Prediction model of performance-energy trade-off for CFD codes on AMD-based cluster

Marcin Lawenda, Łukasz Szustak, László Környei

PII: S0167-739X(25)00105-0
DOI: https://doi.org/10.1016/j.future.2025.107810
Reference: FUTURE 107810

To appear in: *Future Generation Computer Systems*

Received date : 29 November 2024
Revised date : 2 February 2025
Accepted date : 7 March 2025

Please cite this article as: M. Lawenda, Łu. Szustak and L. Környei, Prediction model of performance-energy trade-off for CFD codes on AMD-based cluster, *Future Generation Computer Systems* (2025), doi: https://doi.org/10.1016/j.future.2025.107810.

Prediction model of performance-energy trade-off for CFD codes on AMD-based cluster

Marcin Lawenda[a], Łukasz Szustak[b], László Környei[c]

[a]*Poznan Supercomputing and Networking Center, Jana Pawła II 10, 61-139 Poznań, Poland*
[b]*Czestochowa University of Technology, Dąbrowskiego 69, 42-201 Częstochowa, Poland*
[c]*Széchenyi István Egyetem-University of Győr, Győr Egyetem tér 1. tanulmányi ép. B-604, Hungary*

Abstract

This work explores the importance of performance-energy correlation for CFD codes, highlighting the need for sustainable and efficient use of clusters. The prime goal includes the optimisation of selecting and predicting the optimal number of computational nodes to reduce energy consumption and/or improve calculation time. In this work, the utilisation cost of the cluster, measured in core-hours, is used as a crucial factor in energy consumption and selecting the optimal number of computational nodes. The work is conducted on the cluster with AMD EPYC Milan-based CPUs and OpenFOAM application using the Urban Air Pollution model. In order to investigate performance-energy correlation on the cluster, the `CVOPTS` (Core VOlume Points per TimeStep) metric is introduced, which allows a direct comparison of the parallel efficiency for applications in modern HPC architectures. This metric becomes essential for evaluating and balancing performance with energy consumption to achieve cost-effective hardware configuration. The results were confirmed by numerous tests on a 40-node cluster, considering representative grid sizes. Based on the empirical results, a prediction model was derived that takes into account both the computational and communication costs of the simulation. The research reveals the impact of the AMD EPYC architecture on superspeedup, where performance increases superlinearly with the addition of more computational resources. This phenomenon enables a priori the prediction of performance-energy trade-offs (computing-faster or energy-save setups) for a specific application scenario, through the utilisation of varying quantities of computing nodes.

Keywords: CFD performance, energy efficiency, CVOPTS metric, prediction model, HPC computation

1. Introduction

The relationship between performance and energy consumption in computational fluid dynamics (CFD) codes has become an essential domain of research investigation, especially in light of the growing emphasis on sustainable computing and efficient use of High-Performance Computing (HPC) clusters. As supercomputers evolve to tackle increasingly complex problems, their energy demands escalate exponentially, posing significant operational and environmental challenges [1].

Historically, the primary focus has been on establishing performance related metrics for these codes, which are evaluated based on computational speed (e.g. FLOPS), accuracy, and scalability. However, due to the growing financial and environmental implications of energy consumption [2], new metrics that consider energy efficiency in addition to traditional performance assessments are increasingly being considered [3]. This development poses an additional challenge for researchers tasked with creating algorithms that are optimised not only for speed but also for minimising energy consumption. This includes strategies such as energy-aware scheduling, load balancing, and implementing energy-efficient numerical methods.

This study aims to address the research question concerning the optimal selection of resources within a homogeneous computing cluster, specifically in relation to performance and energy efficiency for Computational Fluid Dynamics (CFD) applications, which directly influences the associated operational costs. To evaluate and compare application performance for different hardware configurations, the `CVOPTS` metric was proposed, which takes into account different mesh sizes and numbers of computing nodes. This enabled the creation of an applied predictive model that enables the selection of the best hardware configuration taking into account two criteria: performance and resource consumption (energy). Numerous validation tests have been performed to confirm the effectiveness of this approach, significantly facilitating the process of scheduling and allocating resources.

The structure of the paper is delineated as follows. Section 2 provides a summary of related works across three domains: resource allocation, management of partial differential equation (PDE) solvers, and the establishment of metrics. Section 3 presents the problem formulation from a mathematical perspective. The subsequent chapter (4) details the application utilised for evaluation and the hardware employed for computations. Section 5 focuses on the motivations that inspired the authors in formulating the assumptions for this research. Chapter 6 discusses the evaluation of the Urban Air Pollution (UAP) application's performance and the definition of the `CVOPTS` met-

*Corresponding author.
Email addresses: `lawenda@man.poznan.pl` (Marcin Lawenda), `lukasz.szustak@pcz.pl` (Łukasz Szustak), `laszlo.kornyei@math.sze.hu` (László Környei)

ric. Chapter 7 introduces the prediction model concerning application execution and cluster utilisation costs. The final chapter offers an evaluation of the prediction model, presenting an analysis aimed at finding a balance between high efficiency and the minimisation of resources (energy) required to execute the computational task.

2. Related work

Many scientific publications address the topic of computational efficiency of the HPC systems in the context of low energy consumption [4, 5, 6]. The scope of topics covered is very broad, covering energy-aware scheduling in the context of resource allocation, data partitioning, and workflow scheduling [7, 8, 9].

Performance and energy efficiency are analyzed together presenting a great convergence. In the work [10], the authors highlight that both performance and energy efficiency are closely aligned with Moore's Law. Over the past sixty years, the electrical efficiency of computation has approximately doubled every eighteen months, a rate of change that is comparable to the advancements seen in computer performance and electrical efficiency during the microprocessor era. Their findings indicate that since 1946, the energy efficiency of computation has doubled roughly every 1.57 years. This rate of improvement is slightly slower than that of personal computers (PCs), which saw a doubling of efficiency every 1.52 years from 1975 to 2009. In the same period, the performance of PCs doubled approximately every 1.5 years.

In the context of this study, related works can be analysed from the following perspectives: resource allocation, CFD applications and metrics that are defined and used for the purpose of comparing the achieved results.

2.1. Resource allocation

In the case of the work [11], which is a comparative study of task scheduling in large parallel systems, the authors analyse the possibilities of minimising waiting time, response time, and energy consumption, and maximizing the overall system utilization. The study based on empirical results (22385 tasks) compares thirteen task scheduling policies to analyse their behaviour. The set of task scheduling policies includes priority-based, first-fit, backfilling, and window-based policies, highlighting the strengths and weaknesses of different task scheduling policies and helping to choose the most appropriate one for a given computing scenario. The effectiveness of most job scheduling policies is largely influenced by various workload characteristics, particularly the duration of job execution. The significant degree of imbalance necessitates a deliberate selection of scheduling methods; for instance, narrow jobs are ideally suited for the combination of MinET (minimum estimated execution time) and SJF (smallest job first) with the FF (first fit) technique, whereas they are ill-suited for the MaxET (maximum estimated execution time) policy. Conversely, wide jobs may be executed on machines with lower performance and power capabilities, allowing LJF-PE (largest job first - power efficiency) to optimise energy consumption effectively. In parallel computing environments, resource management should not rely on a singular policy but should instead adopt dynamic and adaptive scheduling strategies.

The paper [12] elaborates an extensive overview of the architectural, software, and algorithmic challenges associated with energy-efficient workflow scheduling across single-core, multicore, and parallel architectures. Additionally, it presents a structured classification of algorithms found in the literature, categorised according to overarching optimisation goals and the specific characteristics of applications. The authors emphasise the importance of support resources that are heterogeneous and dynamic, as dynamic changes in available resources can significantly affect energy and time requirements and should be carefully considered in scheduling. Similar to the previous paper, here too it is suggested to enhance grids and clouds with fast, dynamic, scalable, and adaptive management mechanisms instead of static and inflexible manual solutions. This can be done by developing new algorithms that leverage dependencies between different tasks to allocate slack. Furthermore, scheduling should take into account the adaptability of priorities as execution progresses and user-defined goals.

2.2. Management of partial differential equation solvers

The study [13] introduces an elastic computational approach that dynamically modifies the resources dedicated to the simulation during execution. To determine the appropriate quantity of resources necessary for executing a computational task, the efficiency of communication is considered. Based on various analytical evaluations, resources are subsequently increased or decreased to align with this criterion, ultimately ensuring an effective simulation process. The communication performance of CFD simulations is evaluated using real execution time measurements using the TALP library [14]. The number of cores needed to meet this goal is estimated on the fly, taking into account the performance target. If the number of cores for simulation needs to be expanded or reduced, the workflow manager (PyCOMPS [15]) interacts with SLURM, and once cores are allocated, the CFD code is restarted, and the simulation continues.

The authors of the paper [16] examine the features of CFD applications and develop a modelling approach that allows the decomposition of these applications into multiple subtasks represented by Directed Acyclic Graphs (DAG). They subsequently introduce a hierarchical framework for resource organisation within a computational grid environment. In conclusion, they address the scheduling strategy pertinent to the outlined scenario and evaluate the proposed algorithm through simulation experiments. The authors assert that the computational grid is appropriate for CFD applications by segmenting it into numerous sub-problems that can be addressed through distributed computations with minimal communication frequency. It is posited that several sub-problems derived from a single large-scale CFD application can be executed in parallel as sub-tasks within the grid framework, while also taking into account the interdependencies among these sub-tasks.

2.3. Metrics

A significant challenge lies in the development of clear metrics that will definitively address which HPC system provides superior capabilities while maintaining reasonable operational costs. They focus on energy-aware techniques, tools, and architectures (clusters, grids, and clouds) used in high-performance computing. Namely, the paper [17] widely describes the optimisation metrics used, including energy measurement tools, and benchmarking, forecasting, and simulation methods for the described problem. The authors note that in terms of metrics designed for optimisation, numerous studies focus on the goal of reducing energy consumption while maintaining minimal impact on performance. This is typically achieved by identifying the specific application phases that present the greatest potential for energy savings. Furthermore, there is a dearth of studies that consider network and memory components in this context. The authors highlight a significant gap in the literature regarding automatic profiling and tuning for parallel applications running on hybrid systems that include both CPUs and GPUs.

The paper [18] introduces a vector-valued metric aimed at enhancing efficiency in supercomputing. The metric is composed of two scalar components: one representing performance and the other denoting energy efficiency, emphasising the notion that energy is equally significant as performance. Notably, the focus of the paper lies more on the dimensionality of the metric space, advocating for the use of a vector metric, rather than on the specific measurement protocols for obtaining each scalar value. An analysis conducts of the historical and current state of the supercomputing industry in relation to efficient supercomputing practices.

3. Problem formulation

In order to better understand the complexity of the problem, this chapter presents a formal definition of the optimization task of resource allocation on a computing cluster.

The symbols used in the definition of the problem are summarized in the Table 1.

For the sake of clarity in notation, we consider the processors to be arranged in a star configuration of set $\mathcal{P}$ of processing elements $i = 1, \ldots, m$. The processing elements are arranged in ζ units on Υ nodes. To choose the best configuration both in terms of performance and energy, we consider various sets $\mathcal{P}_j \subseteq \mathcal{P}', j = 1, \ldots, \kappa$ of processing units. Each set $\mathcal{P}_j$ is composed of m_j processing units (e.g. processor cores).

Based on the diversity of the computational environment, three distinct types of star configurations can be identified, as referenced in [19], [20] and [21]: Unrelated processors, Uniform processors, and Identical processors. Presented analysis will focus on the last category, presuming that all processors and tasks share identical communication speeds and computational speeds. Therefore, for all $\forall_{\mathcal{P}_{ij} \in \mathcal{P}} A_{ij} = A, C_{ij} = C$. Identical processors can be perceived as a specific case of homogeneous processors, exemplified by the execution of the same parallel program in a uniform environment with varying input data sets. To simplify the model, we ignore the issues related to the initial time of loading the code and data and sending them to the individual processing units. However, we do take into account the communication time between processors during processing. The sequence of activating the processors is arbitrary. We assume that all resources (processors) from the pool are available and there are no constraints related to scheduling other tasks. Due to the equal division of work (α_j) and the efficiency of the processors, all tasks are completed at the same time ($t_{ij} = t_j$). It is also assumed that the initial time of sending data to the processors and returning the results (saving them to disk) is negligible. Tasks are distributed in one cycle (single load).

The size of each chunk α_j is the same for all processors in $\mathcal{P}_j$. For identical processors, the computation time for α_j load units is expressed as $\alpha_j A$. The communication time between processing units under processing is considered and is represented as $\sigma_j C$.

Utilising processors in $\mathcal{P}_j$ incurs a cost (e.g. energy) of $f_j + \alpha_j l_j$ for each of them. The final restriction considers the memory capacity that should be restricted to B load units, the load block must not exceed this limit, thus $\alpha_j \leq B$.

The problem entails a bi-criteria optimization scenario. The two criteria under consideration are the schedule length, denoted by C_{max}, and the cost associated with processor usage (e.g. energy), represented by $G = \sum_{j \in \mathcal{P}'} (f_j + \alpha_j l_j)$, where $\mathcal{P}'$ refers to the set of processors in use. This bi-criteria optimisation problem can be simplified into two more basic problems: (i) the minimisation of C_{max} subject to the constraint that $G \leq \overline{G}$, and (ii) the minimisation of G subject to the condition that $C_{max} \leq \overline{C_{max}}$. Here, $\overline{G}$ signifies a predetermined upper limit on the cost associated with the schedule, while $\overline{C_{max}}$ signifies a specified upper limit on the schedule length. Both simplified problems can be addressed in polynomial time through linear programming techniques, assuming that the set $\mathcal{P}'$ of employed processors and their activation sequence are allowed to vary without restriction.

The optimality criterion is schedule length (makespan) $C_{max} = max\{c_j\}$, where c_j is task j completion time.

minimise C_{max} and $\overline{G}$, subject to:

$$\sigma_j C + \alpha_j A \leq C_{max} \qquad j = 1, \ldots, \kappa \tag{1}$$

$$\sum_{i=1}^{m_j} (f_j + \alpha_j l_j) \leq \overline{G} \qquad j = 1, \ldots, \kappa \tag{2}$$

$$0 \leq \alpha_j \leq B \qquad j = 1, \ldots, \kappa \tag{3}$$

$$\sum_{i=1}^{m_j} \alpha_j = V \qquad \forall j \in \kappa \tag{4}$$

$$\zeta \times \Upsilon_j = |\, m_j \,| \qquad j = 1, \ldots, \kappa \tag{5}$$

In the above formulation constraint (1) guarantees that computations and communications are performed in an admissible interval for all $\mathcal{P}_j$ sets. By inequality (2) total cost of the schedule does not exceed the limit $\overline{G}$. Given that the cost remains constant for each processing unit $\subset \mathcal{P}$, it suffices to aggregate their values across all processing units $(1, \ldots, m_j)$. The constraint (3) guarantees that the capacities of the memory buffers will not be surpassed within the designated number of processing units ($\mathcal{P}_j$) and the resultant size of task allocation (α_j). By

Symbol	Description
A	processing rate (reciprocal of speed) of P
α_j	fraction of load (mesh) assigned processing unit in set of P_j
B	memory size of processor P
C	communication rate (reciprocal of bandwidth) of the link from P_i to cooperating processors
c_j	schedule length for P_j processors
$C_{max} = \max\{c_j\}$	schedule length
$G = \sum_{j \in \mathcal{P}'}(f_j + \alpha_j l_j)$	total cost of the schedule on processors in set $\mathcal{P}_j$
f_j	fixed part of the cost of using processors in $\mathcal{P}_j$
$\overline{G}$	an upper limit on cost G
κ	number of sets of processing units
l_j	coefficient of the linear part of the cost of using $\mathcal{P}_j$
m_j	number of processing units in set of $\mathcal{P}_j$
$\mathcal{P}$	set of available processing units
$\mathcal{P}'$	set of processing units participating in any computation
$\mathcal{P}_j$	set of processing units participating in the computation j
σ_j	communication load between any of the processors in set of $\mathcal{P}_j$ and the cooperating processors
t_{ij}	the time moment when $\mathcal{P}_{ij}$ finishes computing
Υ_j	number of nodes used in set of $\mathcal{P}_j$
V	single load size
ζ	number of i processing units per node Υ

Table 1: List of symbols used in the problem formulation

equality (4) it is ensured that all the load is processed and it is true for all tested sets ($\mathcal{P}_j$). Due to the assumption of equal division of tasks ($\alpha_{ij} = \alpha_j$) within given division of $\mathcal{P}_j$, it is enough to sum the appropriate number of times (m_j) the size of each load (α_j). Finally, an equality (5) provides the appropriate granularity of resource selection, ensuring that all available processing units (ζ) within selected nodes (Υ_j) will be utilized within a given number of resources m_j in set of $\mathcal{P}_j$.

The mathematical approach presented is one of the possible solutions to the problem of optimal selection of resources necessary for efficient execution of calculations. However, assuming the complexity of the mathematical process, in the further part of the article we present an alternative method of its determination for the CFD computational task. It consists in empirical evaluation of the system performance and determination of a set of resources ($\mathcal{P}'$) best in terms of performance and minimal energy consumption.

4. Application and HPC system overview

4.1. Implementation domain: Urban air pollution model

For the current benchmark, the performance of the CFD module of the Urban Air Project (UAP) developed under the HiDALGO2 project [22] was utilised, which is based on the air pollution model developed by Horváth et al. [23]. The OpenFOAM based UAP-FOAM module implements the simulation air flow and pollution spread using the air pollution model of UAP [24]. The Reynolds-Averaged Navier Stokes equations [25] are solved with weather-based boundary conditions for air flow coupled with an advection-diffusion equation with traffic-based source terms for pollution spread. A two step method is used for the simulation: first, a simpleFoam based solver is used to calculate a steady state for a specific time, which is used as an initial condition for a pimpleFoam based solver, which simulates time evolution of wind speed and pollution distribution.

In this paper, we focus on the steady-state part with simpleFoam, limiting the number of iterations for the solver to around 400-600 timeStep. However, it is worth remembering that we base our results on the average time of a single timeStep. For the geometry, the urban area of the Hungarian city, Győr is meshed, depicted on Figure 1. Boundary conditions are based on weather conditions, provided by ECMWF via the weather service interface, polytope [26]. Also, pollution source is based on traffic simulation and emission calculation using the COPERT model [27]. The OpenFOAM version used in these investigations was com version v2112 [28]. Simulation results can be observed in Figure 2.

The SCOTCH method is employed for work distribution among processes through domain decomposition, utilizing the

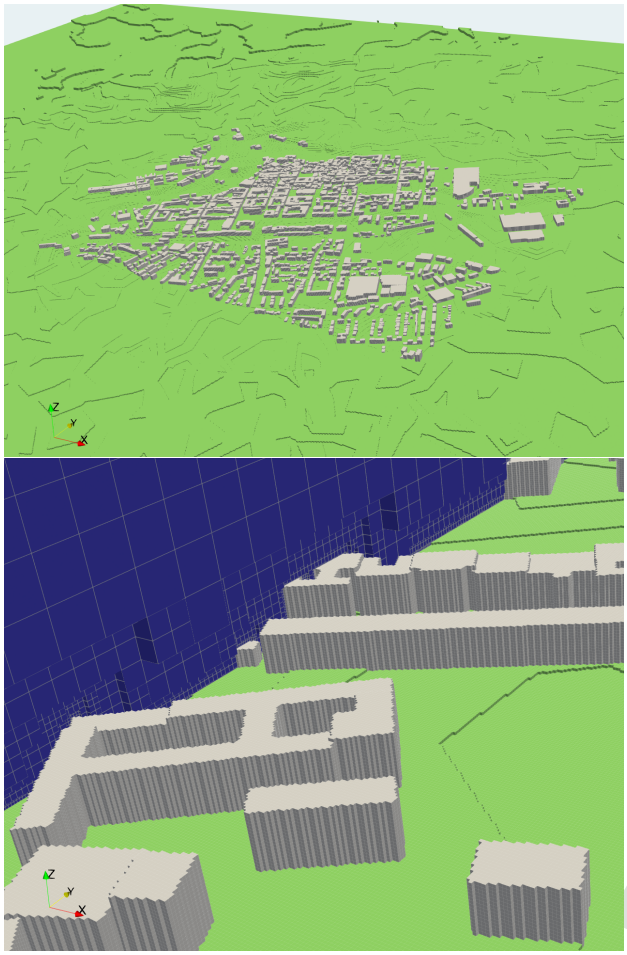

Figure 1: Buildings and ground surfaces of the city of Győr within the high resolution 3D mesh generated for benchmarking. Large scale (top) shows urban area with ground (green) and buildings (grey). Local view (down) shows surface elements for ground and building, as well as volume elements for air (blue).

Figure 2: Visualisation of simulation results of the UAP-FOAM model within the city of Győr. Geometric representation of buildings (brown), ground (grey), etc. come from the original city geometry, not the simulation mesh. Air flow is depicted with streamlines, while vectors indicate flow direction. Pollution concentration is indicated by the grey fog between the buildings. Lighter and darker colours indicate lower and higher concentration of pollution.

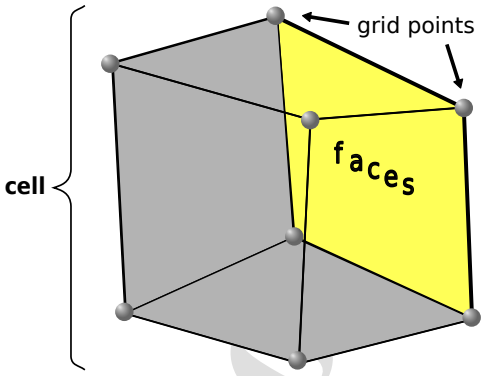

Figure 3: Visualisation of a sample cell having the form of a polyhedron that creates the mesh for the model and covers 8 grid points and 1 face for the MPI exchange

scotch library included in the OpenFOAM distribution [29]. This approach implements a weighing technique to partition the computational domain into segments, aiming to balance the sizes of these segments while minimising the number of interfaces between the decomposed sections. Consequently, the quantity of faces at the inter-process boundaries significantly affects the time allocated for communication between neighbouring processes.

In this finite volume simulation the simulation domain (the air around and within the city) is divided up to several polyhedral cell volumes, hence the cell count of every mesh. Faces are considered the even surfaces on these cells (Figure 3). While internal faces connect two cells, external faces will count as boundaries, like ground and building. After domain decomposition, internal faces may become communication patches between processors. The grid points of the mesh is made up from the cell vertex points.

Meshes used for these benchmarks are octree based and are generated using the in-house SZE tool octreemesher. All meshes use the same geometry, albeit at different resolutions, and are listed in Table 2. All input data including mesh, weather boundary and pollution source are precalculated and present in files for the benchmark.

A finer mesh with smaller cells significantly influences simulation accuracy by improving the resolution of flow features and enabling more precise capture of small-scale vortices, boundary layers, and turbulence structures, provided that grid refinement is appropriately applied in critical regions. For health impact assessments, pollutant concentrations are sampled at a height of 2 meters, where targeted grid refinement has been applied to improve accuracy. The current vertical resolution near the ground is 1 meter for the high-cell-count mesh and 2 meters for the mid-cell-count mesh, resulting in a significant difference in the simulation outcomes.

Table 2: Total numbers of cells for different meshes for UAP

uxlow	ulow	mlow	low	mid	high
36248	139937	228263	728162	3227275	14332247

4.2. AMD EPYC cluster and software stack

In this work, we use the AMD-based cluster consisting of 40 nodes with two AMD EPYC 7763 CPUs of 64 cores each, clocked at 2.45GHz, and 256GB of DDR4-3200. A single AMD EPYC 7763 CPU features Thermal Design Power (TDP) at the level of 280 W [30]. The system operates with SMT disabled and turbo boost enabled. This cluster is interconnected with InfiniBand HDR.

The OpenFOAM-based implementation of UAP module is benchmarked. All OpenFOAM kernels are compiled with the AOCC compiler [31] and linked against the MPI library pre-installed by platform vendors (OpenMPI v.4.1.5). The AOCC compiler (v.4.1.0) is used with the optimisation flag `-O3` and architecture-specific compiler arguments `-march=znver3` for AMD EPYC Milan CPU.

5. Work motivation

The main objective of this study is to assess the performance and energy implications of utilising varying quantities of compute nodes within a 40-node cluster through benchmarking the UAP model. To fulfil this objective, authors analyse and contrast execution duration, speed enhancements, instances of linear acceleration overshoot, costs associated with cluster utilisation, and projected total energy usage.

The execution time is measured by extracting the time stamps written by OpenFOAM as "Execution Time" and subtracting the first value from the last value of the OpenFoam simpleFoam solver. In this way, the time spent on initialisation is discarded, although the execution time of one iteration less is measured. The strong scaling measurement concerns the speedup for a fixed problem size and a different number of nodes. Furthermore, we outline a linear overshoot of the speedup, which compares the linear speedup and the achieved strong scaling speedup.

The utilisation cost of the cluster, measured in core-hours (core-h), is accounted for execution time and the number of reserved cores (one core-h represents the usage of one CPU core for one hour). Since the energy measurement capabilities are limited in the tested cluster, we propose to estimate the total energy consumption assuming that TDP refers to the maximum power requirements under load of each processor. As a result, the total energy consumption is approximated by taking into account the number of compute nodes used (number of processors), the execution time, and the TDP metric.

Figure 4 presents an example of experimental performance results obtained for the UAP model by testing the mesh of size with 14332247 cells. This figure depicts the performance-energy comparison between different numbers of computing nodes, including setups with 1, 2, 4, 16, 24, 32, and 40 nodes (128 up to 5120 cores).

The test revealed that the 32-node configuration achieved the quickest execution time, approximately 39.38 seconds, as illustrated in Figure 4a. This configuration demonstrates a performance improvement of roughly 26.5 times when compared to the 1-node setup shown in Figure 4b. The 32-node configuration necessitates around 44.80 core-hours of computational time and utilises approximately 0.196 kWh of energy, as depicted in Figure 4d.

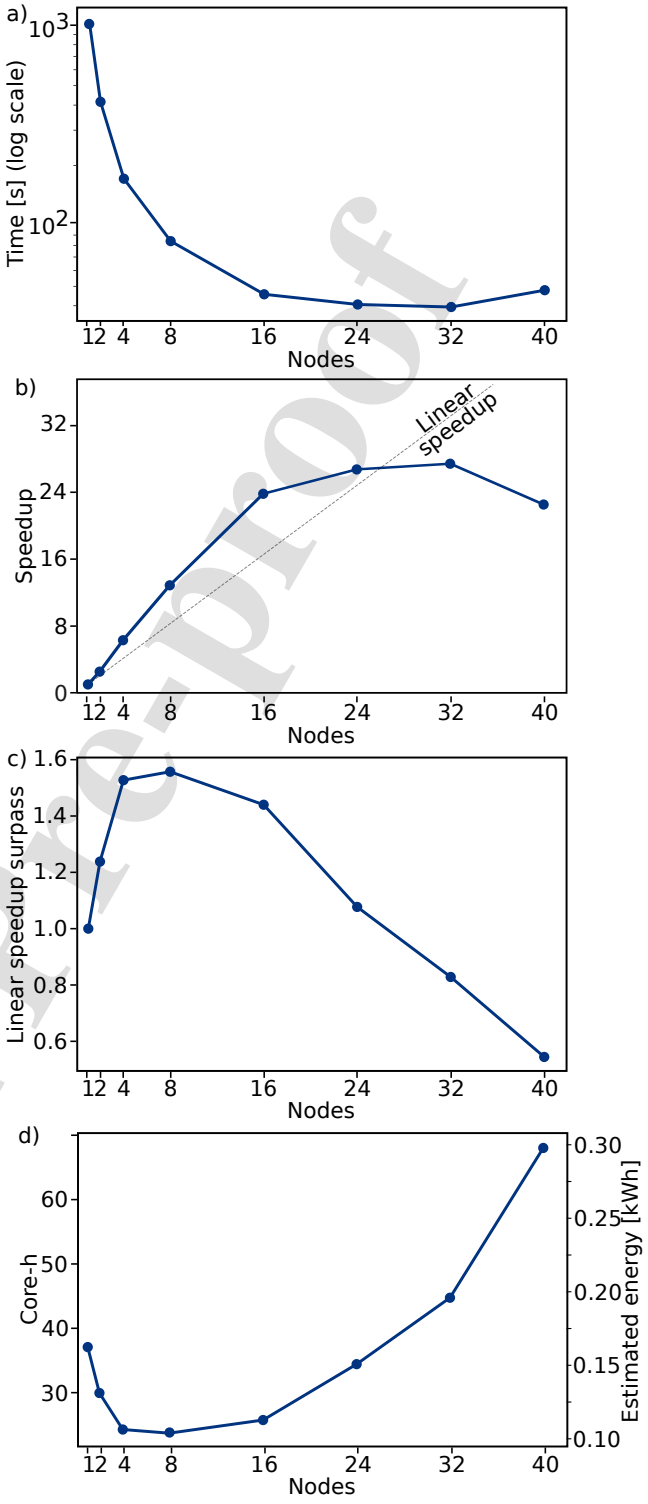

Figure 4: Performance results obtained for UAP of mesh size called high with 14332247 cells on a 40-node cluster, including a) execution time, b) measurement of strong scaling, c) linear speedup surpass, and d) cluster cost utilisation

The highest performance improvement is observed for the

setup with 16 nodes (Figure 4b), reducing the calculation time from 1043.93 to 45.31 seconds and accelerating the computation about 23.03x faster in comparison to 1 node. Consequently, the achieved speedup races 1.44x the linear speedup (Figure 4c), increasing the performance superlinearly. The total utilisation cost of the 16-node setup requires about 25.77 core-h, while the predicted total energy consumption is kept at the level of about 0.113 kWh (Figure 4d).

As shown in Figure 4c and Figure 4d, the 8-node setup features the best linear speedups surpass and cluster utilisation cost. In this instance, the simpleFoam kernel is executed in 83.8 seconds, achieving a super-linear speedup and accelerating computations 12.45 times faster than a single node. It enhances the linear speedup by approximately 1.55 times. This computation consumes 23.83 core-hours and necessitates 0.104 kWh for resource utilisation and energy expenditure, respectively.

The performed benchmark reveals that the 16-node setup achieves desirable utilisation cost of the cluster and performance trade-offs, enabling both computing-faster and energy-save setups for the processed kernel. This setup features slightly slower computation times than the 32-node configuration and also keeps cluster utilisation costs close to the level of the 8-node outcomes.

More precisely, employing 32 computing nodes compared to the 16-node setup processes the simpleFoam kernel 1.15 times faster. However, the cluster utilisation cost is 1.74x and it requires 2x more nodes. In this comparison we can sum up, the performance advantage from the 32-node setup seems to be inadequate compared to the incurred costs.

Furthermore, when comparing the 8-node configuration to the 16-node arrangement, we observe that the cluster utilisation costs are nearly equivalent, with a slight preference for the 8-node setup. As anticipated, the 16-node configuration completed the computations 1.85 times more quickly, albeit requiring double the number of nodes. Therefore, while the 8-node setup incurs lower cluster utilisation costs, the computation duration may restrict the overall performance benefits. Moreover, given that this configuration demonstrates a significant super-linear speedup exceeding 1.55 times the linear speedup, we anticipate superior utilisation of HPC resources in comparison to other node configurations.

6. Metric definition and application evaluation

The subsequent stage of our research will concentrate on thoroughly examining the effects of various workloads and computational duration on total energy consumption and performance. This understanding will enable us to make more informed choices regarding the optimisation of performance and energy efficiency in computing systems. To reach this aim, the `CVOPTS` (Core VOlume Points per TimeStep) metric is introduced, which allows a direct comparison of the parallel efficiency for applications by testing different mesh sizes and numbers of computing nodes. The `CVOPTS` metric is calculated as:

$$CVOPTS = \frac{cells\ per\ core}{timeStep} \tag{6}$$

where the `cells per core` parameter refers to the average number of mesh cells processed by every core while `timeStep` means the average computation time of a single timeStep of the simpleFoam kernel. This metric helps us estimate computing efficiency for different mesh sizes, indicating better platform utilisation for the higher `CVOPTS` level. The value of `CVOPTS` depends on a series of factors, including hardware-/application-specific features (see our previous works [32] for more details).

The results provided emphasise the alteration of grid size and the quantity of computational nodes to identify local maxima, which signify the ideal configuration of cells per core and the number of nodes. Table 3 shows cells assigned to a single core considering a variety of mesh sizes and different numbers of nodes. It outlines how the number of cells changes considering (i) a variety of mesh sizes and a fixed number of computing resources (see rows in the table), as well as (ii) a fixed mesh size and a different number of nodes (see columns in the table). Table 4 presents measurements for the average execution time of a single timeStep obtained for a variety of mesh sizes and different numbers of nodes.

Table 3: Number of cells per core for a variety of mesh sizes and different numbers of nodes

Number of nodes	Mesh sizes					
	uxlow	ulow	mlow	low	mid	high
1	283	1093	1783	5689	25213	111971
2	142	547	892	2844	12607	55985
4	71	273	446	1422	6303	27993
8	35	137	223	711	3152	13996
16	18	68	111	356	1576	6998
24	12	46	74	237	1051	4665
32	9	34	56	178	788	3499
40	7	27	45	142	630	2799

Table 4: Average execution time [s] of a single timeStep for simpleFoam kernel

Number of nodes	Mesh sizes					
	uxlow	ulow	mlow	low	mid	high
1	0.016	0.026	0.028	0.045	0.322	2.616
2	0.020	0.027	0.035	0.039	0.140	1.057
4	0.022	0.034	0.038	0.030	0.080	0.428
8	0.027	0.034	0.036	0.035	0.062	0.210
16	0.028	0.044	0.039	0.035	0.052	0.114
24	0.036	0.041	0.050	0.036	0.057	0.101
32	0.034	0.045	0.047	0.046	0.058	0.099
40	0.042	0.049	0.060	0.051	0.066	0.120

Considering all cell configurations per core (presented in Table 3) and performance measurements (described in Table 4), we investigate the performance metric `CVOPTS` for the simpleFoam kernel. Figure 5 and Figure 6 deliver the `CVOPTS` values for a variety of mesh sizes obtained for a fixed amount of computing resources. More precisely, Figure 5 illustrates `CVOPTS`

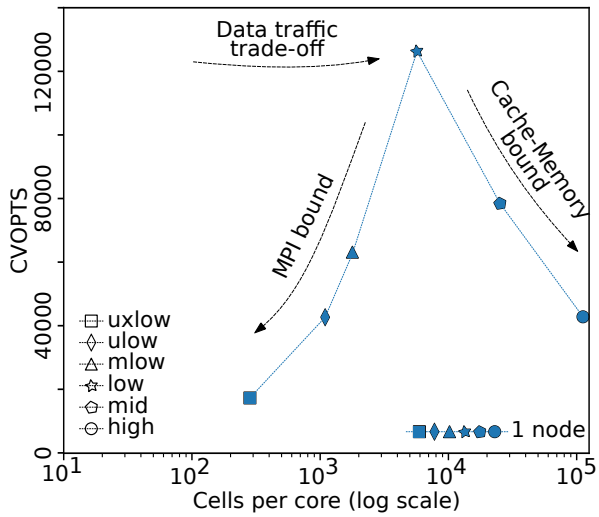

Figure 5: CVOPTS performance metric for the simpleFoam kernel obtained by testing a variety of mesh sizes on 1 node

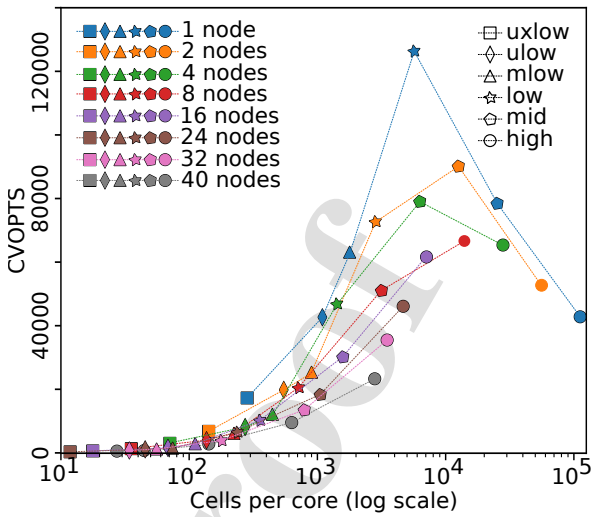

Figure 6: CVOPTS performance metric for the simpleFoam kernel obtained by testing a variety of mesh sizes and different number of nodes

obtained on 1 node, while Figure 6 expands proposed measurements on the 2-, 4-, 8-, 16-, 24-, 32-, and 40-node setups, by testing all meshes for every node setup individually.

As illustrated in Figure 5, CVOPTS increases until a turning point (inflection point) and then decreases. The turning point is observed for a small grid size. To explain the CVOPTS results, a closer look at the hardware specifications and application features is necessary. In our earlier scrutiny [32], we contend that a reduction in the number of cells per core leads to an escalation in the costs associated with MPI communication, thereby progressively constraining the potential performance that can be attained. On the other hand, the increase in the number of cells causes higher and higher demand on the cache and main memory subsystems. As a result, the overall performance becomes increasingly limited through the data traffic between the cache and main memory subsystems. Conversely, the turning point of the CVOPTS curve presents a trade-off among MPI, cache, and main memory concerning data traffic communication, thereby facilitating optimal hardware utilisation. The findings presented in this report suggest the ideal number of cells per core, which aligns with a low grid size in a single-node configuration.

The CVOPTS curves show similar behaviour when testing all meshes on different numbers of nodes separately (Figure 6). However, we observe a drop in the CVOPTS trend between subsequent node setups. The configuration with a single node exhibits the most favourable trend for CVOPTS, whereas an increase in the number of nodes leads to a decline in the CVOPTS trend, culminating in the 40-node configuration, which presents the least beneficial trend for CVOPTS.

The increasing demand for inter-node MPI communication may explain the observed decreasing CVOPTS trends for different numbers of nodes. Figure 7 tracks the average number of faces shared with other MPI processes depending on different numbers of nodes and the underlined mesh sizes. The larger number of nodes causes workload distribution across more cores and requires more communication, resulting in the drop of the CVOPTS trends.

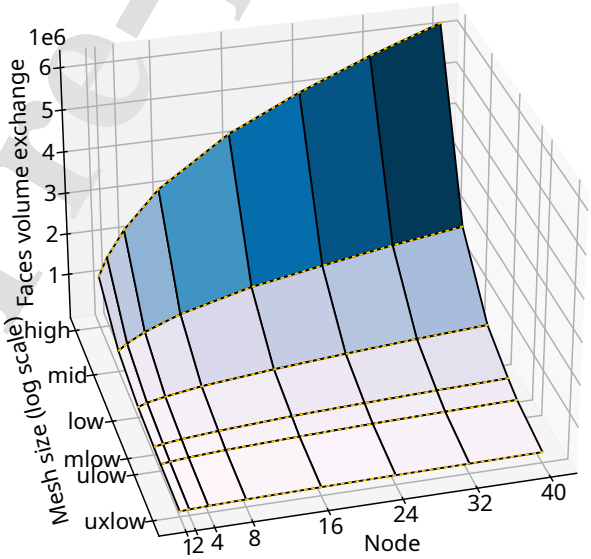

Figure 7: The average number of faces shared with other MPI processors

Moreover, Figure 8 reveals the calculated trend of CVOPTS for a fixed mesh size (high) obtained on different number of nodes. The trend observed for CVOPTS exhibits a pattern akin to that of the prior study: it rises to a certain peak before subsequently declining. The results reported here indicate the optimal number of cells per core that refer to the 8-node setup when processing the high mesh size. Consequently, the 8-node setup features the data traffic trade-offs, resulting in increased performance superlinearly where the improvement in execution time is greater than the proportional increase in computing resources. This occurs due to better cache utilization and lower MPI communication costs. Following the previous examination, the MPI communication cost increases when more and more cores (nodes) are used, consequently increasingly limit-

ing CVOPTS for more than eight nodes. Opposite, utilising a diminishing number of nodes results in an increasing number of cells per core, which escalates the demand for cache-memory subsystems and constrains overall performance.

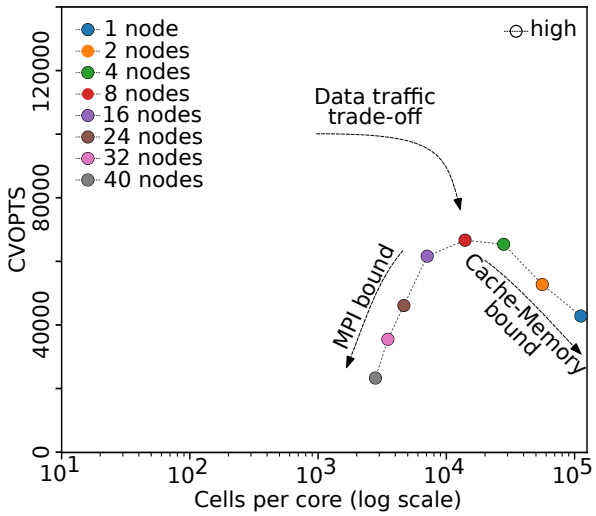

Figure 8: CVOPTS performance metric for the simpleFoam kernel obtained by testing a different number of computing nodes and a fixed mesh size

7. Prediction model of application execution and cluster utilisation costs

The CVOPTS performance metric becomes essential for evaluating and balancing performance with energy consumption to achieve cost-effective hardware configuration. It helps us to indicate the best number of cells per core showing (i) how CVOPTS varies with different meshes for a given node setup, and (ii) how CVOPTS varies with the number of nodes for a fixed mesh size. Based on the empirical results, we deliver the CVOPTS-based prediction model that estimates the CVOPTS trendline to approximate application runtime and cluster utilisation costs.

7.1. CVOPTS-based prediction model

The ultimate objective in addressing this challenge is to create a regression-based model that utilises the CVOPTS measurements collected from fixed node configurations and various mesh sizes. This model aims to forecast the overall CVOPTS trendline specific to a particular mesh size and differing node quantities. To this end, we first select the CVOPTS measurements that can be used as input data to fit the data within a polynomial function. In our experiments, we emphasise that a single-node setup does not incur any inter-node MPI data traffic costs, while the experiments performed on eight and higher numbers of nodes do not feature turning points of CVOPTS curves for tested mesh sizes (see Figure 6). Consequently, we follow the remaining CVOPTS measurements obtained on 2- and 4-node setups that represent the trendline CVOPTS points for cells per core of range [71, 55985] (see Table 3 and Figure 6).

Secondly, we develop the Python-based script using a NumPy library to reveal a nonlinear relationship between the selected CVOPTS measurements and cells per core. To achieve the goal of the best fit curve and defining a polynomial function, we employ both poly1D and polyfit methods offered with the help of the NumPy library and use decimal logarithmic transformations for input data. As a result, this investigation enables us to apply a polynomial function model to the equation formulated in the following manner:

$$f(x) = 498x^6 + 3939x^5 - 146400^4 + 1087000x^3 - 3514000x^2 + 5292000x + 3036000$$

In our example, we propose to fit a polynomial of degree 6 to capture the underlying trend of CVOPTS. It refers to the relationship between the cells per core of range [71, 55985] and their CVOPTS measurement points of 2-/4-node setups. Figure 9a illustrates the CVOPTS trendline based on the fitted model.

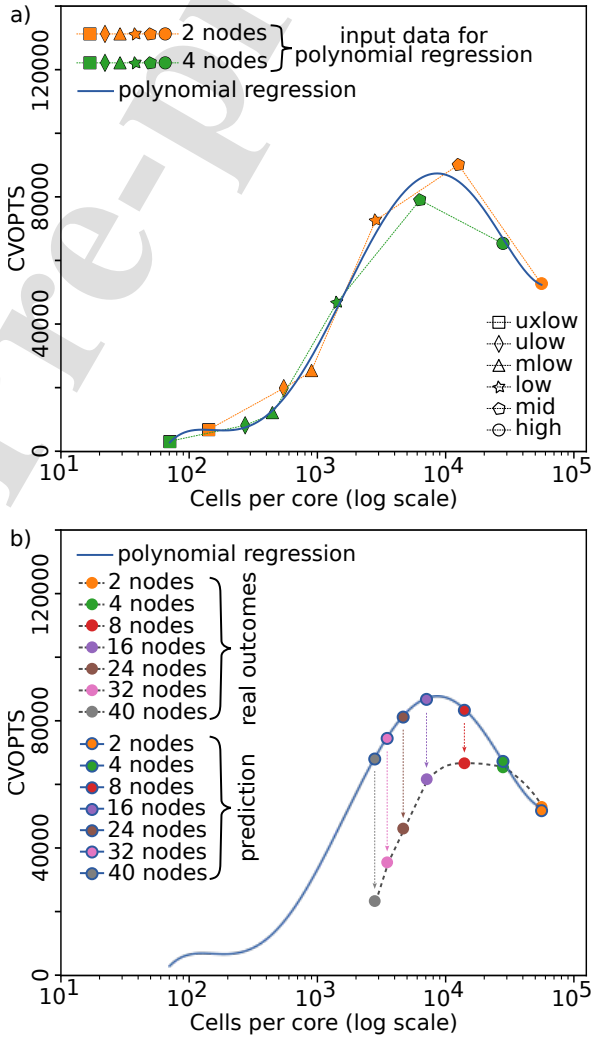

Figure 9: The CVOPTS-based prediction model: a) polynomial regression fitted on 2-/4-node outcomes, and b) model examination for mesh size high and different numbers of node-setups

The defined fit function of the CVOPTS trendline enables the approximation of the value of CVOPTS for a given mesh size

and the various numbers of nodes. However, the approximation is constrained to the range [71, 55985] of cells per core. Therefore, for example, let's focus on a large grid size with 14332247 cells (see Table 3), we will limit the range of node configurations specifying the minimum and maximum number of cores to 256 $\approx \frac{14332247}{55985}$ and 201862 $\approx \frac{14332247}{71}$, respectively. This specifies a range of [2,1577] in the number of nodes that need to be considered, assuming 2x64 cores per node.

Figure 9b demonstrates the prediction results of CVOPTS for the high mesh size and 2, 4, 8, 16, 24, 32, and 40-node setups compared to real outcomes obtained during tests. As expected, the prediction fits perfectly for 2-node and 4-node setups since the applied regression model is trained based on the 2-node and 4-node CVOPTS points. However, for an increasing number of nodes, the CVOPTS-based regression model predicts CVOPTS points that are increasingly different from the obtained results. Note that despite the increasing prediction error, the predicted CVOPTS trendline shows a similar behaviour to the actual results: it increases until a turning point and then decreases.

7.2. *Faces-based extension for prediction model*

To explain the increasing prediction error of the approximated CVOPTS curve, a closer look at the MPI data traffic between MPI processors is required. Following the CVOPTS-based prediction model that approximates CVOPTS for a given mesh size and the various numbers of nodes, we examine the data traffic exchange of the faces shared between MPI processors by fixing the mesh sizes and testing different numbers of nodes.

As illustrated in Figure 7 and Figure 10a, the volume of faces exchange with other MPI processors radically increases when applying more nodes for a given problem size. Since the proposed CVOPTS-based prediction model is based on 2-/4-node setups, the increased communication costs for more nodes are not correlated with the prediction. Consequently, it generates increasing prediction error of the approximated CVOPTS curve. To overcome this limitation, we propose to calibrate the CVOPTS-based prediction model by including communication costs in the estimates.

The communication costs can be modelled as the faces traffic ratio that reveals how faces volume grows for the increased number of nodes. The faces traffic ratio is calculated separately for every mesh size by dividing the total number of faces obtained on subsequent node-setups by the total number of faces from the 4-node setup used in the proposed CVOPTS-based prediction model. Figure 10b shows the faces traffic ratio calculated for all underlined mesh sizes and 4-, 8-, 16-, 24-, 32-, and 40-node setups. In addition, following the obtained measurements, we apply the linear regression model that attempts to simulate the relationship between the faces traffic ratio and the number of nodes (Figure 10). As a result, this linear regression simulates the curve of faces traffic ratio for 4 to 40 nodes.

For the purpose of this work, the predicted curve of the faces traffic ratio is experimentally used for calibrating the CVOPTS-based prediction model. To reach this aim, we propose reducing the predicted CVOPTS by the factor of the estimated ratio for faces traffic. For a predetermined problem size, the CVOPTS-based prediction model calculates the CVOPTS curve, which is

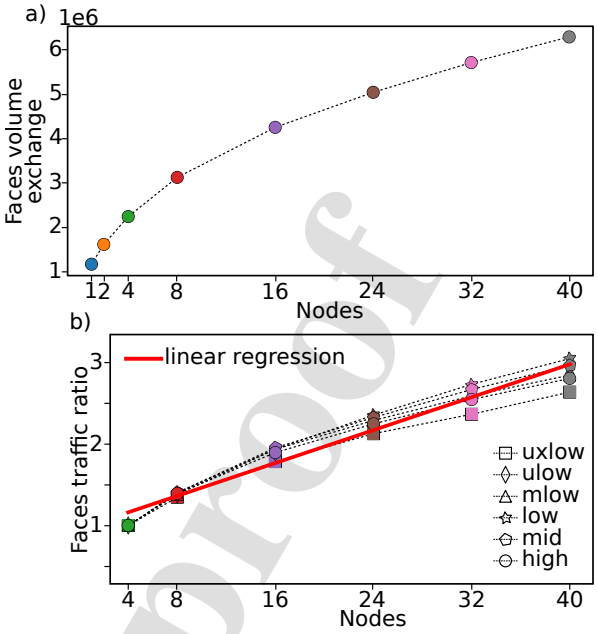

Figure 10: The data traffic volume of the faces shared between MPI processors (a), and the faces traffic ratio calculated in comparison to 4-node setup (b)

subsequently allocated to various node counts based on the grid dimensions and the number of cells designated per core. Each node configuration undergoes calibration of the CVOPTS prediction through the introduced faces-based extension, which involves decreasing its value in accordance with the face ratio factor.

Figure 11 demonstrates the prediction results of the CVOPTS-based model with enabled faces-based extension for the high mesh size and 2-, 4-, 8-, 16-, 24-, 32-, and 40-node setups. The predicted outcomes are examined with comparison to real measurements collected during the testing process.

As shown in Figure 11, the predicted trendline CVOPTS behaves similarly to the actual measurements: it increases until a turning point and then decreases. Furthermore, the proposed calibration enables reducing prediction error of the approximated CVOPTS curve. Table 5 shows the relative errors of the CVOPTS values to the estimated values from the prediction model as well as the faces-based extension method.

Table 5: The relative errors $E1_{rel}$ and $E2_{rel}$ between the measurement of CVOPTS and the estimated values from the prediction model and the faces-based extension method

#Nodes	2	4	8	16	24	32	40
$E1_{rel}$	0.6	2.8	25.4	41.0	76.6	110.7	192.8
$E2_{rel}$	4.4	10.6	8.7	22.5	22.0	22.3	7.3

8. Pursuing performance-energy trade-off

This section aims to assess the proposed prediction model and investigates the performance-energy trade-off. We exam-

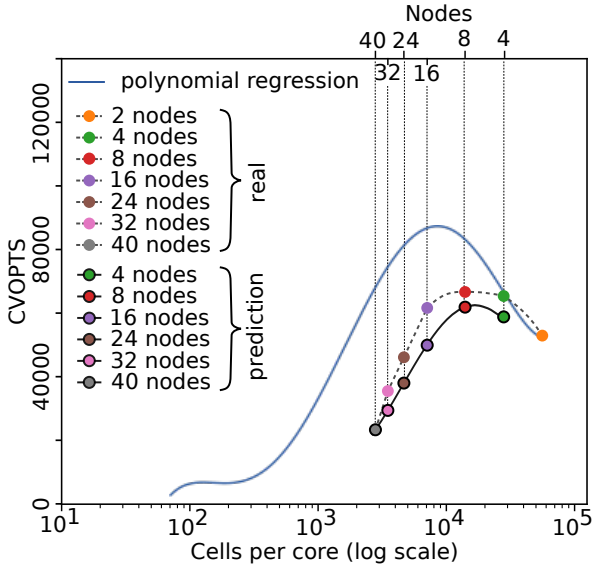

Figure 11: The faces-based extension for CVOPTS-based prediction model

ine the proposed prediction model by testing a new mesh size consisting of 21673212 cells in total. To this end, the CVOPTS-based prediction model calculates the CVOPTS curve following the predetermined problem sizes (see Table 3). The indicated polynomial regression function for CVOPTS is associated with various node counts based on the new mesh size and the number of cells designated per core. Then, the CVOPTS trendline is calibrated through the introduced faces-based extension, which involves decreasing its value by the face ratio factor. As a result, we predict the CVOPTS values for 4 to 40 nodes when performing the 21673212 cells (Figure 12a).

The performed model reveals the node-setups that feature the highest CVOPTS values, including the top 5 configurations with 9, 10, 8, 11, or 7 nodes. In contrast, the CVOPTS values decrease when using larger and smaller numbers of nodes than select top 5 setups. To examine the predicted values, we perform tests for different node-setups with special emphasis on the top 5 list. As shown in Figure 12a, the predicted values of CVOPTS fit the real measurements, where the best results are obtained for 8- and 7-node setups.

According to the CVOPTS performance metric (see equation (6)), we propose to estimate application execution time for a single timeStep, and then approximate the cluster utilisation costs. The Estimated execution Time of a single timeStep (ETS) can be simply defined for every node-setup as:

$$ETS = \frac{cells\ per\ core}{CVOPTS}$$

where the *cells per core* parameter is associated with the number of applied nodes and calculated based on a given mesh size and the number of cells designated per core, while CVOPTS brings the predicted values from the proposed model. Consequently, the utilisation cost of the cluster for a single timeStep is approximated as the product of ETS and the total number of cores. Furthermore, the total energy consumption is defined as the product of ETS, the number of applied nodes, and doubled TDP (assuming two CPUs per node and $TDP = 280$ for used Milan-based CPUs [30]).

Figures 12b and 12c demonstrate both the estimation for application execution time and approximation of cluster utilisation cost for a single time step and mesh size with 21673212 cells. In addition, these figures reveal the comparison between prediction and real measurements. As shown in figures 12b, the shortest execution time is indicated for the 40-node setup considering both prediction outcomes and real measurements. The performed tests reveal that application execution time estimation fits runtime measurements. It should be noted that we observe relatively negligible performance improvements when employing more and more nodes, starting from around the 16-node setup. Conversely, notable improvements in performance are evident for node configurations comprising as many as 16 nodes.

Investigating results of the cluster utilisation costs (see Figure 12c), the lowest costs are observed for node-setups with around 7-9 nodes considering both prediction outcomes and real measurements. For a higher and smaller number of nodes, the costs of cluster utilisation increase significantly. The predicted costs trendline shows similar behaviour to the actual results. However, the prediction error for the cluster utilisation cost seems to be larger compared to CVOPTS and application execution estimations. We underline that this prediction error depends mainly on differences between the predicted and measured time of a single timeStep and then is further expanded by the numbers of nodes/cores and/or TDP parameters.

9. Conclusions

This work investigates the significance of energy efficiency in high-performance computing, emphasising the necessity for sustainable approaches that reduce carbon emissions while ensuring optimal computational performance. The analysis is conducted using the Urban Air Pollution model implemented in OpenFOAM. A test environment comprising 40 nodes, each equipped with two AMD EPYC 7763 processors, was selected for this research. Initially, the study compared execution times, speedups, cluster usage costs, and estimated total energy consumption across various problem sizes. The findings enabled conclusions regarding the impact of different workloads and computational duration on overall energy consumption and performance.

To enhance the comparison of performance and energy efficiency among computing systems, the metric CVOPTS was introduced, representing the average number of grid cells processed per core, while timeStep indicates the average computation time for a single timeStep of the simpleFoam kernel. This metric is essential for assessing and optimising performance in relation to power consumption, thereby facilitating the development of a cost-efficient hardware configuration. Furthermore, CVOPTS served as the foundation for developing a regression-based model designed to forecast the overall CVOPTS trend line pertinent to a specific grid size and varying quantities of nodes. To increase the prediction accuracy, the regression model was

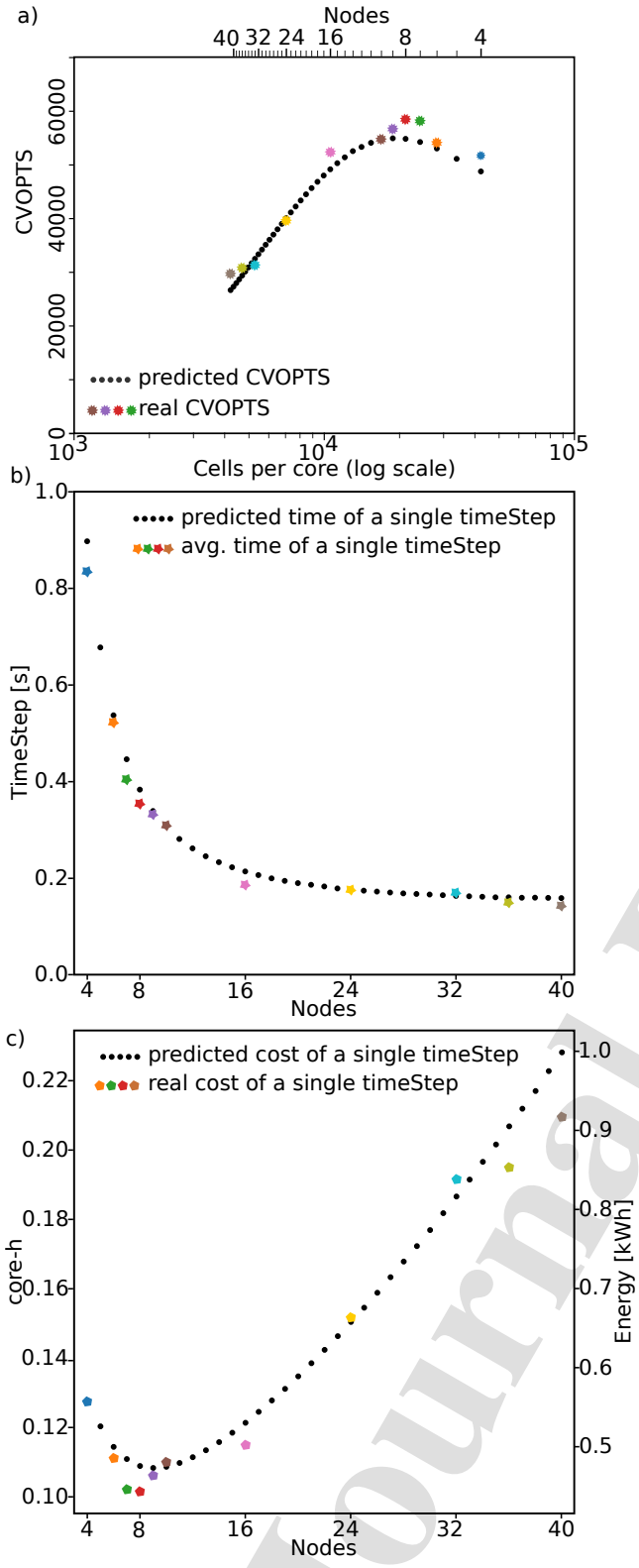

Figure 12: Prediction model examination for the mesh size with 21673212 cells and different numbers of nodes: a) predicted and real CVOPTS comparison, b) application execution time estimations and runtime measurements summary, and c) approximation of the cluster utilisation cost compared to the actual results

extended with an important aspect related to inter-node communication.

In order to validate the assumptions established, the anticipated outcomes are verified through a comparison with the actual measurements obtained during the testing phase. This analysis identified the node configurations exhibiting the highest CVOPTS values, specifically highlighting the top five configurations consisting of 9, 10, 8, 11, or 7 nodes. Nevertheless, it was observed that the CVOPTS values diminish when employing either a greater or fewer number of nodes than those found in the selected top five configurations. On the top of that, utilising the CVOPTS metric, a proposal was made to evaluate the application execution time for a single timeStep (ETS), which subsequently enabled the estimation of the costs associated with utilising the entire cluster.

The analysis of the forecasting outcomes alongside actual measurements indicates that the most economical configurations are those comprising approximately 7 to 9 nodes. In contrast, other configurations of computing nodes exhibit considerably elevated cluster utilisation costs. Furthermore, the forecasting error associated with cluster utilisation costs appears to be more pronounced when compared to the estimates provided by CVOPTS and application execution. This discrepancy is primarily due to the variations between the anticipated and actual duration of a single time step, which is exacerbated by the rising number of nodes/cores and TDP parameters.

Acknowledgements

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking and Poland, Germany, Spain, Hungary, France and Greece under grant agreement number: 101093457. This publication expresses the opinions of the authors and not necessarily those of the EuroHPC JU and Associated Countries which are not responsible for any use of the information contained in this publication.

The authors are grateful to AMD company for granting access to the HPC platform.

References

[1] K. Cameron, R. Ge, X. Feng, High-performance, power-aware distributed computing for scientific applications, Computer 38 (11) (2005) 40–47. doi:10.1109/MC.2005.380.

[2] C. Silva, R. Vilaça, A. Pereira, R. Bessa, A review on the decarbonization of high-performance computing centers, Renewable and Sustainable Energy Reviews 189 (2024) 114019. doi:https://doi.org/10.1016/j.rser.2023.114019.
URL https://www.sciencedirect.com/science/article/pii/S1364032123008778

[3] B. Subramaniam, W.-c. Feng, The green index: A metric for evaluating system-wide energy efficiency in hpc systems, in: 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum, 2012, pp. 1007–1013. doi:10.1109/IPDPSW.2012.123.

[4] Y. Liu, H. Zhu, A survey of the research on power management techniques for high-performance systems, Software: Practice and Experience 40 (11) (2010) 943–964. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.952, doi:https://doi.org/10.1002/spe.952.

URL https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.952

[5] T. Cao, Y. He, M. Kondo, Demand-aware power management for power-constrained hpc systems, in: 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2016, pp. 21–31. doi:10.1109/CCGrid.2016.25.

[6] M. Maiterth, G. Koenig, K. Pedretti, S. Jana, N. Bates, A. Borghesi, D. Montoya, A. Bartolini, M. Puzovic, Energy and power aware job scheduling and resource management: Global survey — initial analysis, in: 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2018, pp. 685–693. doi:10.1109/IPDPSW.2018.00111.

[7] S. Wallace, X. Yang, V. Vishwanath, W. E. Allcock, S. Coghlan, M. E. Papka, Z. Lan, A data driven scheduling approach for power management on hpc systems, in: SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2016, pp. 656–666. doi:10.1109/SC.2016.55.

[8] K. Elangovan, I. Rodero, M. Parashar, F. Guim, I. Hernandez, Adaptive memory power management techniques for hpc workloads, in: 2011 18th International Conference on High Performance Computing, 2011, pp. 1–11. doi:10.1109/HiPC.2011.6152740.

[9] A. Verma, P. Ahuja, A. Neogi, Power-aware dynamic placement of hpc applications, in: Proceedings of the 22nd Annual International Conference on Supercomputing, ICS '08, Association for Computing Machinery, New York, NY, USA, 2008, p. 175–184. doi:10.1145/1375527.1375555.
URL https://doi.org/10.1145/1375527.1375555

[10] J. Koomey, S. Berard, M. Sanchez, H. Wong, Implications of historical trends in the electrical efficiency of computing, IEEE Annals of the History of Computing 33 (3) (2011) 46–54. doi:10.1109/MAHC.2010.28.

[11] A. A. Chandio, K. Bilal, N. Tziritas, Z. Yu, Q. Jiang, S. U. Khan, C.-Z. Xu, A comparative study on resource allocation and energy efficient job scheduling strategies in large-scale parallel computing systems, Cluster Computing 17 (4) (2014) 1349–1367. doi:10.1007/s10586-014-0384-x.
URL https://doi.org/10.1007/s10586-014-0384-x

[12] H. F. Sheikh, H. Tan, I. Ahmad, S. Ranka, P. Bv, Energy- and performance-aware scheduling of tasks on parallel and distributed systems, J. Emerg. Technol. Comput. Syst. 8 (4). doi:10.1145/2367736.2367743.
URL https://doi.org/10.1145/2367736.2367743

[13] G. Houzeaux, R. Badia, R. Borrell, D. Dosimont, J. Ejarque, M. Garcia-Gasulla, V. López, Dynamic resource allocation for efficient parallel cfd simulations, Computers & Fluids 245 (2022) 105577. doi:https://doi.org/10.1016/j.compfluid.2022.105577.
URL https://www.sciencedirect.com/science/article/pii/S0045793022001918

[14] V. Lopez, G. Ramirez Miranda, M. Garcia-Gasulla, Talp: A lightweight tool to unveil parallel efficiency of large-scale executions, in: Proceedings of the 2021 on Performance EngineeRing, Modelling, Analysis, and VisualizatiOn STrategy, PERMAVOST '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 3–10. doi:10.1145/3452412.3462753.
URL https://doi.org/10.1145/3452412.3462753

[15] Badia, Rosa M., Conejero, Javier, Diaz, Carlos, Ejarque, Jorge, Lezzi, Daniele, Lordan, Francesc, Ramon-Cortes, Cristian, Sirvent, Raul, Comp Superscalar, an interoperable programming framework- doi:10.1016/j.softx.2015.10.004.
URL https://core.ac.uk/download/pdf/82338704.pdf

[16] M. Li, C. Weng, X. Lu, Y. Yin, Q. Deng, A resource scheduling strategy for the cfd application on the grid, in: G. Chen, Y. Pan, M. Guo, J. Lu (Eds.), Parallel and Distributed Processing and Applications - ISPA 2005 Workshops, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 305–314.

[17] P. Czarnul, J. Proficz, A. Krzywaniak, Energy-aware high-performance computing: Survey of state-of-the-art tools, techniques, and environments, Scientific Programming 2019 (1) (2019) 8348791. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1155/2019/8348791, doi:https://doi.org/10.1155/2019/8348791.
URL https://onlinelibrary.wiley.com/doi/abs/10.1155/2019/8348791

[18] C.-H. Hsu, J. A. Kuehn, S. W. Poole, Towards efficient supercomputing: searching for the right efficiency metric, in: Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ICPE '12, Association for Computing Machinery, New York, NY, USA, 2012, p. 157–162. doi:10.1145/2188286.2188309.
URL https://doi.org/10.1145/2188286.2188309

[19] P. Michael, Scheduling: Theory, algorithms, and systems, IIE Transactions 28 (8) (1996) 695–697. arXiv:https://doi.org/10.1080/15458830.1996.11770714, doi:10.1080/15458830.1996.11770714.
URL https://doi.org/10.1080/15458830.1996.11770714

[20] M. Drozdowski, M. Lawenda, Scheduling multiple divisible loads in homogeneous star systems, Journal of Scheduling 11 (5) (2008) 347–356. doi:10.1007/s10951-007-0051-7.
URL https://doi.org/10.1007/s10951-007-0051-7

[21] J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt, J. Weglarz, Scheduling computer and manufacturing processes, Journal of the Operational Research Society 48 (6) (1997) 659–659. arXiv:https://doi.org/10.1057/palgrave.jors.2600793, doi:10.1057/palgrave.jors.2600793.
URL https://doi.org/10.1057/palgrave.jors.2600793

[22] HiDALGO2 project website, Accessed: 2024-05-17 (2024).
URL https://www.hidalgo2.eu/

[23] Horváth, Zoltán and Liszkai, Bence and Istenes, György and Zsebők, Péter and Szintai, Balázs and Rácz, Éva V.P. and Környei, László and Harmati, István, Integrated urban air pollution dispersion modelling framework and application in air quality prediction of the city of győr, in: HARMO 2016 - 17th International Conference on Harmonisation within Atmospheric Dispersion Modelling for Regulatory Purposes, Proceedings, Hungarian Meteorological Service, 2016.

[24] L. Kornyei, Z. Horvath, A. Ruopp, A. Kovacs, B. Liszkai, Multi-scale modelling of urban air pollution with coupled weather forecast and traffic simulation on hpc architecture, in: The International Conference on High Performance Computing in Asia-Pacific Region Companion, Association for Computing Machinery, 2021.

[25] G. Alfonsi, Reynolds-Averaged Navier–Stokes Equations for Turbulence Modeling, Applied Mechanics Reviews 62 (4) (2009) 040802. arXiv:https://asmedigitalcollection.asme.org/appliedmechanicsreviews/article-pdf/62/4/040802/5442576/040802_1.pdf, doi:10.1115/1.3124648.
URL https://doi.org/10.1115/1.3124648

[26] M. Leuridan, J. Hawkes, T. Quintino, Polytope: Feature extraction for improved access to petabyte-scale datacubes, EGU23-8839doi:https://doi.org/10.5194/egusphere-egu23-8839.
URL https://meetingorganizer.copernicus.org/EGU23/EGU23-8839.html

[27] COPERT - EU standard vehicle emissions calculator, Accessed: 2024-05-17 (2024).
URL https://copert.emisia.com/

[28] OpenFOAM website, Accessed: 2024-05-17 (2024).
URL https://www.openfoam.com/

[29] F. Pellegrini, J. Roman, A software package for static mapping by dual recursive bipartitioning of process and architecture graphs., in: Proceedings of HPCN'96, Brussels, Belgium. LNCS 1067, pages 493-498, Springer, 1996.
URL https://www.labri.fr/perso/pelegrin/scotch/

[30] AMD Server Processor Specifications, Accessed: 2024-09-02 (2024).
URL https://www.amd.com/en/products/specifications/server-processor.html

[31] AMD Optimizing C/C++ and Fortran Compilers (AOCC), Accessed: 2023-10-22 (2023).
URL https://www.amd.com/en/developer/aocc.html

[32] Szustak, Lukasz and Környei, László and Lawenda, Marcin and Cunha Galeazzo, Flavio Cesar, HiDALGO2 D3.4 Innovative HPC Technologies and Benchmarking (Jul. 2024). doi:10.13140/RG.2.2.25768.79363.
URL https://doi.org/10.13140/RG.2.2.25768.79363

- exploring the importance of performance-energy correlation for CFD codes (OpenFOAM), highlighting the need for sustainable and efficient use of HPC clusters
- CVOPTS (Core VOlume Points per TimeStep) metric is introduced, to enable a direct comparison of the parallel efficiency for applications in modern HPC architectures
- the results confirmed by numerous tests on a 40-node cluster, considering representative grid sizes
- prediction model was derived to determine the optimal number of computational nodes considering both the computational and communication costs of the simulation
- the research reveals the impact of the AMD EPYC architecture on superspeedup, where performance increases superlinearly with the addition of more computational resources

Dr Marcin Lawenda (M) graduated from the Poznań University of Technology and received his M.Sc. in Computer Science (Parallel and Distributed Computation) in 2000. In 2006 he received Ph.D. degree at the same university. He has been working for Poznań Supercomputing and Networking Center for more than 20 years. ML is the project coordinator of HiDALGO2 and the leader of a national and European projects oriented on grid technology and instrumentation (e.g. HiDALGO, AMUNATCOLL, CoeGSS, e-IRGSP5, SGIGrid, RINGrid, DORII, Powiew, PRACE). His research interests include parallel and distributed environments, scheduling and Grid technologies especially in area of applied sciences. He is also author and co-author of reports and papers (100+) in conference proceedings and journals. He has been a member of the Polish Information Processing Society since 2000.

Łukasz Szustak received a D.Sc. degree in Computer Science in 2019 and a Ph.D. granted by the Czestochowa University of Technology in 2012. My main research interests include parallel computing and mapping algorithms onto parallel architectures. My current work is focused on the development of methods for performance portability, scheduling, and load balancing, including the adaptation of parallel applications to modern HPC architectures.

László Környei graduated as a computer scientist and physicist from the University of Szeged and earned his PhD in 2010. He has worked in HPC environments at SZFKI and AUDI Hungária. Currently, he teaches at Széchenyi István University, with research focused on air pollution modeling.

Marcin Lawenda

Conceptualization
Data curation
Formal analysis
Funding acquisition
Investigation
Methodology
Project administration
Software
Supervision
Validation
Writing – original draft
Writing – review & editing

==========

Łukasz Szustak

Conceptualization
Data curation
Formal analysis
Investigation
Methodology
Resources
Software
Supervision
Validation
Visualization
Writing – original draft
Writing – review & editing

==========

László Környei

Conceptualization
Data curation
Formal analysis
Methodology
Software
Validation
Visualization
Writing – original draft

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: