GDAŃSK UNIVERSITY OF TECHNOLOGY

TASK Quarterly 29 (1) 2025

High Performance Computing - a gentle introduction for non-specialists

Tomasz Fruboes*, Arkadiusz Ćwiek, Piotr Prusiński, Piotr Wasiuk

National Centre for Nuclear Research, Andrzeja Sołtana 7/3, 05-400 Otwock, Poland

*tomasz.fruboes@ncbj.gov.pl

https://doi.org/10.34808/tq2025/29.1/d

Abstract

High-performance computing (HPC) is a field of computer science focused on the use of highly efficient computer systems to perform complex calculations in a short time. In the article, we present the basic principles of HPC and explain how systems based on parallel computational architectures allow for the performance of multiple concurrent calculations. HPC technologies are crucial in various areas such as scientific research, industry, engineering, and data analysis. Applications include (but are not limited to) physics simulations, genome analyses, climate change forecasting or artificial intelligence research. The article also discusses the main building blocks of the HPC environment, such as supercomputers or software allowing parallel calculations and analyses of large data volumes. The main HPC challenges, including cost, scalability, and the need for specialist technical knowledge, are also considered. This article aims to present the reader with an overview of what HPC is, how it works, and what its current role is.

Keywords:

High-Performance Computing, Parallel Computing, Supercomputers, Scalability, Scientific Computing

1. Introduction

In the 1922 book "Weather Prediction by Numerical Process" [1], Lewis Fry Richardson was the first to propose (based on earlier work by researchers C. Abbe and V. Bjerknes) and test a numerical weather model. Due to the need for complex calculations (at the time still performed manually using paper, pencil and mechanical calculators), this test was relatively simple - it concerned forecasting pressure changes over Central Europe for a short, 6 hour long period. The resulting forecast significantly differed from the later measured values of the air pressure, which the author correctly attributed to the poor quality of the input data. Nevertheless, the approach proposed by the author is considered a milestone for numerical weather forecasting.

In his book, Richardson discusses the possibility of conducting weather forecasting using his model applied to the entire world. He assumed the use of a large number of "computers" (at that time this term meant a person skilled in mathematics and performing calculations according to a given scheme). Each "computer" was to be responsible for performing calculations (i.e. making a forecast) appropriate for a small patch of land based on previous results for its patch and also for neighboring patches, calculated by other "computers".

The approach described by Richardson is consistent with how we calculate numerical weather forecasts today. The area for which calculations are performed is divided into small patches, for which weather conditions are calculated for subsequent time steps. Calculations for a given patch are based on previous results for itself and for neighboring patches. These calculations are performed by many computers (here already understood in modern terms - as computing machines equipped with a processor, RAM and other components), which is a consequence of the numerical complexity of the calculations, but also of the very number of patches into which the area of the forecast was divided. The smaller the patches are, the more accurate the numerical forecast will be since it allows for a more detailed representation of the impact of different types of terrain, as well as more accurate modeling of processes occurring in the atmosphere itself.

The decision to utilize multiple computers to carry out the calculations necessary to resolve a particular problem may be attributed to a variety of factors. Usually, one of them is the time it takes to perform the calculations - using more than one computer (using more computing power) can shorten this time. In many cases, this is of key importance - in the considered example of calculating weather forecasts, the longer the waiting time, the less usable the forecast is. In the most extreme case, calculations could end so late that the obtained result would concern the past in its entirety. In the case of weather forecasts developed at the Interdisciplinary Center for Mathematical and Computational Modeling of the University of Warsaw (ICM), calculations for the British MetOffice Unified Model (UM) (in which the area for which calculations are performed is divided into 1.5 km patches, 1024 in an east-west direction and 1472 in a north-south direction and the forecast is computed for 78 hours) are completed in 70 minutes, using about 130 computers (so called computing nodes), each having 24 processor cores [2].

Another reason why using more computing nodes may be necessary is the size of the data itself. In many cases, this is forced by the limited capacity of a single computing node, e.g. the size of its RAM (which cannot be expanded beyond a certain maximum size for a single server). Both input data and current calculation results are stored in RAM - the data volume may be too large for a single server. Here again, we can use the example of the UM model mentioned above, where the number of computing nodes used cannot be reduced below 4 due to the size of installed RAM on each node. In this case, the calculation time was as long as 29 hours for a 78-hour forecast length [2].

It is worth mentioning that using multiple computers to solve a single problem (so-called parallel processing), is not always possible. In some cases, the problem cannot be divided into smaller pieces that could be processed at the same time by more than one computing node (or even more than one processor core of the same server). Parallel processing also has its limitations, because dividing the calculations between multiple computers will usually result in a reduction of the calculation time by a factor lower than the number of computers participating in the calculations. This is related to:

- the time necessary to divide the problem between different computing nodes (initialization cost),
- the time spent on communication between nodes; this is because to perform calculations on a given computing node, a result obtained earlier on another node may be necessary - communication between nodes is much more time-consuming than communication between processor cores operating within one computing node (communication cost),
- the time necessary to collect the results obtained on all computing nodes and merge them into the final result (merging cost).

These limitations are among the reasons why different computational algorithms vary in how well they can be distributed across multiple computational nodes — we might say that some algorithms parallelize well, while others do not. Another, even more important reason may be the properties of the algorithm itself, which may make it completely unsuitable for parallelization. One of the in-

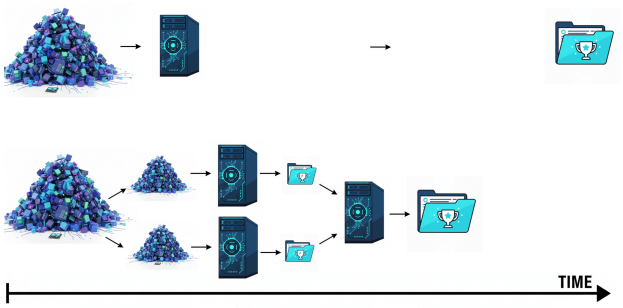

Figure 1: Illustration of the benefits and costs of parallel processing. The use of two computers allows for faster data processing when compared to a single-computer case. The diagram takes into account additional time necessary for input data splitting, extra communications, and merging of partial results in the two-computer case. Despite the additional steps, data processing time is improved.

teresting consequences of the above-described costs is the so-called scalability limit - for a given data set and algorithm, it is not possible to arbitrarily shorten the execution time by adding more computational nodes. Typically, those costs (initialization, communication, and merging) will increase as the number of nodes increases. In extreme cases, the cost increase of adding new nodes may even outweigh the savings, leading to an increase in the time needed to complete the calculations.

Most of the concepts discussed so far are illustrated in Fig. 1. In the above introduction, we have only superficially described the basic principles of High-Performance Computing (HPC), a field of computer technology that focuses on using high-performance computing systems to perform complex calculations in an efficient manner (e.g. by shortening the computation time). In a further part of the article, the chapter "What is HPC used for", we will describe different examples of HPC applications other than numerical weather forecasting. In the chapter "What distinguishes the HPC hardware", we will discuss, among other things, how the need to minimize the above-described costs (initialization, communication and merging) affects the architecture of computing nodes that form the computing cluster. Finally, in the chapter "HPC - is it worth it", we will look into the costs and benefits of HPC usage.

2. What is HPC used for?

High Performance Computing serves as the computational backbone for numerous scientific and industrial fields where traditional computing resources would be insufficient. While we have already discussed numerical weather forecasting, the applications of HPC extend far beyond this single domain.

2.1. Artificial intelligence and machine learning

The rapid advancement of artificial intelligence, particularly large language models (LLMs) like GPT-4, Claude, or Gemini, relies heavily on HPC infrastructure. Training these massive models involves processing enormous datasets using complex neural network architectures that require thousands of high-performance GPUs (specialized processing cards with high performance in AI-related workflows) working in parallel. For instance, training a state-of-the-art language model with hundreds of billions of parameters demands computational resources that would take a single high-end computer thousands of years to complete—yet HPC systems can accomplish this in weeks or months [3].

The reason AI workloads align so well with HPC environments is their inherent parallelizability. Neural network training can be distributed across many computing nodes, with each handling a portion of the dataset or model parameters and regularly synchronizing results. The high-bandwidth, low-latency connections in HPC environments are crucial for reducing the communication overhead during this process.

2.2. Molecular modeling and drug discovery

HPC plays a vital role in pharmaceutical research and drug discovery. Simulating the interactions between potential drug compounds and their biological targets (typically proteins) requires modeling the forces between thousands or millions of atoms. These simulations must track extremely small time steps (femtoseconds) while running for biologically relevant timeframes (microseconds or longer) [4].

Interestingly, the above research field often relies on HPC in another, indirect way. In order to perform the described simulation, one must know in advance the structure of the simulated compounds. In the classical approach, those structures would be determined experimentally, using techniques such as crystallography or nuclear magnetic resonance. Sadly, such approaches may be high-cost, time-consuming, and without a guarantee of success in every considered case. Fortunately, in recent years, new approaches have emerged.

Tools like AlphaFold [5] have revolutionized protein structure prediction by using deep learning techniques that demand substantial computational resources. The ability to predict protein structures from genetic sequences has profound implications across biology and medicine.

2.3. Physical simulations and astronomy

The field of astrophysics has been transformed by HPC capabilities. Simulating phenomena such as black hole mergers, galaxy formation, or supernova explosions requires solving complex differential equations across immense spatial scales and timeframes. The LIGO gravitational wave detections, which confirmed Einstein's predictions about gravitational waves, relied on massive computational simulations to interpret the data and match observed signals to theoretical models [6].

Cosmological simulations, such as the Millennium Simulation or IllustrisTNG, model the evolution of the universe from shortly after the Big Bang to the present day. These simulations track billions of bodies representing dark matter, gas, and stars, requiring usage of thousands of CPU cores (more than eight thousand for IllustrisTNG) in parallel and vast amounts RAM [7–10]. Once again, build-level optimizations of HPC hardware, such as usage of fast and low-latency server-to-server communication cards, are crucial for perfoming calculations.

2.4. Climate modeling

Building on our earlier discussion of weather forecasting, climate modeling represents an even more computationally intensive application. While weather models typically simulate atmospheric conditions for days or weeks, climate models must run simulations spanning decades or centuries. These models incorporate not just atmospheric dynamics but also ocean circulation, ice sheets, land surface processes, and biogeochemical cycles.

The latest generation of Earth System Models uses grid resolutions fine enough to resolve individual cloud systems. This high resolution is essential for accurately representing crucial climate processes like cloud formation and tropical cyclones [11], but results in a necessity of using a supercomputer to perform the calculations.

2.5. Materials science

Developing new materials with specific properties—such as superconductors, more efficient solar cells, or stronger yet lighter alloys—benefits tremendously from HPC. Simulations allow researchers to explore the properties of theoretical materials before attempting to synthesize them in a laboratory. Density functional theory (DFT) calculations, which predict electronic structures from first principles, are particularly computationally intensive but valuable for materials discovery [12, 13].

2.6. Business and industry

Apart from scientific applications, HPC is of great importance in multiple areas of business and industry. In the automotive industry, HPC helps with efficient testing and optimization of designs, reducing the need to manufacture a given part in order to test it physically. It is commonly used in aerospace engineering (Fig. 2) or to optimize the general shape of a new car model to improve fuel efficiency or to reduce its weight by simulating strenght of individual parts to decide where less material can be used. Similar approaches are possible for architecture and civil engineering to predict (and possibly improve) the strength of the yet-to-be-built construction. Finally, we would like to mention finance, where HPC is commonly used to perform risk analysis. Here, efficient low-latency processing of large datasets is crucial and of great benefit to institutions such as banks or insurance companies. Another example where using HPC technologies is beneficial in the area of finance is fraud detection. Here, AI-based approaches are common. The use of HPC technologies reduces the time required to develop new and improve existing AI models, leading to greater consumer satisfaction, for example, by reducing the number of false positives.

The above list of applications if far from complete. For more examples, please refer to the list of HPC success stories available online [14].

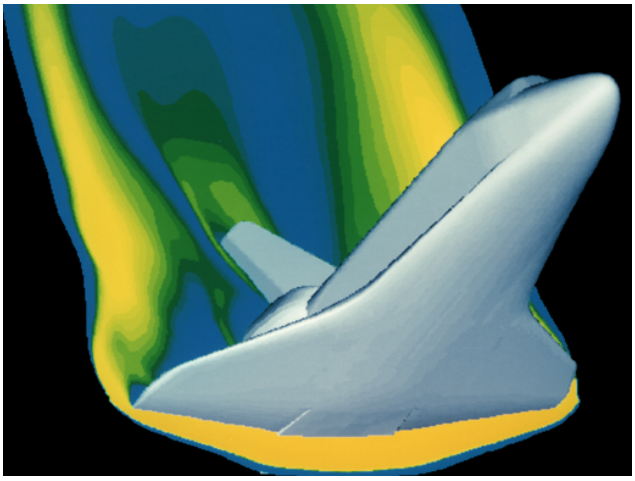

Figure 2: Visualization of CFD simulation of shuttle atmospheric entry.

3. What distinguishes the HPC hardware

In the "Introduction" section, we described the basics of parallel processing - computing a given problem after dividing it into parts, where different parts can be processed simultaneously by different computing units to

shorten the computation time. It is also possible that the computational problem itself may be so large that the resources available on a single computing node will be insufficient to run the computation; therefore, dividing the computational task between many servers becomes a necessity. We also mentioned that such an approach also has some negative effects (related to, for example, the time that must be spent on communication between computing units involved in the computation), which partially absorb gains related to engaging a larger number of computing units.

The architecture of the HPC server (i.e. a machine that, as one of many, can be engaged in solving a given computational problem; also often called a computational node) aims to limit the negative effects of dividing computations between many servers. The most basic principle that HPC system architects follow is to optimize for a high density of computing power. This approach limits its data transmission delays between computing units - these increase with distance, as measurable negative effects are caused by each additional meter of cable/fiber used for data transmission [15]. A high density of computing power is achieved in two ways:

- using the most efficient computing units (e.g. processors or GPU cards),
- using the highest possible density of computing units (e.g. as many CPUs as possible located within the shortest practically achievable distance from each other).

We will illustrate this principle using the example of the fastest Polish supercomputer, Helios, currently ranked 69th in the world (as of March 2025, [16]). A single Helios supercomputer server contains two high-performance (i.e. with high computing power) processors, each of which consists of 96 cores (a processor core can be understood as the smallest unit of computer processor that can independently of the other cores conduct various calculations; at the time of writing this article, a typical laptop contains one processor consisting of several to a dozen or so cores, whose computing performance is significantly inferior to the performance of processor cores used in computing servers). The use of high-performance processors, combined with a large amount of RAM installed, allows for more computation to be performed by a single server, which consequently allows for performing calculations using a smaller number of servers. This in turn limits the amount of data transmissions between servers (simply because there are fewer of them) and as a result reduces the previously described delays (cost) associated with the exchange of data between servers during the calculations. One of the methods used to optimize the size of servers (routinely utilized when designing HPC hardware) is to resign from any component that is not crucial for performing the calculations. For example, the servers that form the Helios supercomputer allow only for the installation of at most one hard drive. Installation of more hard drives would be an unnecessary waste of space in the case of the workloads that HPC servers are expected to run.

A single Helios supercomputer rack (with dimensions of approx. 2.5x1.2x1.7m) can hold a maximum of 256 such servers, which corresponds to 49,152 cores. In total, the entire Helios supercomputer consists of 82,944 high-performance CPU cores and 440 GPU cards with high computing power (it is worth noting the organization of calculations on GPU cards is different than for CPUs - the equivalent of a CPU core in the case of GPUs is simpler and slower, and there are many more such "cores" than in the case of a single CPU - as a result, in some applications, GPU cards achieve higher performance than CPUs). The whole Helios supercomputer consists of 5 racks, of which 2 are responsible just for cooling. As one can easily calculate, the entire computational part has dimensions of only 2.5x3.6x1.7m, which means that the distances between computing units are small. As a result, the amount of time spent on communication is reduced. This time is also limited by using specialized (e.g. different than utilized by home computers) computer network solutions, in which the emphasis is placed on the highest possible data transmission speeds, but - equally importantly - on limiting the so-called latency, i.e. the waiting time for data to be sent. This is crucial, since - as we described earlier - for many algorithms, the processor must suspend calculations until it receives data calculated by a different server.

High computing power requires the use of high electrical power (for the Helios computer, this is more than 450 kW), which results in the production of large amounts of heat in the described (small) volume. This is related to another HPC challenge - the need to use high-performance cooling installations inside the supercomputer, which usually use a liquid as a coolant. This also often results in the need to have a special, additional cooling installation (once again typically liquid-coolant based), which is part of the server room itself, whose task is to receive heat directly from the supercomputer's cooling installation.

Users of the supercomputer usually have very differing needs for computing power - from workloads requiring a single CPU to ones that may utilize nearly all available servers for prolonged periods of time. Therefore, a mechanism that allows for the fair use of available computing resources by multiple users is needed. In this mechanism, typically:

- Rather than running the computation directly (e.g. via an interactive user interface), users first create a job

description in which they define what program to use, its parameters, and input data. Computing resources necessary to perform the calculation are also declared. Such a job description is then submitted by the user to specialized software, a so-called scheduler.

- The scheduler manages allocations of computing resources in order to effectively complete users' jobs. In the case of insufficient computing resources available at any given moment, jobs may get queued (postponed for later execution). In such a situation, the waiting time for the job to start will depend on different factors, including the necessary computing resources of the job (declared in the job description) or how much computing resources were used recently by the owner of this job.

It is worth noting that the job-description approach provides further benefits, also from the users' point of view. Job descriptions are rather easy to create and can be easily re-used or even parametrized. This allows for a nearly effortless computation of multiple different variants of the same computation problem. A practical example of such an approach would be an engineer designing an airplane wing. For a single design of a wing, a vast amount of independent simulations must be carried out with different simulation parameters (e.g. varying wind speeds, directions, or wings' angles of attack). This ensures the required characteristics of the wing in a wide range of conditions.

It is worth stressing that the construction of HPC servers is significantly different from the construction of typical cloud servers (due to different usage goals), even if the basic technologies used may be similar or the same. An example here is the processor itself - cloud servers usually use processors whose base single-core architecture may be nearly identical as in the case of cores of the processor in the HPC server (in the HPC case, the CPU core may contain additional instructions allowing for processing larger chunks of data at once). The most striking difference will be in the number of cores (in the case of a cloud server, it may be much smaller) and other resources necessary to efficiently use the available number of cores - e.g. amount of processor cache, sufficient amount of RAM installed in the server or large-enough CPU-to-memory bandwidth.

Since cloud servers do not require the highest possible density of computing power, they can be larger in size (i.e. have fewer CPU cores available per unit of volume) and contain additional components (e.g. more than one data disk, which may enable, for example, safer data storage on the server). The described differences in the structure of HPC servers and cloud servers result directly from the different requirements set by the software typically run on these servers (e.g. running efficiently on multiple computing nodes). Quite often it is possible to run the algorithms (software) that are typically run on HPC setups on a number of cloud servers, but due to the lack of described build optimizations execution times or the maximal size of data that can be processed with cloud-based setups will usually be much worse. Key differences between HPC and cloud-based computing are summarized in Tab. 1.

4. HPC - is it worth it?

In the "Introduction" section, we discussed one of the examples of HPC applications, which was weather forecasting. As we showed, parallel processing techniques are a key element of these forecasts - enabling the performance of calculations for complex models, i.e. those whose level of detail requires the usage of multiple computational servers, due to limited resources (e.g. RAM size) of a single server. Another key aspect is the reduction of the calculation time - the longer this is, the less up-to-date and usable the forecast is at the moment of its completion.

HPC is used in many different areas. One of them is engineering simulations, which often use algorithms that can be calculated using many servers simultaneously. One such technique is Computational Fluid Dynamics (CFD), which allows for the prediction (using numerical methods) of how the flow of both liquid and gaseous substances will behave in given conditions, e.g. around an obstacle of a given shape. The use of CFD simulations can help to predict the behavior of the object (e.g. aerodynamic drag of a racing car) during the design stage, i.e. without having to physically manufacture it to study its properties. Thanks to this, one can test many variants of an object with ease and determine the best one, which will then be manufactured and tested.

An example of the use of CFD simulations are analyses conducted by one of the recipients of services offered by the Polish National Competence Centre in HPC. The Building Research Institute (the client) is a publicly funded research institute that conducts research in the field of construction and related fields, aimed at their implementation and application in practice. The department we cooperated with deals with fire safety issues. It carries out both scientific research and commercial projects. The commercial activities of the client include expert assessments of fire safety of existing buildings and also those in the design phase.

Such assessments often must take into account the simultaneous impact of the wind and the fire development changing with time. To obtain a statistically valid result, it is necessary to carry out dozens of simulations considering various directions of wind and its speed. This, in turn,

Table 1: Key differences between HPC and cloud-based computing.

	Cloud computing	HPC
Main purpose	scalability, universality	specialized - efficient parallel computation on multiple servers
Communication	standard (ethernet)	additional specialized hardware for a low-latency, high-throughput server-to-server communications
Construction	universal	optimized for high density of computing power
Cooling	primarily air-based cooling	primarily liquid-based cooling
Resource allocation method	via web interface or using specialized APIs	via user-created job descriptions, submitted to the scheduler

requires extensive HPC resources and scalable computational models.

One of these assessments was performed for a multi-storey car park, where one of the challenges was to precisely determine the impact of wind direction. During the study, a 1° wind direction resolution was taken into account, resulting in 360 different wind directions considered. To make the assessment valuable, for a single wind direction multiple CFD simulations had to be carried out, examining different wind speeds and multiple different variants of parked cars configuration. This led to a very large number of different cases that had to be independently performed. Visualization of simulation results for a single set of simulation parameters are shown in Fig. 3.

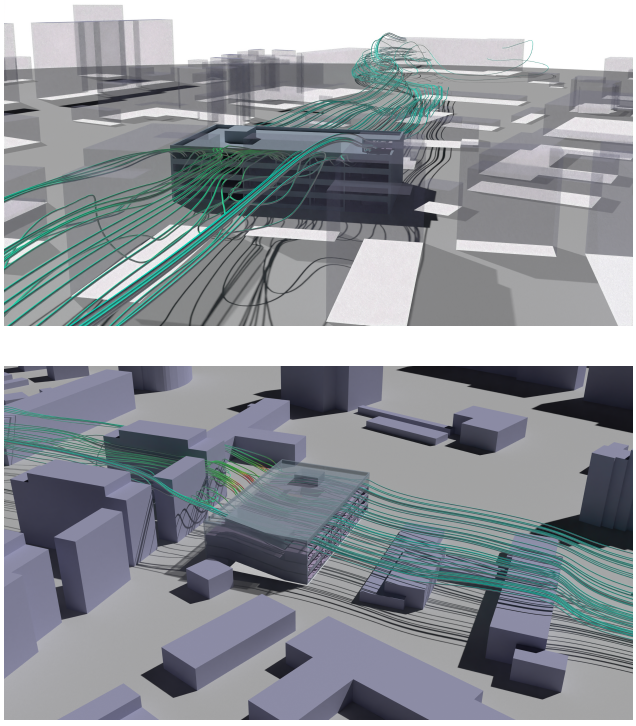

Figure 3: Visualisation of the scope of wind path lines ventilating a car park with one of the vehicles being on fire. Results correspond to a single set of simulation parameters (e.g. single specific wind direction).

Initially, the client performed simulations using his own infrastructure - a single simulation took approximately 6 hours using 128 computer cores. During tests using an HPC cluster built and operated by the National Center for Nuclear Research (NCBJ), we performed so-called scalability benchmarks, where exactly the same CFD simulation (i.e. using the same case setup and the same software, utilizing built-in HPC support) was executed multiple times with a differing number of CPU cores (using multiple servers) used. Results are shown in Fig. 4.

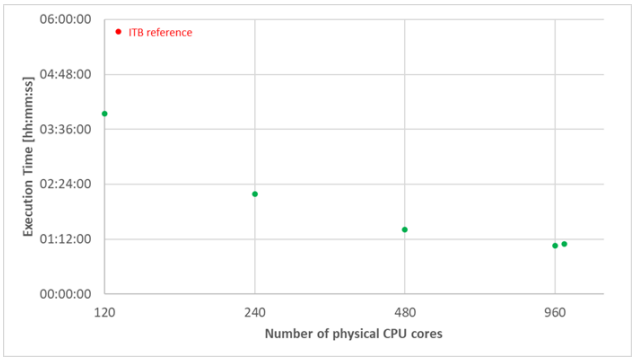

Figure 4: Execution time of CFD simulation as a function of the number of cores used. Visible asymptotic behavior is the consequence of inherent limits of parallel processing (e.g. due to communication costs).

Interestingly, running on a similar number of cores (128) with respect to the client's infrastructure (120) already leads to a decrease of the simulation time of nearly 2 hours (from 5 hours 44 minutes to 3 hours 54 minutes), which shows the benefits of using an HPC-optimized setup. Further reduction of simulation time is possible by increasing the number of cores (servers) involved in performing the simulation. It is worth noting that simulation execution time does not change linearly with respect to the number of cores used - doubling the number of cores from 120 to 240 reduces execution time by a factor of 1.8. Yield from another doubling of the number of cores (i.e. going from 240 to 480) is smaller, simulation time is further reduced only by a factor of 1.54. This behavior illustrates a typical scalability curve, the shape of which is an effect of the parallelization costs

mentioned in the Introduction - e.g. the communication overhead or necessity to merge partial results.

Finally, we would like to point out that at some point adding more resources makes little to no sense - adding an additional server to the execution (i.e. increasing the number of cores from 960 to 1000) does not further reduce the execution time (in this case, it was in fact harmful).

Of course, the use of HPC infrastructure usually involves costs that depend on the access model adopted. An entity that would like to use HPC can build HPC installations on its own or rent computing time on an existing HPC infrastructure. In the case of building its own computing infrastructure, the following costs should be taken into account:

- costs of purchasing and installing equipment - e.g. the cost of servers or necessary network equipment
- operating costs - costs of electricity or replacing damaged equipment. In some cases, also costs of colocation (in the case when the entity does not have its own server room)
- administration costs - costs of personnel who will be responsible for the correct operation of the HPC infrastructure, both in terms of configuration and ongoing repairs

This model may be a necessity for some entities due to the nature of their business - e.g. processing sensitive data that should not leave the company's headquarters. In such cases, the entity may consider outsourcing the administration of the HPC infrastructure to another, external entity. An external entity with HPC competences can be helpful already at the infrastructure design stage, allowing for optimization in terms of both cost and performance, as well as for avoiding potential errors at the design stage.

In cases where there is no need to locate the infrastructure on the entity's premises, the second of the aforementioned access models may be worth considering, i.e. renting computing time on the existing infrastructure. This access method has several advantages:

- no need to incur the costs of building your own HPC infrastructure
- no maintenance costs, including the need to employ staff with the competencies necessary to manage the HPC infrastructure
- adjustment of the computing power with respect to current needs - no costs during periods without using HPC, access to computing power in exactly the amount needed

Such access is possible, for example, on one of the supercomputers built and managed by the institutions that form the Polish National Competence Centre in HPC.

For the sake of completeness, it is also worth mentioning a "hybrid" approach, in which the entity independently builds the computing infrastructure based on machines rented in the cloud model. However, such an approach often does not guarantee the highest possible performance, e.g. due to limitations in the available networking technology between computing nodes, which may negatively affect the time of calculation execution.

5. Summary

In this article, we presented the basic concepts and principles related to parallel processing and High-Performance Computing - for example, how it enables highly complex calculations to be performed efficiently by simultaneous processing using multiple computing servers optimized for high performance. We also showed the limitations of this approach, as well as how the way of building the HPC infrastructure serves to minimize their impact on the speed of calculations. We provided several examples of how HPC is used and also discussed these. By necessity being brief and compact, this article should be treated only as an introduction to the vast knowledge field of HPC and its uses. Different examples of applications can be found in other articles prepared by the Polish National Competence Centre in HPC members in this TASK Quarterly issue.

The Polish National Competence Centre in HPC is an organization whose goal is to popularize knowledge about HPC and facilitate its implementation by public, academic and business institutions. We achieve these goals, among others, through free training or free tests of the computing infrastructure. We encourage you to contact us.

Acknowledgements

This work has been supported by the EuroCC2 project that has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and from Croatia, Bulgaria, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Ireland, Italy, Lithuania, Netherlands, Norway, Poland, Portugal, Romania, Spain, Sweden, and Turkey.

References

[1] L. F. Richardson, *Weather Prediction by Numerical Process.* Cambridge University Press, 1922.

[2] J. Jędraszczyk, "Direct communication on um model calculation times," 2025. Personal communication.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[4] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[5] J. M. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, pp. 583 – 589, 2021.

[6] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, *et al.*, "Observation of gravitational waves from a binary black hole merger," *Physical Review Letters*, vol. 116, no. 6, p. 061102, 2016.

[7] V. Springel, S. D. M. White, A. Jenkins, C. S. Frenk, N. Yoshida, L. Gao, J. Navarro, R. Thacker, D. Croton, J. Helly, *et al.*, "Simulations of the formation, evolution and clustering of galaxies and quasars," *Nature*, vol. 435, no. 7042, pp. 629–636, 2005.

[8] S. J. Hilbert, *Ray-Tracing through the Millennium Simulation*. PhD thesis, Ludwig-Maximilians-Universität München, 2008.

[9] V. Springel, A. Pillepich, D. Nelson, S. Genel, M. Vogelsberger, R. Pakmor, P. Torrey, F. Marinacci, L. Hernquist, J. Naiman, *et al.*, "First results from the illustristng simulations: matter and galaxy clustering," *Monthly Notices of the Royal Astronomical Society*, vol. 475, no. 1, pp. 676–698, 2018.

[10] D. Nelson, A. Pillepich, V. Springel, R. Pakmor, R. Weinberger, S. Genel, P. Torrey, M. Vogelsberger, F. Marinacci, and L. Hernquist, "The first results from the tng50 simulation: The evolution of galaxies and the cosmic web," *Monthly Notices of the Royal Astronomical Society*, vol. 490, no. 3, pp. 3234–3261, 2019.

[11] F. Hourdin, T. Mauritsen, A. Gettelman, J.-C. Golaz, V. Balaji, L. Duan, J.-L. Dufresne, J. Dunne, S. M. Griffies, J. E. Kay, *et al.*, "The art and science of climate model tuning," *Bulletin of the American Meteorological Society*, vol. 98, no. 3, pp. 589–602, 2017.

[12] W. Kohn, A. D. Becke, and R. G. Parr, "Density functional theory of electronic structure," *The Journal of Physical Chemistry*, vol. 100, no. 31, pp. 12974–12980, 1996.

[13] M. T. Lusk and A. E. Mattsson, "High-performance computing for materials design to advance energy science," *MRS Bulletin*, vol. 36, no. 3, p. 169–174, 2011.

[14] E. project, "https://hpc-portal.eu/materials/publications."

[15] NVIDIA, "Cable latency in data centers," 2024.

[16] "Helios supercomputer at national centre for nuclear research, poland," 2025.