Challenges in High-Performance Computing

西北工業大學

Research Report

得 分：

Student ID **2024280084**

Name **Mohammed Tanjim uddin**

Course Name **High Performance Computing**

Date **01/22/2025**

西北工业大学研究生院

Challenges in High-Performance Computing

Abstract

High-performance computing (HPC) has emerged as one of the most productive areas of computer science. The state-of-the-art algorithms from various fields, including Big Data, Artificial Intelligence, data analysis, and areas associated with physical sciences, chemistry, and biological sciences, are typically demanding on technology, primarily due to their high processing capabilities. Consequently, it is crucial to select the appropriate computer system to maximise their effectiveness. This article explores the primary obstacles that will be encountered in the development of future supercomputer systems. It emphasises the areas that require the most HPC servers, such as the adoption of HPC on cloud servers, the development of new architectures, the use of heterogeneous processors based on artificial intelligence chips, and the difficulties that software developers encounter when parallelising applications.

1. Introduction

High-Performance Computing (HPC) is a recognised domain that concentrates on supercomputers and other systems with exceptionally high processing capabilities. **Dongarra and Strohmaier (2020)** indicates that annually, two selections are routinely made to identify the computers globally that have achieved the maximum processing power.

The HPC landscape has evolved in recent years due to the processing power demands of Artificial Intelligence (AI), neural networks (ML), Deep Learning (DL), and other algorithms, alongside the need for this power to be applicable in data learning within Big Data contexts **[Verbraeken et al., 2020].**

High-performance computing addresses not just the requirements of individual scientists in fields such as physics, chemistry, and biology but also the broader expectations of society.

A more substantial alteration has occurred. Cloud providers are investing in vast global links of massively parallel machines for high-performance computing (HPC) **[Reed et al., 2022].** A substantial shift is occurring from simultaneous application performances on specialised HPC servers to cloud infrastructures. This transition is propelled by reasons like as being able to access and the capacity to offer high-performance designs on demand while sustaining low operational expenses. Over the years, hardware and software approaches have been presented to facilitate the persistent execution of HPC workloads in cloud settings.

Moreover, HPC environments are becoming progressively diversified to satisfy the performance and energy demands associated with applications in scientific fields. In this context, most supercomputers utilise hardware accelerators and techniques that exhibit diverse computing power, including **graphical processing units (GPUs)**, **field-programmable gate arrays (FPGAs), processing in memory (PIM),** and the recent incorporation of quantum processors. As a result, many parallel programming libraries and patterns are developed each year to enhance performance from these designs.

2. The Need for HPC

Researchers will soon face problems in the **HPC** sector related to hardware and software designed to enhance application execution in terms of performance, energy efficiency, and resilience. Moreover, there are developments linked to the increasing accessibility of heterogeneous architectures in **HPC** systems and methodologies to optimize the efficacy of each device.

2.1 Demands from Big Data area

In every two years, the quantity of data produced doubles and continues to expand exponentially **[Desjardins, 2019].** This progression brings us to the **Yotta data level**, equivalent to **1024 bytes** or **10008 zettabytes**. This volume of data, referred to as **Big Data**, originates from various sources (e.g., sensors inside the Internet of Things) and requires processing by unconventional systems for manipulation, analysis, and information extraction from the dataset. Nonetheless, merely approximately 20% of the recovered information would prove beneficial.

Consequently, Big Data comprises extensive and intricate datasets that are challenging to manage with conventional database technologies. It frequently comprises a compilation of legacy data. Alongside the necessity for innovative data management strategies, the Big Data domain requires substantial computational capacity for data processing.

Moreover, we must acknowledge that data consist of numbers and codes devoid of interpretation. Information, conversely, constitutes processed data. The significance of the data will be determined by how it is processed. Ultimately, knowledge is the ability to gather information about a particular subject and apply it in a practical manner.

Challenges in High-Performance Computing

Additionally, it is crucial to note that information equals power in today's world. The capacity to analyses and make decisions was always present; however, the emergence of significant volumes of data and the capacity to process them made it feasible.
Organisations that possess data and possess the capability to process and analyse it possess a capacity much surpassing that of governments. Moreover, there are no boundaries for data, and these organisations are acquiring data from nearly every part of the globe. In conclusion, high-performance machines, like supercomputers, are essential for data storage and retrieval in the Big Data approach.

2.2 Challenges from the Artificial Intelligence Area

As per the "AI for Science" report by the US Department of Energy [**Stevens et al., 2020**], the ongoing development and growth of science infrastructure through Exascale systems will necessitate the implementation of new artificial intelligence techniques. **Data analysis methods**, simulations with high-performance computing, machine learning, and the collective wisdom of the scientific community have opened up exciting new avenues for research and discovery, as well as stronger strategies for accelerating scientific progress and its positive social impacts.
The integration of high-performance computing with artificial intelligence enables simulation environments to utilise deep reinforcement learning across diverse challenges, including the simulation of robotics, aircraft, and autonomous vehicles. Deep learning techniques enhance simulations by substituting models in climate forecasting, geoscience, pharmaceuticals, and other fields. Advancements in physics are being achieved by the enhanced application of **Partial Differential Equations (PDE)** in conjunction **with Deep Learning (DL)** for simulations.
Conversely, a significant impediment in employing ML and DL algorithms is the pre-utilization learning phase. This activity may require weeks or even months, based upon the available computational infrastructure. This is where high-performance processing expedites this phase.
Ultimately, in response to the high-performance computing requirements of the artificial intelligence sector, companies are investing in parallel frameworks to enhance machine learning platforms, which offer tools and libraries for seamless code deployment across heterogeneous devices. Notable examples include **PyTorch,** a GPU-accelerated tensor computation framework, and **TorchANI**, an implementation of an **Accurate Neural Network Engine** for Molecular Energies. The Intel Neural Compressor tool assists software developers in efficiently deploying inference solutions on widely-used deep learning frameworks, such as **TensorFlow** and **PyTorch**.

Challenges in High-Performance Computing

Consequently, a problem that software developers will have in the integration of HPC with AI is the optimal utilisation of available hardware to maximise the potential of all offered frameworks. Concurrently, hardware vendors have the formidable challenge of building optimised processors for AI computation and executing AI applications. **NVIDIA** offers many solutions that enable consumers to utilise GPUs optimised for AI algorithms. TensorFlow, an open-source framework.

2.3 Needs in Data Science and Associated Fields

HPC and data science uses are now more similar than ever because of improvements in both supercomputing and data generation technologies. According to **Xenopoulos et al. (2016)**, scalable computational methods and analytics, efficient methods to regulate enormous amounts of data, and large-scale data generation technologies have all come together to form scalable Data Science, which is driving innovation and discovery in the scientific community.

Data science is a discipline that integrates various domains, such as statistical analysis, mathematics, artificial intelligence, machine learning, and confined computing, with a subject matter to reveal insights concealed inside an organization's data.

The developmental process of data science typically entails the following steps: data collection, data storage and processing, data analysis, and the presentation of analyses and data visualizations that illuminate the insights [**Kelleher and Tierney, 2018**]. This process is facilitated by tools, roles, and procedures.

All processes are expedited by the utilization of HPC systems, yielding insights advantageous to society, including fraud and risk identification, search engines, enhanced picture recognition, speech recognition, and airline route optimization. In summary, despite employing HPC to enhance the execution of data science algorithms, a primary difficulty in the field is the efficient utilization of hardware resources to minimize the training costs of learning algorithms.

Challenges in High-Performance Computing

3. High-Performance Computing Architectures and Processors Challenges

This part will talk about the main hardware problems that HPC needs to solve in order obtain findings on time in these areas.

3.1 Next Generation of Processors and Accelerators

Recent suggestions for parallel architectures, shown in **Figures 1 and 2**, have the potential to significantly impact the processor market. We will now identify the four supercomputers with the greatest computational capacity at present.
The 64th edition of the TOP500 reveals that **El Capitan** has achieved the top spot and is officially the third system to reach exascale computing after Frontier and Aurora, JUNE-2024. Both systems have since moved down to No. 2 and No. 3 spots, respectively. Additionally, new systems have found their way onto the Top 10.
The A64FX CPU from ARM enabled Fujitsu's FUGAKU computer with NVIDIA GPUs to top the TOP500 list for two years (2020-2021), achieving 442 Petaflops (Figure 1).

Following the breach of the exaflop performance threshold, it is expected that the quantity of supercomputers attaining this level of performance would rise. Aurora (Figure 1) is a supercomputer being developed at Argonne's laboratory. It is being powered by Intel processors that utilize the Ponte Vecchio architecture, which incorporates the Xeon processor and the Xe accelerator on a single chip [Aurora, 2021].

Oak Ridge National Laboratory
U.S. Department of Energy
Cray
AMD
FRONTIER
Argonne National Laboratory
U.S. Department of Energy
intel
Hewlett Packard Enterprise
Aurora
U.S. Department of Energy
NNSA
Lawrence Livermore National Laboratory
AMD
Cray
EL CAPITAN

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

Besides these two machines, other Exascale Systems are scheduled for delivery in the forthcoming years. The Department of Energy (DoE) of the United States funded national computing facility centers to acquire new systems featuring diverse designs from various vendors. **Figure 2** delineates the progression of succeeding machine deployments, commencing with Perlmutter at NERSC - Berkeley, followed by Polaris at ALCF - Argonne, Frontier at OLCF - Oak Ridge, and Aurora at ALCF - Argonne. Other countries, such as China with the advancements of the Sunway and Tianhe, and Japan with the new Fugaku, are also set to deliver Exascale Machines.
The designs of these processors and systems are becoming increasingly diverse, enabling programs to be executed on the device that offers optimal performance.

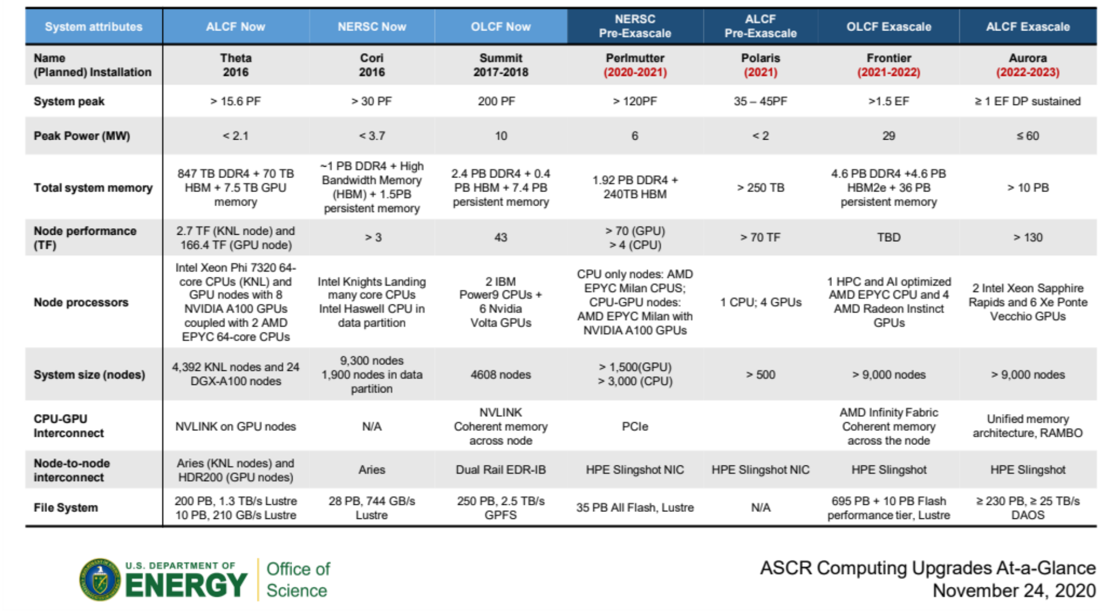
3.2 The Upcoming Generation of Accelerators and Processors

The processor market has the potential to be substantially influenced by the recent recommendations for parallel architectures, as illustrated in **Figures 1 and 2**.

After the exaflop performance threshold is broken, more supercomputers should reach it. Argonne's Aurora supercomputer (Figure 1) is being created. It is powered by Intel Ponte Vecchio processors, which combine the Xeon CPU and Xe accelerator on a chip [**Aurora, 2021**].

Additional Exascale Systems will be delivered in the future years. US Department of Energy (DoE) funds national computing facility centers to buy new systems with varied designs from multiple manufacturers. Figure 2 shows the machine deployments from Perlmutter at NERSC-Berkeley to Polaris, Frontier, and Aurora at ALCF-Argonne. Others will deliver Exascale Machines, including China with the Sunway and Tianhe and Japan with the Fugaku.

Increasingly diversified processor and system designs allow applications to be executed on the best-performing device.



System attributes	ALCF Now	NERSC Now	OLCF Now	NERSC Pre-Exascale	ALCF Pre-Exascale	OLCF Exascale	ALCF Exascale
Name (Planned) Installation	Theta 2016	Cori 2016	Summit 2017-2018	Perlmutter (2020-2021)	Polaris (2021)	Frontier (2021-2022)	Aurora (2022-2023)
System peak	> 15.6 PF	> 30 PF	200 PF	> 120PF	35 – 45PF	>1.5 EF	≥ 1 EF DP sustained
Peak Power (MW)	< 2.1	< 3.7	10	6	< 2	29	≤ 60
Total system memory	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
Node performance (TF)	2.7 TF (KNL node) and 166.4 TF (GPU node)	> 3	43	> 70 (GPU) > 4 (CPU)	> 70 TF	TBD	> 130
Node processors	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
System size (nodes)	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
CPU-GPU Interconnect	NVLINK on GPU nodes	N/A	NVLINK Coherent memory across node	PCIe		AMD Infinity Fabric Coherent memory across the node	Unified memory architecture, RAMBO
Node-to-node interconnect	Aries (KNL nodes) and HDR200 (GPU nodes)	Aries	Dual Rail EDR-IB	HPE Slingshot NIC	HPE Slingshot NIC	HPE Slingshot	HPE Slingshot
File System	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

U.S. DEPARTMENT OF ENERGY | Office of Science

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

Figure 2. Evolution of supercomputers

3.3 Heterogeneous Architectures

New processors have heterogeneous architectures. Unlike uniform systems (e.g., CPU-Only), high-performance computing systems incorporate multiple processing units with different computing capabilities to improve performance and energy efficiency. Besides **CPU-GPU** and **FPGA-**based systems, heterogeneous architectures are projected to become popular in high-performance computing (Figure 5).

Heterogeneous memory systems balance performance and energy efficiency in data-intensive applications by combining **DRAM** and SRAM. Neural Processing Units (NPUs) are processors designed to improve AI performance. Quantum processors, still in development, could outperform traditional computers in calculation speed [**Fu et al., 2016**]. A "AI chip" may include GPUs, FPGAs, and AI-specific

ASICs. CPUs can handle basic AI tasks, but their effectiveness is decreasing as AI evolves. Thus, AI chips have a high volume of parallel calculations, mixed precision , accelerated memory access, and programming languages that convert AI code for AI chips.

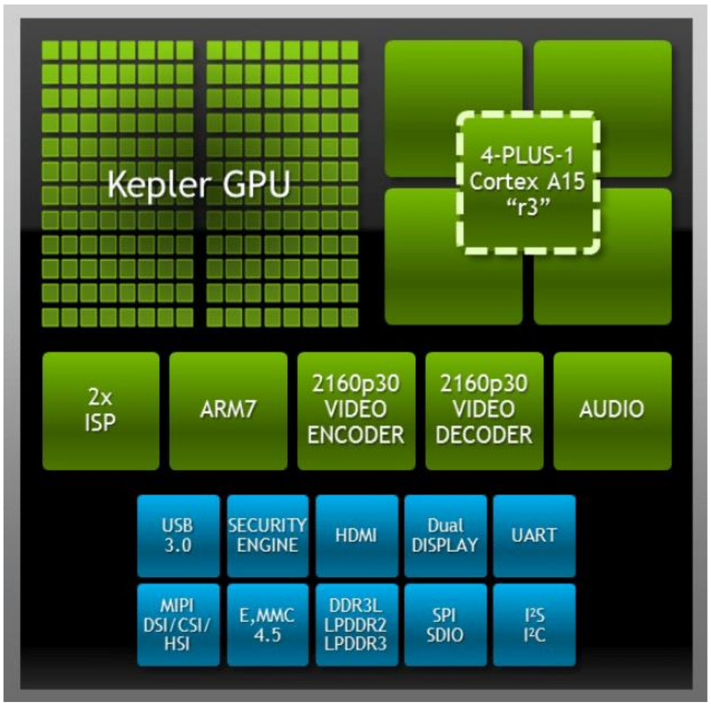


Figure 3. NVIDIA Tegra K1 chip integrating processors with GPUs in an SoC

Selecting the AI chip that will implement a given AI algorithm is a major task. Graphics processors are used during training. Inference is efficient with FPGAs. In contrast, ASICs can conduct both steps.

The 2020 SuperComputing conference announced Cerebras [Rocki et al., 2020] and SambaNova [2021] AI chips. WSE-2, the second version of Wafer Scale Engine, powers Cerebras chips' deep learning and sparse tensor operations. It has 2.6 trillion transistors, 850,000 AI-optimized cores, and 40 gigabytes of high-performance on-wafer memory. Figure 4 [Cerebras, 2021] compares the Cerebras WSE-2 chip to the biggest GPU.

The Reconfigurable Dataflow Architecture (RDA) on the SambaNova chip integrates machine learning and high-performance computing [SambaNova, 2021]. Dataflow procedures can be executed on the RDA for various computing issues. It also has no fixed Instruction Set Architecture (ISA) and is customized for each instance.

Intel (Ponte Vecchio) and AMD (APUs) sell systems with CPUs and GPUs on a chip [**Dávila et al., 2019**]. Another option is the System on Chip (**Figure 3**), which has processors, accelerator, memory, and I/O. These chips are used in sensors, IoT, and edge computing.

In addition to on-chip heterogeneity, the CPU can operate with the GPU or FPGA (Intel A10) on the board.

In this increasingly diversified world of accelerators, the programming paradigm must adjust from catering to one type of accelerator to a multi-acceleration environment [**Vetter et al., 2022].** With the advent of AMD and Intel GPUs, new parallel programming interfaces are needed instead of NVIDIA's exclusive CUDA language.

In conclusion, AI chips with specialized hardware components may change AI algorithm execution in the coming years. Thus, integrating these chips into high-performance computing systems and developing new artificial intelligence frameworks may be problematic.

3.4 AI Revolution

In the forthcoming years, AI is projected to assume a pivotal role in several domains, particularly in national and international security . Nevertheless, general-purpose AI software, databases, and algorithms are ineffective when operated on conventional HPC systems; hence, attention has transitioned to computer hardware specifically designed to perform contemporary AI applications.This hardware is referred to as a "AI chip," which may encompass accelerators such as **GPUs**, **FPGAs**, and **ASIC**s (application- specific integrated circuits) that are optimized for artificial intelligence. Although computers with general-purpose processors (e.g., CPUs) may do basic AI tasks, their use is diminishing as AI progresses. Consequently, AI chips possess optimized design attributes that enhance the computations necessary for AI algorithms, such as a substantial capacity for parallel calculations, the implementation of mixed precision , expedited memory access, and the provision of programming languages that effectively interpret AI code over implementation on AI chips.

Challenges in High-Performance Computing

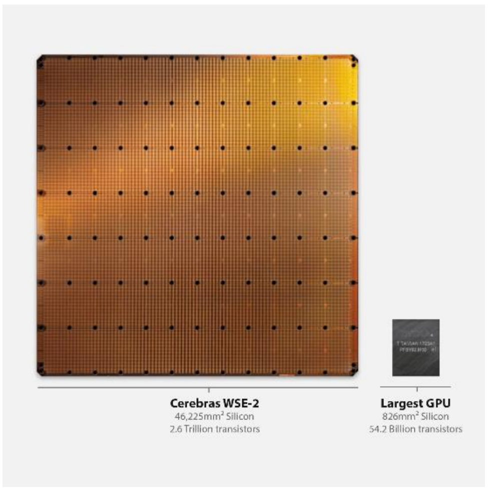


Figure 4. Comparison of Cerebras WSE-2 with the largest GPU at the moment.

In this context, a primary problem is the accurate selection of the AI chip that will implement a certain AI algorithm. Graphics Processing Units are frequently utilized during the training phase. FPGAs proficiently execute the inference process. Conversely, ASICs are capable of performing both stages.

During the 2020 SuperComputing conference, two novel AI chips were unveiled: **Cerebras** [Rocki et al., 2020] and **SambaNova** [2021]. Cerebras chips utilize the second generation of **Wafer Scale Engine**, WSE-2, a central processor designed for deep learning and sparse tensor operations, with 2.6 trillion transistors, **850,000** AI-optimized cores, and **40 gigabytes** of high-performance on-wafer memory.

The SambaNova microprocessor has the **Versatile Dataflow Architecture** (RDA) , enabling the integration of machine learning and high-performance computing (HPC).

In summary, dedicated hardware devices for AI chips will gain popularity in the next years, perhaps transforming the execution of AI algorithms. Consequently, issues may arise in integrating such chips into conventional HPC systems and the development of novel AI frameworks. The RDA offers a flexible execution mechanism for various dataflow computation challenges. Moreover, it lacks a set Instruction Set Architecture (ISA) and is programmed individually for each instance.

3.5 Processing in Memory

As per **Lee et al., (2021**), Processing In Memory (PIM) refers to the execution of instructions within memory, eliminating the conventional need to transfer data to the processor . This technology eliminates delay in transmission of information, which is its primary advantage. PIM has been expanding in research, and some components are emerging in the market. Nonetheless, while it enables data-intensive programs to circumvent the transfer of data from memory to CPU, it presents new issues for high-performance computing framework designers and software designers.

Despite the resolution of several obstacles in developing PIM architectures in recent years, certain issues remain unresolved [**Ghose et al., 2019**]. One aspect is to how HPC programmers might leverage the advantages of PIM without resorting to intricate programming methods. A further problem is comprehending the limitations of various substrates in the design of PIM logic. Consequently, the creation of a PIM programming model, data mapping, and runtime scheduling for PIM are complex issues that must be resolved prior to the majority of HPC developers use PIM.

High-tech firms, such as Samsung, Micron, and Synopsys, commenced the development of memory systems with computational capabilities. Ultimately, they contend that memories and AI computation will integrate into a singular architecture, resulting in AI-driven memory chips.

3.6 Quantum Computing

As quantum processors materialize, it is conceivable to consider heterogeneous systems incorporating quantum processing units. The potential development of atom-sized operators utilizing germanium transistor technology [**Hendrickx et al., 2021**] is facilitating the emergence of qubit processors in the market. Consequently, next designs will have x86 processor units with qubit units (Figure 5).

Initially, quantum computers will function as accelerators. They will address issues, particularly in security, cryptography, meteorology, medicines, biotechnology, and economic models.

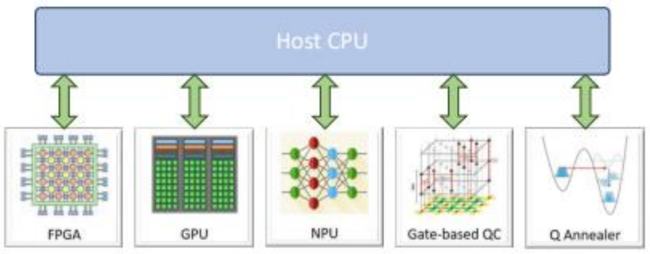


Figure 5. Future of Heterogeneous Architectures with Quantum Processors - Based on Fu *et al.* [2016]

In the coming years, quantum computing will not replace traditional computing [**Matsuoka et al., 2023**]. Quantum computing will facilitate the resolution of intricate issues that require substantial time when addressed by traditional computers. Instances of issues that will gain from it are evident in modeling. In recent years, more quantum computers have been unveiled, including the 54-qubit Sycamore processor and IBM's 14th quantum computer model using 53 qubits. Nevertheless, as scientists anticipate real computing, machines require about one thousand qubits. In this context, IBM aims to construct a quantum computer including 1000 qubits by the year 2025.

Ultimately, a significant issue of quantum machines is rectifying the many mistakes often associated with quantum processes. The researchers concentrate on minimizing the overall error rate of the system. Quantum computing will represent the next frontier in processing capability, enabling the timely resolution of currently intractable scientific issues. **Zuchongzhi with 66 qubits**, **Google with protein or molecular simulations**, the development of new strong encryption, and data processing from accelerators such **as CERN**, and other intricate challenges.

3.7 Cloud Computing

The major cloud computing providers began to express interest in offering these capabilities in response to the growing demand for HPC in various sectors, as indicated in previous Sections Users were able to generate a set of machines that could satisfy a demand for more processing power by observing the formation of higher-powered GPUs, FPGAs, and enhanced interconnection systems within the cloud.

Following **Paillard et al., (2015)** HPC as a Service (HPCaaS) is being implemented in this situation**. Figure 6** illustrates an example of infrastructure for executing parallel workloads on the Cloud. In this setup, end-users submit their jobs via the internet, and the environment of cloud computing **(HPCaaS)** plays the role for channeling machines and running the assigned task for execution on HPC systems.

The advertisements on the websites of the prominent cloud providers, 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," and "Google Cloud - HPC in the cloud becomes a reality," clearly demonstrate the importance and interest that these companies are placing in offering HPC in the cloud, an essential and expanding component of HPC.

The initial cloud performance issues are progressively being resolved, enabling the execution of HPC applications in this environment to yield satisfactory results. The cloud can now implement complex applications with sufficient processing speed due to the increased use of quicker and better connections, the accessibility of new gen processors, and improved storage management. To operate high-performance computing (HPC) applications, cloud servers—which may provide resources on demand over the Internet—need an affordable and efficient infrastructure.

Additionally, we must address the issue of serverless computing, a novel approach to cloud application development. Baldini et al. (2017) refer to this as a new level of cloud computing, and Function as a Service (FaaS), which simplifies the user experience.

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 Design Patterns for Parallel Algorithms

Software developers can utilize many methods of communication when scaling applications to facilitate association amongst concurrently executing cores, including memory sharing and message-passing. The former relies on the presence of a memory address space accessible by all CPUs. It is commonly utilized when parallelism is leveraged at the thread level, as they share the identical memory address space. Conversely, message-passing is utilized in contexts where memory is distributed and/or processes do not share a common memory address space. The objective is to employ the programming approach that optimally leverages the target architecture. Shared memory generates superior results for multiple cores and many-core processors, but message-passing is optimal for huge systems that interact over an interconnection link. Besides the communication mechanism, another challenge is selecting the parallel programming paradigm to derive parallelism from sequential code. For several years, the fork-join paradigm has been the predominant approach utilized by software developers due to its simplicity in harnessing parallelism. In the fork-join approach seen in **Figure 7**, the master thread (illustrated by the orange rectangle) initiates the execution of the sequential phase. Fork procedure creates threads team to perform the parallel area concurrently. Subsequently, at the conclusion of the region, all threads execute a join operation to achieve synchronization. From this point forward, only the master thread processes the application binary until it encounters another parallel zone. Given that this model is extensively utilized for parallelizing applications on multicore systems, and that processor manufacturers (e.g., AMD, ARM, Intel, and NVIDIA) are increasingly producing processors with additional cores, the challenge lies in developing algorithms that optimize performance on these architectures. In short, the primary objective is to increase the number of threads while maintaining performance and energy efficiency.

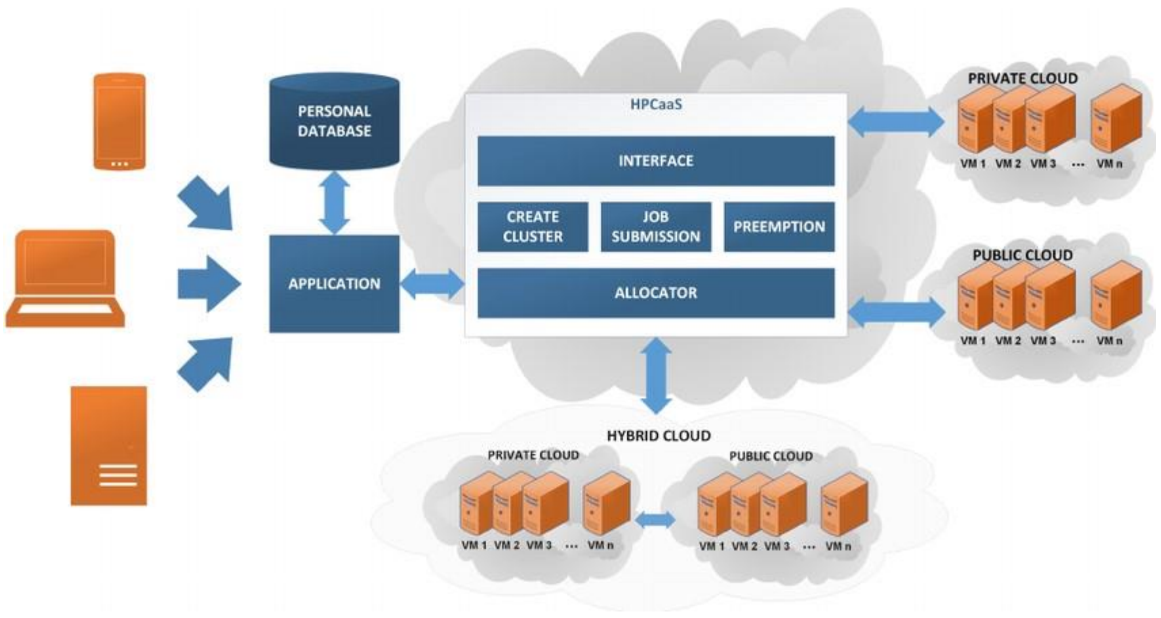


Figure 6. HPC as a Service on Cloud Computing - Paillard et al. [2015]

Nonetheless, the inflexible execution model and absence of adaptability in parallel applications designed with the fork-join paradigm may inadequately address certain hardware and software factors (e.g., data synchronization and cache contention) in the presence of variability in application behavior or execution environment, hindering linear performance enhancements. In such scenarios, inflexible fork-join implementations might elevate power consumption and compromise the performance of parallel applications. This indicates a trend towards the popularization of task-based programming paradigms, as elaborated in the subsequent section (Section 5), which can mitigate **fork-join constraints**, offering enhanced flexibility and improved load balancing on multicore systems. Nonetheless, the challenge encountered by software developers in using task-based level parallelism is in delineating the data dependencies across various parallel zones, which may constrain the benefits of parallelization. Irrespective of the programming paradigm, software developers can utilize design patterns, such as Map, Stencil, and Reduction, to facilitate the exploitation of parallelism, which are now the most prevalent. The Map design, seen in **Figure 8**, partitions the workload (e.g., array, list, or other related collections) into separate segments that can execute concurrently without data dependencies, exemplifying a form of parallelization known as embarrassed parallelism. **Voss et al. (2019).** A function is applied on all members of a collection, often resulting in a new collection that retains the same structure as the input.

Furthermore, the quantity of repetitions and the inputs may be same and predetermined. The Map pattern presents a difficulty for modern computer architectures due to the requirement for a substantial function to be executed on each member of a collection and the necessity for several iterations to engage the processor's cores effectively.

According to **Voss et al., (2019),** The Map's Stencil pattern combines and applies functions in a collection of acquaintances from every matrix's . After computing its neighbors' values, **Fig. 9** shows a stencil operation on one matrix element. The collaboration incurred between threads/processes makes boundary conditions difficult for software engineers. Since GPUs can run hundreds of threads simultaneously, the Stencil design scales well. Finally, the primary difficulty will be to balance memory operations and calculation time such that neighbors are loaded quicker than computed. Many applications include many threads/procedures doing the same task on different data. Here, the Reduction pattern uses the combined function to combine each thread/process's computed elements. **Figure 10** illustrates a parallel reduction method that calculates pairs of items till the final result. Although it performs well when combining values calculated by each thread, its implementation becomes more complicated as the number of reductions after each iteration changes until the final result is obtained. To optimize hardware resources, software writers must coordinate computing throughout each reduction process.

4.2 Parallel Programming Tools

Many languages for programming and packages that let software developers use parallels were published in the literature as new architectures displaying different computing capabilities emerged. We examine each one following and show in **Figure 11** the change of the amount of citations on the Scopus database of the most utilized parallel programming languages over the years.

As shown, one of the key substitutes for using parallelism throughout the years is **Message-Passing Interface**, MPI. Every HPC system requires a library to develop processes and control communication between them in distributed memory settings, so its popularity results from this necessity. Following **Gabriel et al., (2004)**, The difficulties in the technology accessible in HPC systems are related to (i) balance communication and analysis across the nodes in order to better use hardware resources; (ii) effectively use asynchronous communication to overlap computation; and (iii) ensure fault tolerance mechanisms. **MPI-1, MPI-2, and MPI-3**

Challenges in High-Performance Computing

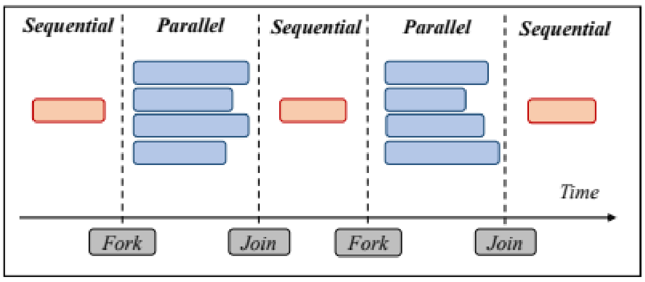


Figure 7. The fork-join shared-memory programming model.

Concurrent with the rise in core counts in multicore architectures, libraries using parallelism in shared memory became very popular (for example, OpenMP and POSIX Threads). First to produce parallel programs was the lightweight thread implementation, **POSIX** Threads library. According to **Barney, (2009),** The complexity and demand programmers to decide on several operating system components. It is still used only for creating operating system-target development applications.

Under this situation the OpenMP library took the role of the POSIX Threads library. OpenMP comprises environment variables, li-brary tools, and compiler instructions intended to simplify the bur-den of thread management in the code as per Chandra **et al., (2001)**. Consequently, when extracting parallelism using OpenMP, compared to POSIX Threads, the work is often less. By including instructions in the code guiding the compiler on whether and how to run parallel elements of the program, par-allelism is taken advantage of. Given the complexity of the memory hierarchy and the growing number of cores in shared-memory architectures, the challenge of utilizing **OpenMP** will be to **(i)** determine the optimal number of threads to execute each parallel region, **(ii)** establish thread and data placement strategies that minimize cache contention and last-level cache misses, and (iii) effectively implement directives for heterogeneous computing. Furthermore, other libraries, including **Colk Plus** [Schardl et al., 2018], created by **MIT** and subsequently maintained by Intel, appeared to enable vector and task programming but owing to the development of the OpenMP library they also stopped to be utilized. Providing higher malleability and better load-balancing on multi-core systems, **OmpSs-2** has become a substitute for OpenMP in the past years when utilizing parallelism via tasks.

Challenges in High-Performance Computing

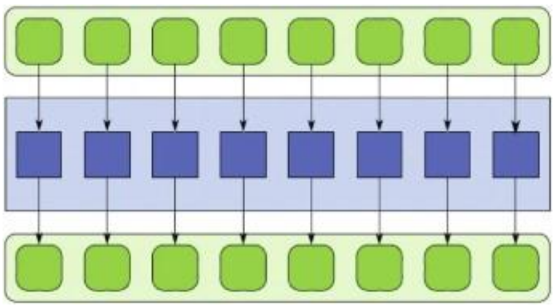


Figure 8. Map pattern example, where a function is applied to all elements of a collection, producing a new collection with the same shape

The situation dominated by OpenMP and MPI drastically shifts with the acceptance of GPU architectures. From then on, **CUDA** became among the most often used paral-lel programming interfaces to speed multi-domain applications on NVIDIA GPUs **[Sanders and Kandrot, 2010].**The CUDA library comprises of a collection of extensions for C, C++ and **FORTRAN** that let the programmer create functions running on NVIDIA GPU-type graphics devices **[Cook, 2012].** Apart from this library, OpenCL also became a framework for heterogeneous computing, enabling the software developer to implement programs running on graphics cards from any manufacturer on both multicore CPUs **[Munshi et al., 2011]**. Ultimately, the **OpenACC** library has gained popularity as, when using parallelism, it resembles the OpenMP library [**Farber, 2016**]. It also makes simple programming for GPUs by using pragmas and quick speed.

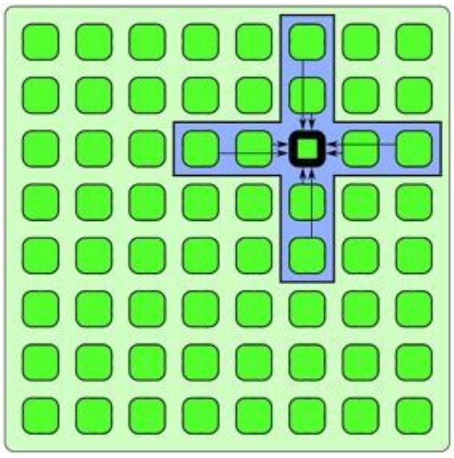


Figure 9. Stencil Computation Example [Voss et al., 2019]

Intel developed **OneAPI** in the end of 2018 in view of end customers expected rising usage of heterogenous designs. By offering the same programming languages and models over various accelerator designs, it streamlines software development. OneAPI aims to provide developers of software stack portability, performanace transparency, and source-level compatibility.   Under this situation, the difficulty will be determining the optimal architecture to run every piece of parallel code without compromising the whole performance of the HPC system.

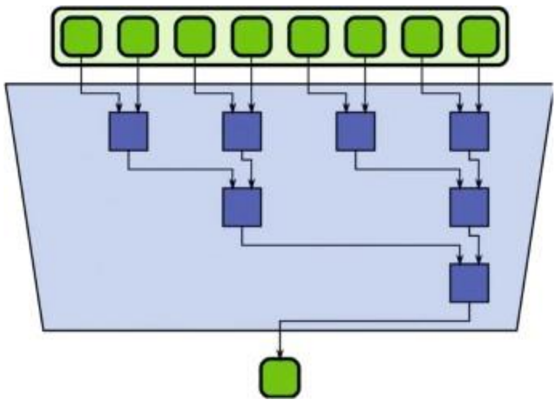


Figure 10. Parallel reduction, where threads produce subresults that are combined to produce a final single answer [Voss et al., 2019]

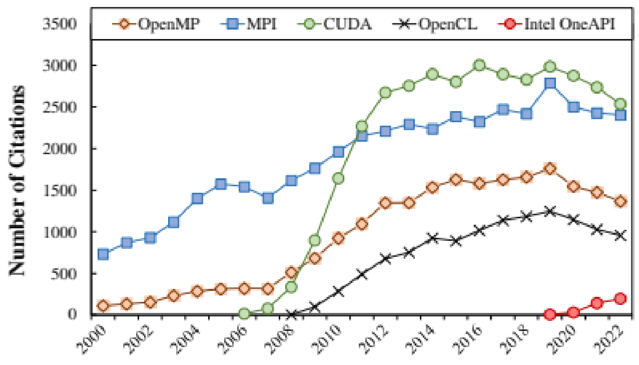


Figure 11. Number of citations from the foremost parallel programming libraries over the years in the Scopus database.

More than one parallel programming interface is used in current and effective parallel applications. While the OpenMP and CUDA / Ope- nACC libraries make use of numerous processor cores and GPU-type graphics devices, the MPI library is usually used for message exchange between several computing nodes. Compiler emergence and development in both self-parallelization and data placement optimization follows a pattern.

5. Additional Challenges

Other challenges become significant to be solved given the rising homogeneity of IT assets and libraries and frameworks needed to leverage parallelism in HPC servers outlined in the final sections. Therefore, it addresses in this part the trends in energy usage, that is, what one should pay attention to in HPC systems to raise their energy efficiency. As covered in Section 5.2, future HPC systems will also need increased resilience to lower the quantity of hardware and software failures. In Section 5.3 and 5.4 respectively, we also cover two substitutes to maximize the energy economy of HPC servers: mixed-precision and data locality.

5.1 Energy Requirement

Growing need for processing capability of HPC computers has driven manufacturers to link hundreds of processors in clusters, hence creating excessive power consumption. Under this situation, some TOP500 list computers utilize almost 30 MW, which is equivalent to a place with about **300,000 people**. As such, generators of processors and factories strive to maximize designs so that consumption lowers. Among other strategies, these optimizations include changing the CPU design, controlling non-active components, and lowering the running **frequency [Padoin et al., 2019**]. New computers nowadays are designed to boost energy consumption and instruction speed, which sometimes runs the other way.

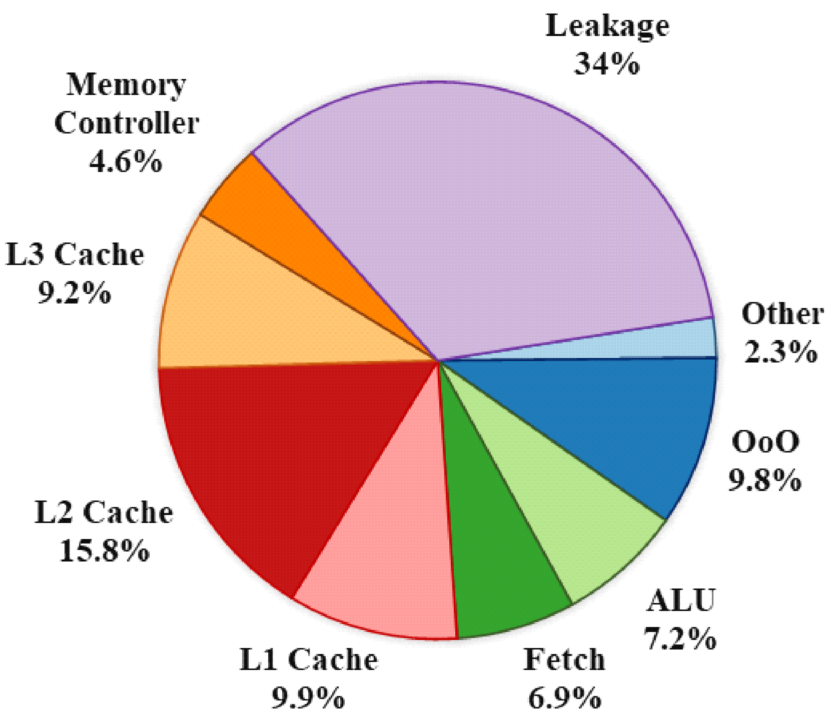


Figure 12. Distribution of energy consumption in a core.

As per **LeBeane et al., (2015)** Extracted from a recent multicore processor using Watt Watcher program while running well-known parallel tasks , Figure 12 completely demonstrates one of the difficulties for the development of CPU designs. It is confirmed that, out of all the energy, around 17% is devoted to the effective analysis (OoO and ALU). Comparatively, roughly 34% is spent on static energy, leakage, circuits. Moreover, 11% on registers and around 35% on accesses to different levels of cache memory access. In the end, even if these per-centages may increase throughout the next few years, it seems that the energy invested in the ultimate aim, which is the execution of the instruction, is modest compared to the amount spent in other areas of the processor's functioning. Consequently, this is a major obstacle that has to be overcome in next CPU designs.

5.2 Resilience

Resilience addresses a system's capacity to keep running in the midst of performance variances or breakdowns. In supercomputers—which contain hundreds of cores, memory, and circuits linking them—which handle high-performance apps—the likelihood of a breakdown in any one of their elements becomes more noticeable. Thus, one of the difficulties to be overcome is resilience, which will help to maintain cheap infrastructure-related expenses and continuous high-performance execution of scientific applications.

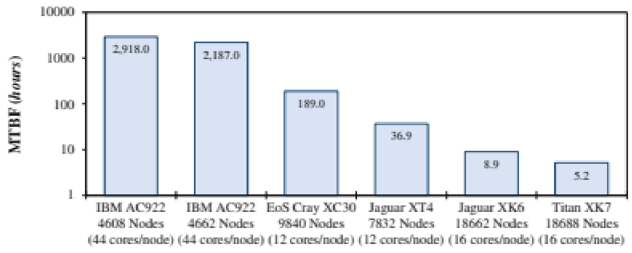


Figure 13. Mean time between failures on different HPC servers.

Three main factors will lead to much less dependability of newer HPC systems: the decreasing device dependability and the shrink-ing process technology; the increasing complexity of the system de-sign and number of hardware resources (e.g., cores and mem- ories). Extracted from [**Gupta et al., 2017**], Figure 13 displays the mean time be- in between failures (MTBF) of several HPC systems with varying number of nodes and CPUs per node. The mean time (in hours) between failures is clearly shorter in the HPC system the more nodes it has. Resilience is thus highly important to keep the system going as newer supercomputers with hundreds of CPUs and nodes will have servers breaking daily.

With more and more cores reaching thousands of them, analysis throughout time reveals the increase of faults with the development of supercomputers. **Figure 13** depicts this progression; consequently, a supercomputer with thousands of CPUs would have daily server failures, hence the need of resilience to keep the system operating. Hardware and software that will dynamically identify the issue, diagnose, reconfigure, and fix it will help to build resilience thereby enabling processing to go on without user knowledge. Thus, this region becomes important in next machines to satisfy the demands of high-performance processing and will enable this processing to run until producing results without interruption.

5.3 HPC and Mixed Precision

Researches optimize operations taking into account the right demand of accuracy in each calculus due to the total amount of time consumed with high precision. Executing applications now requires a new level of precision reduction.

Improved storage, energy, and computing needs are possible with mixed-precision designs, which typically enable multiple floating-point precision algorithms. As per **Freytag et al., (2022**), By limiting the accuracy of certain data and arithmetic operations of the problems, it is feasible to mitigate the preliminary result by improving the speed and energy efficiency .

Other than the accuracy decrease, there is Estimated Arithmetic that works with reduced arithmetic's units, that are less costly on area and thus on energy, whose outcomes are less accurate.

5.4 Data Interpretation

The application's performance and energy consumption will be greatly impacted by the amount of energy spent on data movement alone on future microprocessors with more complicated memory hierarchies. Therefore, the amount of energy that may be used for processing will decrease for every nano-joule that is used to transfer the data. Following **Cruz et al., (2021)** , In this case, task mapping and programming need to be optimized in the interconnection network, prioritizing location to reduce data transfer as much as feasible . Thus, the ten-dency is to put data locality ahead of processor performance, even if processing times are often quicker when using local data.

Challenges in High-Performance Computing

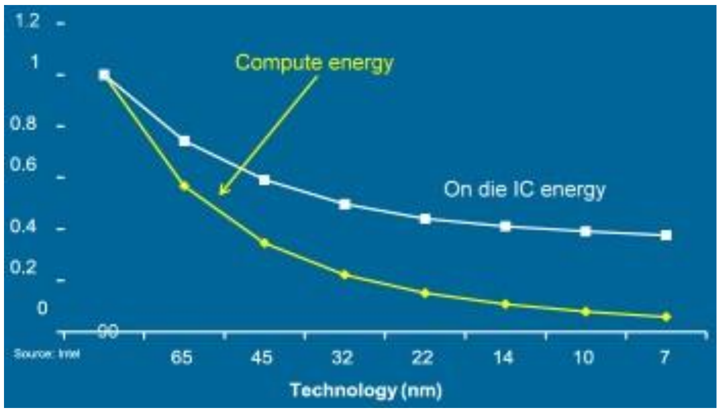


Figure 14. Energy consumption at Pico Joules versus technology

As per the report of **DoE by Vetter et al., (2022),** demonstrates that, even though chip technology has progressed to a point where it can be made thinner—with technologies reaching **7 nm**—the overall reduction in energy consumption has not kept up with the reduction in computing power. A major paradigm shift is required, both in processor design and instruction execution, since this trend demonstrates that data transfer consumes more power than computational operations on the device. Compilers should be concerned with bringing data closer to the processors.

Conclusion: High-Performance Processing has transitioned from a particular field. that served specific processing demands to a key component in the evolution of computing, in light of the increasing processing resource requirements in science, through fields like Big Data, Artificial Intelligence, Data Science, etc. as alluded in the text above. This evolution passes through major trans- formations of machines and processors, such as the rising cloud and cloud usage to accomplish this need of the computational power. So energy consumption is often the real optimization goal, not only in the quest for performance. Heterogeneity is a fundamental characteristic of processors and machines, and the introduction of quantum processors will add another level of diversity. Resilience thus is crucial to new HPC systems as failures, er- rors, or interruptions at the system level can lead to loss of data, delays, or system down- time and incur significant financial, productivity, or even safety costs in crucial applications. Moreover, new paradigms in programming and storage are a key component of HPC evolution. In conclusion, the advancement of computing must be the constant development of processing power and better methods for it.

Reference

- Aurora (2021). Argonne leadership computing facility. Available at:https://www.alcf.anl.gov/aurora. Ac- cessed: 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: Cur- rent trends and open problems. Research advances in cloud computing, pages 1–20. DOI: 10.1007/978-981-10-5026-81 .
- Barney,B.(2009).POSIXthreadsprogramming.NationalLaboratoryAvailab leat: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 sci- entific visualization. In ACM siggraph, volume 97, pages 21–38. MRJ/NASA AmesResearchCenter.Availableat: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.
- how-much-data-is-generated-each-day. Accessed: Mar. 12, 2021.
- Dongarra, J. H. M. and Strohmaier, E. (2020). Top500 su- percomputer:. 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 Compu- tational Science and Its Applications–ICCSA 2022 Work- shops: Malaga, Spain, July 4–7, 2022, Proceedings, Part IV, pages 570–587. Springer. DOI: 10.1007/978-3-031-10542-53 9.
- Frontier (2021). ORNL Exascale Supercomputer. Available at:https://www.olcf.ornl.gov/frontier/. Accessed: Apr. 10, 2021.
- Fu, X., Riesebos, L., Lao, L., Almudever, C. G., Se- bastiano, F., Versluis, R., Charbon, E., and Bertels, K. (2016). A heterogeneous quantum computer architec- ture. In Proceedings of the ACM International Con- ferenceonComputingFrontiers,pages323–330.DOI:10.1145/2903150.29068

- Fujitsu (2021). Supercomputer fugaku. Available at:https: //www.fujitsu.com/. Accessed: Apr. 10, 2021.
- Gabriel, E., Fagg, G. E., Bosilca, G., Angskun, T., Don- garra, J. J., Squyres, J. M., Sahay, V., Kambadur, P., Bar- rett, B., Lumsdaine, A., et al. (2004). Open mpi: Goals, concept, and design of a next generation mpi implemen- tation. 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 11, pages 97–104. Springer. DOI: 10.1007/978-3-540-30218-61 9.
- 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 Devel- opment, 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 In- ternational Conference for High Performance Comput- ing, 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-Challenges in High-Performance Computing
- pucci, G., and Veldhorst, M. (2021). A four-qubit ger- manium 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.
- Lee, S., Kang, S.-h., Lee, J., Kim, H., Lee, E., Seo, S., Yoon, H., Lee, S., Lim, K., Shin, H., et al. (2021). Hard- ware 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 Tech- nology & Electronic Engineering, 19:1236–1244. DOI: 10.1631/FITEE.1800494.
- LLNL (2021). DOE/NNSA Lab announces a partner- ship with Cray to develop NNSA's first exascale su- percomputer. Jeremy Thomas. Available at:https:// www.llnl.gov/news/. Accessed: Sep. 10, 2021.
- Matsuoka, S., Domke, J., Wahib, M., Drozd, A., and Hoe- fler, 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 im- balance 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.
- Reed, D., Gannon, D., and Dongarra, J. (2022). Rein- venting high performance computing: Challenges and
- Navaux et al., 2023
- opportunities. arXiv preprint arXiv:2203.02544. DOI: 10.48550/arXiv.2203.02544.
- SambaNova (2021). Accelerated computing with a recon- figurable 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 Pro- ceedings 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 intelli- gence (ai) for science. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States).
- Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Ver- belen, T., and Rellermeyer, J. S. (2020). A survey on dis- tributed 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 het- erogeneity: 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 Thread- ing Building Blocks, pages 233–248. DOI: 10.1007/978-1- 4842-4398-58 .
- Xenopoulos, P., Daniel, J., Matheson, M., and Sukumar, S. (2016). Big data analytics on HPC architectures: Perfor- mance and cost. In 2016 IEEE International Conference on Big Data (Big Data), pages 2286–2295. IEEE. DOI: 10.1109/BigData.2016.7840861.