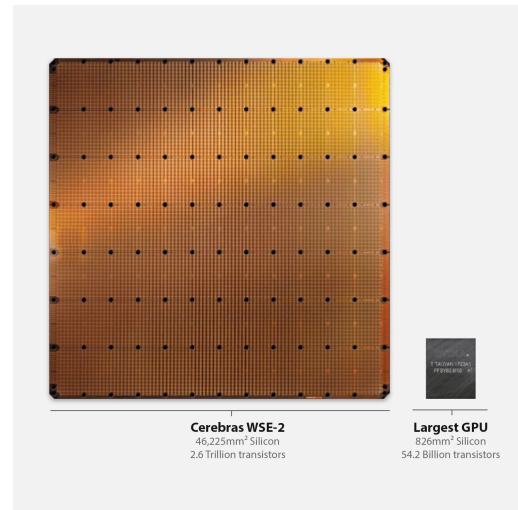
In this scenario, besides traditional CPU-GPU and FPGA-based systems, the following heterogeneous architectures will become popular in the near future in HPC (Figure 5). Heterogeneous memory systems employ a combination of memories, such as DRAM and SRAM, to improve the trade-off between performance and energy consumption in data-intensive workloads. Neural Processing Units, NPUs, are processors designed to speed up AI workloads. Quantum processors, which are in the early stages of development, but with the potential to perform calculations that are exponentially faster than classical computers [Fu *et al.*, 2016].

In the market, one can find systems composed of CPUs and GPUs on the same chip [Dávila *et al.*, 2019], such as those from Intel (Ponte Vecchio) and AMD (APUs). Another possibility is the SoC, System on Chip, where on the same chip (Figure 3), there are processors, accelerator, memory, and I/O system. These chips are generally used in sensors, Internet of Things, IoT, and edge computing environments. In addition to this on-chip option, heterogeneity is possible at the board level, where the processor coexists with the GPU or the FPGA (Intel A10).

In this increasingly heterogeneous environment with different accelerators, the programming model must change from serving a specific type of accelerator to being prepared for a programming environment to suit different accelerators [Vetter *et al.*, 2022]. For example, with the emergence of GPUs from AMD and Intel, new parallel programming interfaces should be used instead of the CUDA language, which is proprietary to NVIDIA.

### 3.3 AI chips

In the upcoming years, AI is set to play a crucial part in many fields, specifically in national and international security [Khan and Mann, 2020]. However, general-purpose AI software, datasets, and algorithms are not effective when running on traditional HPC systems, the focus has shifted towards a computer hardware specialized to execute modern AI applications.



**Figure 4.** Comparison of Cerebras WSE-2 with the largest GPU at the moment.

This hardware is called as “AI chip”, which may include accelerators such as, GPUs, FPGAs, and application-specific integrated circuits, ASICs, that are specialized for AI. Even though general-purpose processors (e.g., CPUs) can be employed to execute simple AI tasks, they are becoming less useful as AI advances. Hence, AI chips have optimized design features that accelerate the calculations required by AI algorithms, e.g., large number of parallel calculations; the use of mixed precision (discussed in Section 5.3); speeding up memory accesses; and providing programming languages to efficiently translate AI code for execution on AI chips.

In this scenario, one of the main challenges is to rightly choose the AI chip that will execute a given AI algorithm. GPUs are often used for the training step. FPGAs efficiently perform the inference phase. On the other hand, ASICs can be used to execute both phases.

At the 2020 SuperComputing conference, two new AI chips were announced, Cerebras [Rocki *et al.*, 2020] and SambaNova [2021]. Cerebras chips employ the second generation of Wafer Scale Engine, WSE-2, a central processor for deep learning and sparse tensor operations that contains 2.6 trillion transistors, 850,000 AI-optimized cores, and 40 gigabytes of high-performance on-wafer memory. Figure 4, extracted from [Cerebras, 2021], compares the Cerebras WSE-2 chip with the largest GPU at the moment.

The SambaNova chip has the Reconfigurable Dataflow Architecture, RDA [SambaNova, 2021], facilitating machine learning and HPC convergence. The RDA provides a flexible dataflow execution model to all types of dataflow computation problems. Furthermore, it does not have a fixed Instruction Set Architecture (ISA) but is programmed for each



model.

In summary, AI chips with specialized hardware devices will popularize in the next few years, which will likely change how AI algorithms are executed. Therefore, one can envision challenges regarding adopting such chips on traditional HPC systems and the emergence of new AI frameworks.

### 3.4 Aware Computing

In recent years, there has been a growth of different types of Aware Computing in HPC systems. When it is applied to the execution of applications, the hardware and software knobs (e.g., number of threads, cores, processor frequency, and the amount of memory used) are adapted according to a given optimization heuristic to the best situation so that the outcome in non-functional metrics (i.e., performance, energy, and power consumption) is improved. Consequently, this situation is changing the complexity of hardware and software behavior.

With the need for reaching more performance with every new generation of supercomputers, techniques like power- and energy-aware computing have become widely used. The reason is that such HPC systems must improve performance without increasing power and energy consumption, so the costs of cooling and electricity infrastructure do not grow. This type of aware computing employs strategies to adapt the hardware and software to keep power and energy demands below a safety threshold. A popular technique widely used in hardware is dynamic voltage and frequency scaling, which automatically manages hardware components' frequency and voltage levels based on their workload usage. Another technique is thread-throttling, where the number of running threads is adjusted according to the thread-level parallelism degree of a given application.

Similarly, since new HPC applications from scientific fields increasingly require more memory space to store data, memory-aware computing is becoming popular. In this type, the design of a computer system is optimized for memory performance, taking into account the increasing gap between processor speed and memory latency. Memory-aware computing involves data compression, prefetching, memory hierarchy optimization, and efficient data placement. This type of computing is particularly challenging for applications that rely on memory, such as big data processing, machine learning, and scientific simulations. One example is the AMD High Bandwidth Memory, HBM, designed to provide high-performance memory for graphics and other memory-intensive applications.

Other types of aware computing are also becoming important in HPC servers. In network-aware computing, the network performance is optimized with load balancing, network topology optimization, and congestion control. Security-aware computing focuses on ensuring the security of systems. Furthermore, one can also find data-aware and user-aware computing, where the former focuses on efficient management of data, and the second is concerned with providing personalized user experiences.

### 3.5 Processing In Memory

Processing In Memory, PIM, is a way to execute instructions in memory without traditionally moving data to the processor [Lee et al., 2021]. With that, the time lost in data transfer is eliminated, being the main benefit of this technique. PIM has been growing in its study, and some chips are appearing on the market. However, even though it can allow data-intensive applications to avoid moving data from memory to CPU, new challenges are introduced for HPC system architects and software developers.

Even though many challenges to designing PIM architectures have been overcome during the last years, some points are still open [Ghose et al., 2019]. One of them relies on how HPC programmers can extract the benefits of PIM without resorting to complex programming models. Another challenge is understanding the constraints of distinct substrates when designing PIM logic. Therefore, developing a PIM programming model, data mapping, and runtime scheduling for PIM are challenging topics that should be addressed before most HPC developers can employ PIM.

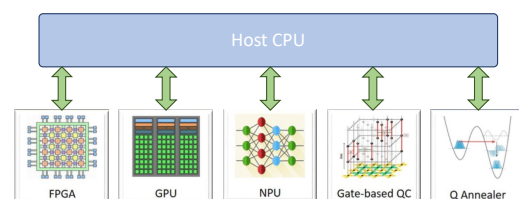
High-tech companies (e.g., Samsung, Micron, and Synopsys) started developing memories with computing capabilities. In the end, they believe that memory and AI computing will converge into the same architecture, making AI-based memory chips.

### 3.6 Quantum Computing

With quantum processors becoming a reality, it is possible to imagine heterogeneous machines with quantum processing units. The prospect of getting atom-sized operators capable of operating using germanium transistor techniques [Hendrickx et al., 2021] is enabling the arrival of Qubits processors on the market. With this, future architectures would have x86 processor units with qubit units (Figure 5).

In this perspective, quantum processors will be used as accelerators at first. They will solve problems, especially in security, cryptography, meteorology, pharmaceuticals, biotechnology, and economic models.

In the subsequent years, quantum computing will not substitute classical computing [Matsuoka et al., 2023]. Instead, Quantum computing will help to solve complex problems that take a lot of time with classical computers. Examples of problems that will benefit from it can be found in modeling protein or molecular simulations, developing new robust encryption, processing data from accelerators, like CERN, and other complex problems.



**Figure 5.** Future of Heterogeneous Architectures with Quantum Processors - Based on Fu et al. [2016]

In recent years, many Quantum machines have been announced, like the Zuchongzhi with 66 qubits, Google with

its 54-qubit Sycamore processor, and IBM's 14th quantum computer model with 53 qubits. However, as scientists preview practical computing, machines need nearly a thousand qubits. In this scenario, IBM has an ambitious goal of building one quantum computer containing 1000 qubits by 2025.

In the end, one of the big challenges of quantum machines is correcting the myriad errors that usually come with quantum operations. So the researchers focus on having a lower error rate on the entire system. Quantum Computing will be the next frontier in the changes in processing capacity, which will allow solving problems in Science that are currently unfeasible to solve on time.

### 3.7 Cloud Computing

With the increasing demand for HPC for several areas, as mentioned in Section 2, the big cloud computing providers started to be interested in providing these facilities. The emergence of instances with more powerful processors, GPUs, FPGAs, and improved interconnection systems within the cloud was observed to allow users to instantiate a set of machines able to meet a demand for greater processing power.

In this scenario, HPC as a Service (HPCaaS) is becoming employed. In Figure 6, extracted from [Paillard et al., 2015] example of infrastructure for executing parallel workloads on the Cloud, where the end-users submit their job through internet, and the cloud environment (HPCaaS) is responsible for allocating machines and deploying the job for execution on HPC systems.

On the websites of the leading cloud providers, there are advertisements such as "High-Performance Computing on AWS Redefines What is Possible," "Cray in Azure - a dedicated supercomputer on your virtual network," "Build your high-performance computing solution on IBM Cloud," "Google Cloud - HPC in the cloud becomes a reality" clearly showing the importance and interest that these companies are giving to offer HPC in the cloud essential and growing part of HPC.

The initial cloud performance issues are gradually being overcome, allowing satisfactory results when running HPC applications in this environment. The increased use of better and faster connections, the availability of new generation processors, and a better storage administration allow enough processing speed to execute complex applications in the cloud. In the end, cloud servers must have a scalable and cost-effective infrastructure for running HPC applications, given their capability of provisioning resources on-demand over the Internet.

On top of that, we also need to take care to Serverless Computing, a new paradigm for developing cloud applications. It is a new level of virtualization, containers, and Function as a Service, FaaS, reducing complexity for the user [Baldini et al., 2017].

### 3.8 Moving to Zettascale Computing

In a paper from Liao et al. [2018], the authors suggested that, by 2035, we will have HPC systems with one Zettascale computing (1021 floating-point operations per second).

They understand that the evolution will increase from machines with 2 to 3 Eflops in 2025 to 2030, with performance scaling to 50–80 Eflops, and the last step will end in 2035 with the Zflops machines.

To reach this level of computing, the authors expected a list of technical metrics in a table presented in Table 1. The computing system will have a power efficiency of 10 Tflops/W for a total power consumption near 100MW for all the systems. Interesting to observe that the performance estimations per node will be 10Pflops and the bandwidth between nodes 1.6 Tb/s. The data for all the system storage capacity will be around 1ZB (Zeta byte).

**Table 1.** Metrics for a Zflops Machine [Liao et al., 2018]

Metric	Value
Peak Performance	1 Zflops
Power consumption	100 MW
Power efficiency	10 Tflops/W
Peak performance per node	10 Pflops/node
Bandwidth between nodes	1.6 Tb/s
I/O bandwidth	10-100 PB/s
Storage capacity	1 ZB
Floor space	1000 m <sup>2</sup>

## 4 Programming Challenges in HPC

This section discusses the impact of design decisions regarding implementing parallel applications that run on top of HPC servers. For that, we start by describing parallel algorithms design patterns and the parallel programming interfaces that can be used to implement to extract the most from the HPC systems. Furthermore, given the importance of memory on the application execution behavior, we discuss techniques to optimize data and thread locality.

### 4.1 Parallel Algorithms Design Patterns

Software developers can employ different communication models when parallelizing applications to ensure the cooperation between the cores that execute concurrently, such as shared memory and message-passing. The former is based on the existence of an address space in the memory that all processors can access. It is widely used when parallelism is exploited at the thread level, as they share the same memory address space. On the other hand, message-passing is employed in environments where the memory space is distributed and/or processes do not share the same memory address space. In summary, the challenge is to use the programming model that best benefits from the target architecture: while shared memory delivers better outcomes for multicore and many-core processors, message-passing is most suitable for large computers that communicate via an interconnection link.

In addition to the communication model, another challenge is choosing the parallel programming style to extract parallelism from sequential codes. For many years, the fork-join model has been the style most used by software developers because of its ease in exploiting parallelism. When

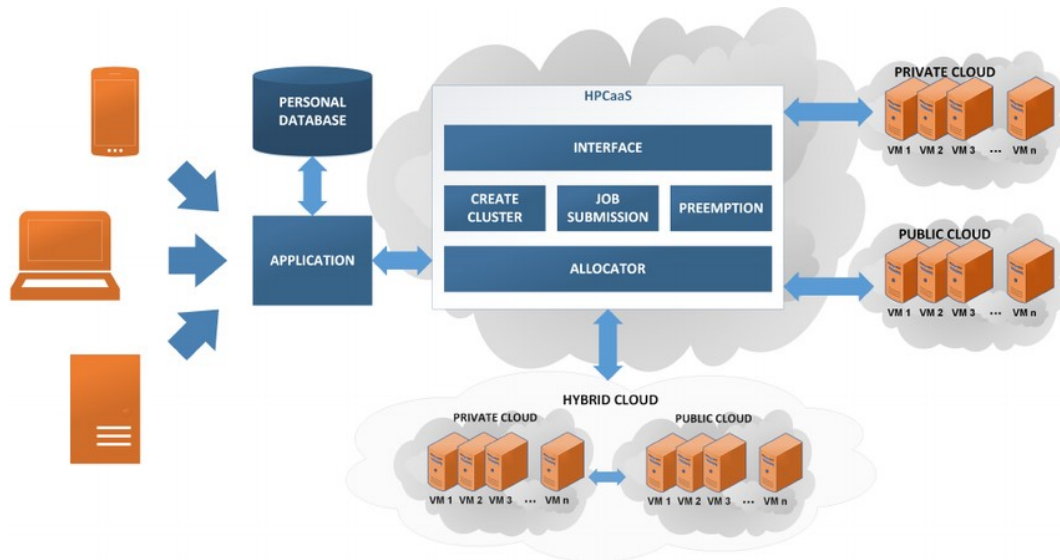


Figure 6. HPC as a Service on Cloud Computing - Paillard et al. [2015]

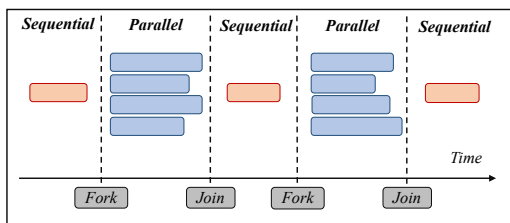


Figure 7. The fork-join shared-memory programming model.

employing the fork-join model, as shown in Figure 7, the master thread (represented by the orange rectangle) starts the execution of the sequential phase. When it reaches a parallel region, a team of threads is created to execute the parallel region concurrently (fork operation). Then, at the end of the region, all threads perform a join operation to synchronize. From this moment on, only the master thread executes the application binary until reaching another parallel region. Considering that this model is widely employed when parallelizing applications that run on top of multicore systems and that processor companies (e.g., AMD, ARM, Intel, and NVIDIA) have been releasing processors with more and more cores, the challenge is the development of algorithms that can get the most out from the architectures. In summary, the main point is to scale the number of threads without losing performance and energy efficiency.

However, the rigid execution model and lack of malleability of parallel applications implemented with the fork-join style may not deal with some hardware and software aspects (e.g., data synchronization and cache contention) when there is variability in the application behavior or execution environment, preventing linear performance improvements. When a scenario like this arises, the rigid fork-join-based implementations can increase power consumption and jeopardize the performance of parallel applications. With that, one can note a popularization of task-based programming models, as discussed in the next section (Section 5), which can overcome fork-join limitations providing more malleability and better load-balancing on multicore systems. However, the challenge software developers face when exploiting task-based

level parallelism is defining the data dependency between different parallel regions, which may limit the gains with parallelization.

Regardless of the programming model, when exploiting parallelism, the software developers can employ design patterns to help in this task, such as Map, Stencil, and Reduction, which are the most used nowadays. The Map pattern, shown in Figure 8, divides the workload (e.g., array, list, or other related collection) into independent parts that can run in parallel with no data dependency, representing a parallelization referred to as embarrassing parallelism Voss et al. [2019]. A function is applied to all collection elements, usually producing a new collection with the same shape as the input. Moreover, the number of iterations and the inputs may be the same and known in advance. For the Map pattern, the challenge with new computer architectures will be the need for a heavy function to be applied to each element of a collection and a high number of iterations to fulfill the processor's cores.

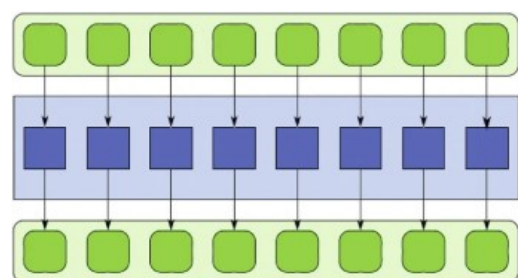


Figure 8. Map pattern example, where a function is applied to all elements of a collection, producing a new collection with the same shape as the input [Voss et al., 2019]

A generalization of the Map is the Stencil pattern, which focuses on combining and applying functions in a set of neighbors from each matrix point [Voss et al., 2019]. Figure 9 exemplifies a stencil operation over one element of the matrix, resulting from the computation of the values on its neighbors. Software developers face a critical challenge in handling boundary conditions in this scenario due to the communication overhead between threads/processes. The Sten-

cil pattern has good performance scalability on GPUs, as they are organized so thousands of threads can be executed simultaneously. In the end, the main challenge will be to balance the memory operations time with the computation time in the way that the loading of neighbors should be faster than the computation applied to them.

There are also many applications where many threads/processes simultaneously apply the same operation over different data. In this case, the Reduction pattern combines every element calculated by each thread/process using an associative function called the combined function [Voss et al., 2019]. Figure 10 shows an example of a reduction operation executing in parallel, where pairs of elements are calculated simultaneously until the final result is reached. Even though it achieves satisfactory performance results when combining values computed by each thread, its implementation gets more complex as the number of reductions after each iteration changes at every step until the final result is reached. Therefore, software developers face the challenge of coordinating the computation during each reduction operation to maximize the usage of hardware resources.

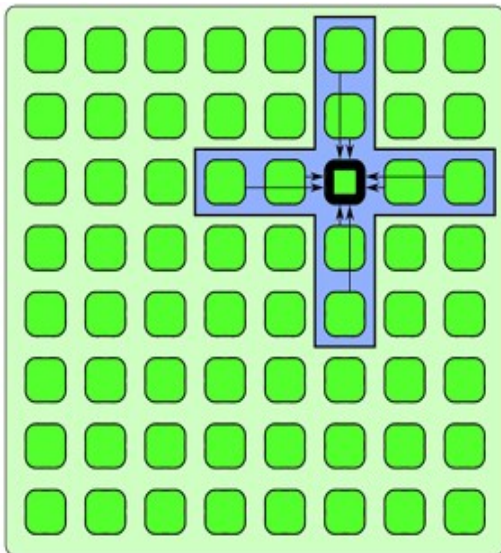


Figure 9. Stencil Computation Example [Voss et al., 2019]

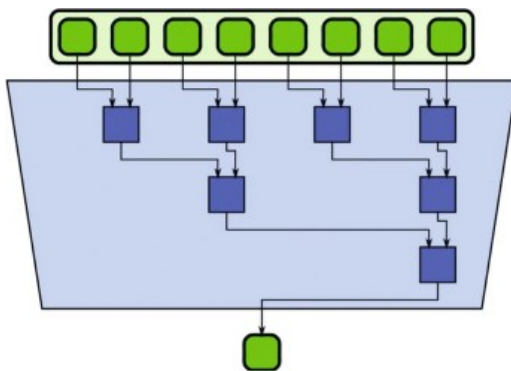


Figure 10. Parallel reduction, where threads produce subresults that are combined to produce a final single answer [Voss et al., 2019]

## 4.2 Parallel Programming Libraries

With the emergence of modern architectures that present different computing capabilities, many programming languages and libraries to help software developers exploit parallelism were introduced in the literature. We illustrate in Figure 11 the evolution of the number of citations on the Scopus database of the most used parallel programming languages over the years and discuss each one next.

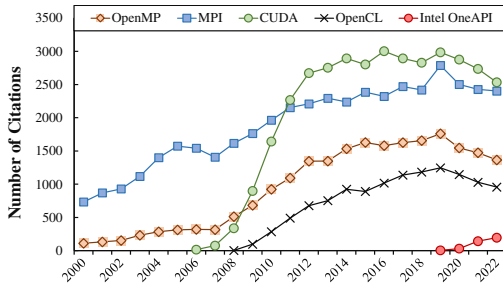
As observed, Message-Passing Interface, MPI, has established itself over the years as one of the main alternatives for exploiting parallelism. This popularity happens due to the need for a library to create processes and manage communication between them in distributed memory environments, which is found in every HPC system. Given that, along with the evolution of MPI [Gabriel et al., 2004] (e.g., MPI-1, MPI-2, and MPI-3) and in the technology available in HPC systems, the challenges are related to (i) balance communication and computation between computing nodes to better use hardware resources; (ii) efficiently use asynchronous communication to overlap communication and computation; and (iii) ensure fault tolerance mechanisms.

Simultaneously, with the increase in the number of cores in multicore architectures, there was a popularization in the use of libraries that exploit parallelism in shared memory (for example, OpenMP and POSIX Threads). The POSIX Threads library, a lightweight thread implementation, was the first to create parallel applications [Barney, 2009]. However, due to the complexity and the need for the programmer to decide different aspects of the operating system, it is no longer used for this purpose. It continues to be used only for developing applications aimed at the operating system.

In this scenario, the OpenMP library replaced the POSIX Threads library. OpenMP consists of compiler directives, library functions, and environment variables that ease the burden of managing threads in the code [Chandra et al., 2001]. Therefore, extracting parallelism using OpenMP usually requires less effort when compared to POSIX Threads. Parallelism is exploited by inserting directives in the code that inform the compiler how and which parts of the application should be executed in parallel. With the increasing number of cores and complexity of the memory hierarchy in shared-memory architectures, the challenge when using OpenMP will be related to (i) finding the optimal number of threads to execute each parallel region; (ii) defining thread and data placement strategies that reduce cache contention and last-level cache misses; and (iii) efficiently employ directives for heterogeneous computing. Furthermore, other libraries, such as Cilk Plus [Schardl et al., 2018], developed by MIT and later maintained by Intel, emerged to facilitate vector and task programming; however, due to the evolution of the OpenMP library, they also ceased to be used. In the last years, OmpSs-2 has emerged as an alternative to OpenMP when exploiting parallelism through tasks by providing more malleability and better load-balancing on multi-core systems.

The scenario dominated by OpenMP and MPI drastically changes with the popularization of GPU architectures. From then on, CUDA became one of the most used parallel programming interfaces to accelerate multi-domain applications on NVIDIA GPUs [Sanders and Kandrot, 2010].





**Figure 11.** Number of citations from the foremost parallel programming libraries over the years in the Scopus database.

The CUDA library consists of a set of extensions for C, C++ and FORTRAN, which allows the programmer to create kernels, which are functions that can run on NVIDIA GPU-type graphics cards [Cook, 2012]. In addition to this library, OpenCL also emerged as a framework for heterogeneous computing, allowing the software developer to implement applications that run on both multicore processors and graphics cards from any vendor [Munshi et al., 2011]. Finally, the OpenACC library has become popular as it looks similar to the OpenMP library when exploiting parallelism [Farber, 2016]. It also uses pragmas and makes programming for GPUs fast and straightforward.

Considering the imminent increase in the use of heterogeneous architectures by end users, Intel released OneAPI at the end of 2018. It simplifies software development by providing the same programming languages and models across all accelerator architectures. oneAPI seeks to provide software developers with source-level compatibility, performance transparency, and software stack portability. In this scenario, the challenge will be defining the ideal architecture to execute each piece of parallel code without jeopardizing the performance of the entire HPC system.

Current and efficient parallel applications employ more than one parallel programming interface. The MPI library is generally used for message exchange between different computational nodes while the OpenMP and CUDA / OpenACC libraries exploit the use of multiple processor cores and GPU-type graphics cards. The trend is the emergence and evolution of compilers, both in self-parallelization and data location optimization.

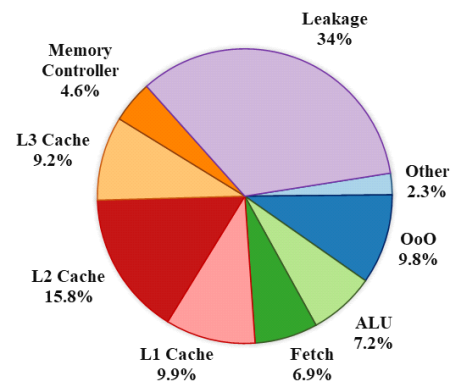
## 5 Other Challenges

Given the increasing heterogeneity of hardware resources and availability of frameworks and libraries to exploit parallelism in HPC servers shown in the last sections, other challenges become important to be addressed. Hence, we discuss in this section, the trends related to energy consumption, that is, what should one care about in HPC systems to improve their energy efficiency. Similarly, new HPC environments will require more resilience to reduce the amount of software and hardware failures, as discussed in Section 5.2. We also discuss two alternatives to optimize the energy efficiency of HPC servers, such as mixed-precision and data locality in Section 5.3 and 5.4, respectively.

## 5.1 Energy Demand

HPC machines' growing demand for processing capabilities has led manufacturers to associate thousands of processors in clusters, which generates high power consumption. In this scenario, some machines from the TOP500 list consume around 30 MW, corresponding to a city of approximately 300,000 inhabitants. Consequently, machine builders and processor manufacturers aim to optimize architectures so that consumption decreases. These optimizations include changing the processor architecture, managing non-active units, and reducing the processor operating frequency, among other techniques [Padoin et al., 2019]. Today, new machines are made to increase instruction speed and energy consumption, which sometimes occurs the opposite.

Figure 12, extracted from a modern multicore processor with WattWatcher tool when executing well-known parallel workloads [LeBeane et al., 2015], clearly demonstrates one of the challenges for the evolution of processor architectures. It is verified that the percentage of energy dedicated to the effective processing and execution of the instruction is about 17% of all energy (OoO and ALU). In comparison, about 34% is spent on static energy, leakage, the energy the circuit consumes, even with nothing running. Furthermore, about 35% is spent on accesses to different levels of cache memory access and 11% on registers. In the end, even if these percentages can improve during the next few years, it appears that the energy invested in the final goal, which is the execution of the instruction, is small compared to the point spent in other parts of the processor's operation. Therefore, this is a significant challenge to be improved in future processor architectures.



**Figure 12.** Distribution of energy consumption in a core.

## 5.2 Resilience

Resilience deals with the ability of a system to continue operating in the presence of failures or performance fluctuations. In supercomputers, which process high-performance applications and have thousands of cores, memories, and circuits connecting them, the probability of a failure in any of its elements becomes more prominent. Therefore, resilience is one of the challenges to be faced in order to continue delivering high-performance execution of scientific applications while keeping infrastructure-related costs low.

Analysis over the last years shows that newer HPC systems will be much less reliable as a consequence of three main factors: the increasing complexity of the system design and number of hardware resources (e.g., cores and memories); the decreasing device dependability; and the shrinking process technology. Figure 13 shows the mean time between failures (MTBF) of distinct HPC servers with different number of nodes and cores per node, extracted from [Gupta et al., 2017]. As can be observed, the greater the number of nodes in the HPC system, the shorter the mean time (in hours) between failures. This scenario means that newer supercomputers with thousands of processors and nodes will have servers crashing every day, hence the importance of resilience to keep the machine running.

Analysis over time shows the growth of failures with the evolution of supercomputers, with more and more cores reaching thousands of them. Figure 13 shows this evolution, which means that a supercomputer with thousands of processors will have servers crashing every day, hence the importance of resilience to keep the machine running. Resilience is achieved through hardware and software that will dynamically detect the failure, diagnose, reconfigure, and repair it, allowing processing to continue without user awareness. Therefore, this area becomes essential in future machines to meet the needs of high-performance processing and will enable this processing to occur until obtaining results without interruption.

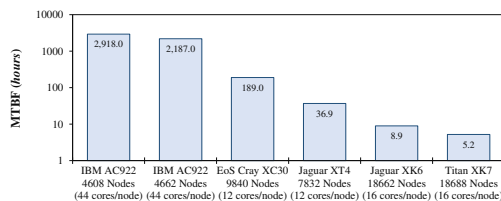


Figure 13. Mean time between failures on different HPC servers.

### 5.3 HPC and Mixed Precision

The amount of energy and time spent with operations with high precision induce researchers to optimize operations considering the correct need of precision in each calculus. Lowering the precision can save cost, time and energy. The proper reduction of accuracy is the new challenge in application execution.

Mixed-precision architectures usually support two or more floating-point precision arithmetic operations and allow the reduction of both storage, energy, and computational requirements. By reducing the precision of some data and arithmetic operations of the problems, it is possible to trade-off the quality of the result by the performance and energy efficiency of the execution [Freytag et al., 2022].

Other than the precision reduction, there is Approximate Arithmetic that works with simplified arithmetic's units, that are less expensive on area and so on energy, whose results are less precise.

### 5.4 Data Locality

In future microprocessors, where the memory hierarchy will become more complex, the energy spent only to move data will critically affect the application's performance and energy consumption. Hence, any nano-joule employed to move data up and down the memory hierarchy will decrease the energy available for computation. In this scenario, task mapping and scheduling need to be optimized in the interconnection network, prioritizing location to restrict data movement as much as possible [Cruz et al., 2021]. Therefore, the tendency is to prioritize data locality over processor speed, although local data tends to aid in faster processing. Energy conservation becomes the priority.

Figure 14, extracted from the DoE report [Vetter et al., 2022], shows that despite the evolution of chip technology, with increasingly thinner technologies reaching 7nm, the decrease in energy consumption of the entire chip does not keep the same pace with the reduction of energy due to computing. This evolution shows that energy consumption for moving data spends more power than performing computing operations on the chip, which implies an important revolution, not only in the architecture of processors but also in the way instructions are executed. Compilers should be concerned with bringing data closer to the processors.

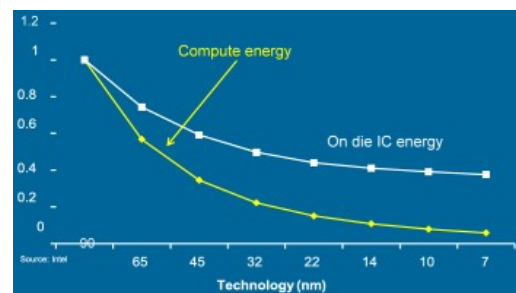


Figure 14. Energy consumption at Pico Joules versus technology evolution Vetter et al. [2022]

## 6 Conclusion

As discussed in the text above, High-Performance Processing has gone from a specific area, meeting certain processing needs, to a central theme in the evolution of computing, considering the growing needs of processing power in science fields, such as Big Data, Artificial Intelligence, Data Science, among others. This evolution goes through important transformations of machines and processors, including the growing use of the cloud and cloud to meet these demands for processing power. The search for greater performance is not always the main priority; today, there is often an attempt to optimize energy consumption. Heterogeneity is an integral part of processors and machines, and the arrival of quantum processors should increase this diversity. Resilience is crucial for new HPC systems since system failures, errors, or interruptions can result in data loss, delays, or system downtime, leading to significant financial losses, productivity declines, or even safety risks in critical applications. Furthermore, new forms of programming and storage are an essen-

tial part of the HPC development. It is concluded that the evolution of computing involves the continuous growth of processing power and new ways of doing it.

## Declarations

### Authors' Contributions

All authors contributed to the writing of this article, read and approved the final manuscript.

### Competing interests

The authors declare that they have no competing interests.

### Funding

This study was partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, by Petro bras grant n.º 2: 020/00182-5, by CNPq/MCTI/FNDCT n.º 308877/2022-5 and grant Universal 18/2021 n.º 406182/2021-3.

### Availability of data and materials

Data can be made available upon request.

## References

- Aurora (2021). Argonne leadership computing facility. Available at: <https://www.alcf.anl.gov/aurora>. Accessed: Apr. 11, 2021.
- Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., et al. (2017). Serverless computing: Current trends and open problems. *Research advances in cloud computing*, pages 1–20. DOI: 10.1007/978-981-10-5026-8<sub>1</sub>.
- Barney, B. (2009). POSIX threads programming. National Laboratory. Available at: <https://computing.llnl.gov/tutorials/pthreads>. Accessed: Mai. 4, 2022.
- Cerebras (2021). The future of ai is here. Available at: <https://cerebras.net/chip/>. Accessed: Sep. 10, 2021.
- Chandra, R., Dagum, L., Menon, R., Kohr, D., Maydan, D., and McDonald, J. (2001). *Parallel programming in OpenMP*. Morgan Kaufmann.
- Cook, S. (2012). *CUDA programming: a developer's guide to parallel computing with GPUs*. Newnes.
- Cox, M. and Ellsworth, D. (1997). Managing big data for scientific visualization. In *ACM siggraph*, volume 97, pages 21–38. MRJ/NASA Ames Research Center. Available at: [https://www.researchgate.net/profile/David-Ellsworth-2/publication/238704525\\_Managing\\_big\\_data\\_for\\_scientific\\_visualization/links/54ad79d20cf2213c5fe4081a/Managing-big-data-for-scientific-visualization.pdf](https://www.researchgate.net/profile/David-Ellsworth-2/publication/238704525_Managing_big_data_for_scientific_visualization/links/54ad79d20cf2213c5fe4081a/Managing-big-data-for-scientific-visualization.pdf).
- Cruz, E. H., Diener, M., Pilla, L. L., and Navaux, P. O. (2021). Online thread and data mapping using a sharing-aware memory management unit. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, 5(4):1–28. DOI: 10.1145/3433687.
- Dávila, G. P., Oliveira, D., Navaux, P., and Rech, P. (2019). Identifying the most reliable collaborative workload distribution in heterogeneous devices. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1325–1330. IEEE. DOI: 10.23919/DATE.2019.8715107.
- Desjardins, J. (2019). How much data is generated each day? Available at: <https://www.visualcapitalist.com/how-much-data-is-generated-each-day>. Accessed: Mar. 12, 2021.
- Dongarra, J. H. M. and Strohmaier, E. (2020). Top500 supercomputer. Available at: <https://www.top500.org/lists/top500/2020/11/>. Accessed: Mar. 10, 2021.
- Farber, R. (2016). *Parallel programming with OpenACC*. Newnes.
- Freytag, G., Lima, J. V., Rech, P., and Navaux, P. O. (2022). Impact of Reduced and Mixed-Precision on the Efficiency of a Multi-GPU Platform on CFD Applications. In *Computational Science and Its Applications—ICCSA 2022 Workshops: Malaga, Spain, July 4–7, 2022, Proceedings, Part IV*, pages 570–587. Springer. DOI: 10.1007/978-3-031-10542-5<sub>39</sub>.
- Frontier (2021). ORNL Exascale Supercomputer. Available at: <https://www.olcf.ornl.gov/frontier/>. Accessed: Apr. 10, 2021.
- Fu, X., Riesebois, L., Lao, L., Almudever, C. G., Sebastiano, F., Versluis, R., Charbon, E., and Bertels, K. (2016). A heterogeneous quantum computer architecture. In *Proceedings of the ACM International Conference on Computing Frontiers*, pages 323–330. DOI: 10.1145/2903150.2906827.
- Fujitsu (2021). Supercomputer fugaku. Available at: <https://www.fujitsu.com/>. Accessed: Apr. 10, 2021.
- Gabriel, E., Fagg, G. E., Bosilca, G., Angskun, T., Dongarra, J. J., Squyres, J. M., Sahay, V., Kambadur, P., Barrett, B., Lumsdaine, A., et al. (2004). Open mpi: Goals, concept, and design of a next generation mpi implementation. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface: 11th European PVM/MPI Users' Group Meeting Budapest, Hungary, September 19–22, 2004. Proceedings II*, pages 97–104. Springer. DOI: 10.1007/978-3-540-30218-6<sub>19</sub>.
- Ghose, S., Boroumand, A., Kim, J. S., Gómez-Luna, J., and Mutlu, O. (2019). Processing-in-memory: A workload-driven perspective. *IBM Journal of Research and Development*, 63(6):3–1. DOI: 10.1147/JRD.2019.2934048.
- Gupta, S., Patel, T., Engelmann, C., and Tiwari, D. (2017). Failures in large scale systems: long-term measurement, analysis, and implications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12. DOI: 10.1145/3126908.3126937.
- Hendrickx, N. W., Lawrie, W. I., Russ, M., van Riggelen, F., de Snoo, S. L., Schouten, R. N., Sammak, A., Scap-

- pucci, G., and Veldhorst, M. (2021). A four-qubit germanium quantum processor. *Nature*, 591(7851):580–585. DOI: 10.1038/s41586-021-03332-6.
- Kelleher, J. D. and Tierney, B. (2018). *Data science*. MIT Press.
- Khan, S. M. and Mann, A. (2020). Ai chips: what they are and why they matter. *Center for Security and Emerging Technology*. DOI: 10.51593/20190014.
- LeBeane, M., Ryoo, J. H., Panda, R., and John, L. K. (2015). Watt watcher: fine-grained power estimation for emerging workloads. In *2015 27th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 106–113. IEEE. DOI: 10.1109/SBAC-PAD.2015.26.
- Lee, S., Kang, S.-h., Lee, J., Kim, H., Lee, E., Seo, S., Yoon, H., Lee, S., Lim, K., Shin, H., et al. (2021). Hardware architecture and software stack for pim based on commercial dram technology: Industrial product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 43–56. IEEE. DOI: 10.1109/ISCA52012.2021.00013.
- Liao, X.-k., Lu, K., Yang, C.-q., Li, J.-w., Yuan, Y., Lai, M.-c., Huang, L.-b., Lu, P.-j., Fang, J.-b., Ren, J., et al. (2018). Moving from exascale to zettascale computing: challenges and techniques. *Frontiers of Information Technology & Electronic Engineering*, 19:1236–1244. DOI: 10.1631/FITEE.1800494.
- LLNL (2021). DOE/NNSA Lab announces a partnership with Cray to develop NNSA’s first exascale supercomputer. Jeremy Thomas. Available at: <https://www.llnl.gov/news/>. Accessed: Sep. 10, 2021.
- Matsuoka, S., Domke, J., Wahib, M., Drozd, A., and Hoefler, T. (2023). Myths and legends in high-performance computing. *arXiv preprint arXiv:2301.02432*. DOI: 10.48550/arXiv.2301.02432.
- Munshi, A., Gaster, B., Mattson, T. G., and Ginsburg, D. (2011). *OpenCL programming guide*. Pearson Education.
- Padoin, E. L., Diener, M., Navaux, P. O., and Méhaut, J.-F. (2019). Managing power demand and load imbalance to save energy on systems with heterogeneous CPU speeds. In *2019 31st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 72–79. IEEE. DOI: 10.1109/SBAC-PAD.2019.00024.
- Paillard, G. A. L., Coutinho, E. F., de Lima, E. T., and Moreira, L. O. (2015). An architecture proposal for high performance computing in cloud computing environments. In *4th International Workshop on Advances in ICT Infrastructures and Services (ADVANCE 2015), Recife*. Available at: [https://www.researchgate.net/profile/Emanuel-Coutinho/publication/293481549\\_An\\_Architecture\\_Proposal\\_for\\_High\\_Performance\\_Computing\\_in\\_Cloud\\_Computing\\_Environments/links/56b897a708ae3c1b79b2dff5/An-Architecture-Proposal-for-High-Performance-Computing-in-Cloud-Computing-Environments.pdf](https://www.researchgate.net/profile/Emanuel-Coutinho/publication/293481549_An_Architecture_Proposal_for_High_Performance_Computing_in_Cloud_Computing_Environments/links/56b897a708ae3c1b79b2dff5/An-Architecture-Proposal-for-High-Performance-Computing-in-Cloud-Computing-Environments.pdf).
- Reed, D., Gannon, D., and Dongarra, J. (2022). Reinventing high performance computing: Challenges and opportunities. *arXiv preprint arXiv:2203.02544*. DOI: 10.48550/arXiv.2203.02544.
- Rocki, K., Van Essendelft, D., Sharapov, I., Schreiber, R., Morrison, M., Kibardin, V., Portnoy, A., Dietiker, J. F., Syamlal, M., and James, M. (2020). Fast stencil-code computation on a wafer-scale processor. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14. IEEE. DOI: 10.1109/SC41405.2020.00062.
- SambaNova (2021). Accelerated computing with a reconfigurable dataflow architecture. white paper. Available at: <https://sambanova.ai/>. Accessed: Sep. 10, 2021.
- Sanders, J. and Kandrot, E. (2010). *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional.
- Schardl, T. B., Lee, I.-T. A., and Leiserson, C. E. (2018). Brief announcement: Open cilk. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*, pages 351–353. DOI: 10.1145/3210377.3210658.
- Stevens, R., Taylor, V., Nichols, J., Maccabe, A. B., Yelick, K., and Brown, D. (2020). AI for science: Report on the department of energy (doe) town halls on artificial intelligence (ai) for science. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States).
- Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., and Rellermeier, J. S. (2020). A survey on distributed machine learning. *Acm computing surveys (csur)*, 53(2):1–33. DOI: 10.1145/3377454.
- Vetter, J. S., Brightwell, R., Gokhale, M., McCormick, P., Ross, R., Shalf, J., Antypas, K., Donofrio, D., Humble, T., Schuman, C., et al. (2022). Extreme heterogeneity 2018-productive computational science in the era of extreme heterogeneity: Report for DOE ASCR workshop on extreme heterogeneity. DOI: 10.2172/1473756.
- Voss, M., Asenjo, R., Reinders, J., Voss, M., Asenjo, R., and Reinders, J. (2019). Mapping parallel patterns to TBB. *Pro TBB: C++ Parallel Programming with Threading Building Blocks*, pages 233–248. DOI: 10.1007/978-1-4842-4398-5\_8.
- Xenopoulos, P., Daniel, J., Matheson, M., and Sukumar, S. (2016). Big data analytics on HPC architectures: Performance and cost. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 2286–2295. IEEE. DOI: 10.1109/BigData.2016.7840861.