revista principia

ISSN 2447-9187

https://doi.org/10.18265/2447-9187a2025id8918

ORIGINAL ARTICLE

SUBMITTED February 27, 2025

APPROVED July 1, 2025

PUBLISHED ONLINE July 3, 2025

FINAL FORMATTED VERSION April 1, 2026

ASSOCIATE EDITOR
Dr. Francisco Petrônio Alencar de Medeiros

Enhancing the computational power of a general-purpose computer laboratory through a Beowulf cluster using OpenFOAM

Wilson Ramalho Caldeira Filho [1]

Alexandre Ramos Fonseca [2]

Thiago Parente Lima [3] *

[1] wilson.filho@ufvjm.edu.br

[2] arfonseca@ufvjm.edu.br

[3] thiago.parente@ufvjm.edu.br

Instituto de Ciência e Tecnologia, Universidade Federal dos Vales do Jequitinhonha e Mucuri (UFVJM), Diamantina, Minas Gerais, Brazil

* *Corresponding author.*

ABSTRACT: In the context of education and research in Computational Fluid Dynamics (CFD), the high costs associated with software licenses, dedicated hardware, and cloud-based simulation services create significant barriers for smaller universities and emerging research groups. At the same time, general-purpose computer laboratories, often used for undergraduate teaching, tend to be underutilized, leading to inefficient use of institutional resources. This study assesses the viability and performance of a Beowulf-type cluster built with OpenFOAM to simulate a real-world engineering problem: steady-state, external, incompressible airflow over a motorcyclist in a wind tunnel. The cluster was assembled from 60 general-purpose desktop computers located in an undergraduate teaching laboratory. A speedup factor of 55 was achieved by combining the 60 machines, allowing simulations with a mesh of 64 million cells. RAM usage stayed below 1 GB per machine, making it feasible to share the infrastructure between research and regular classroom activities. The findings indicate that the implementation of a Beowulf cluster in such an environment can significantly increase computational capacity while reducing idle time and maximizing the return on institutional investment in computing resources.

Keywords: Beowulf cluster; CFD; computational infrastructure; OpenFOAM; resource optimization.

Aprimorando o poder computacional de um laboratório de computação de uso geral por meio de um cluster Beowulf utilizando OpenFOAM

RESUMO: *No contexto de educação e pesquisa em Dinâmica de Fluidos Computacional (CFD), os altos custos associados a licenças de software, hardware dedicado e serviços de simulação baseados em nuvem criam barreiras significativas para universidades menores e grupos de pesquisa emergentes.*

Ao mesmo tempo, laboratórios de informática de uso geral, frequentemente usados para ensino de graduação, tendem a ser subutilizados, o que leva ao uso ineficiente de recursos institucionais. Este estudo avalia a viabilidade e o desempenho de um cluster do tipo Beowulf construído com OpenFOAM para simular um problema de engenharia do mundo real: escoamento de ar externo, incompressível e em estado estacionário sobre um motociclista em um túnel de vento. O cluster foi montado a partir de 60 computadores desktop de uso geral localizados em um laboratório de ensino de graduação. Um fator de aceleração de 55 foi alcançado pela combinação das 60 máquinas, permitindo simulações com uma malha de 64 milhões de células. O uso de RAM permaneceu abaixo de 1 GB por máquina, tornando viável o compartilhamento da infraestrutura entre a pesquisa e as atividades regulares em sala de aula. As descobertas indicam que a criação de um cluster Beowulf em tal ambiente pode aumentar significativamente a capacidade computacional, reduzindo o tempo ocioso e maximizando o retorno do investimento institucional em recursos de computação.

Palavras-chave: *CFD; cluster Beowulf; infraestrutura computacional; OpenFOAM; otimização de recursos.*

1 Introduction

As research in Computational Fluid Dynamics (CFD) advances, new numerical models have been developed to more accurately represent the physical phenomena involved in various industrial processes. Direct Numerical Simulation (DNS), Large Eddy Simulation (LES), and radiation transfer models are examples of computationally intensive tools available in many general-purpose CFD codes. These tools play a critical role in industries such as oil and gas, aerospace, and power generation, as well as in simulations involving wind loads on buildings, wind distribution in urban agglomerations, and the validation of turbulence models.

Despite their availability, the effective execution of these models requires substantial computational resources that are not always readily accessible. Expenses related to software licensing, high-performance hardware, or cloud-based simulation hours create substantial barriers, especially for smaller research groups. Limited access to computational power can entrench these groups in a cycle of underperformance, limiting their ability to conduct impactful research and thus lowering their chances of securing funding to upgrade their infrastructure. Even when cloud-based computing is accessible, the costs remain prohibitive for many small institutions.

In complex simulations or product development stages, a significant amount of time is allocated to initial runs before establishing a final setup suitable for reporting. These early simulations also demand considerable computational power and should be included in the planning of a research project's budget. In this context, combining open-source software with personal computer (PC) clustering presents a cost-effective alternative for accessing high-performance computing resources.

In undergraduate engineering and other STEM programmes, general-purpose computer laboratories are common. These facilities are often underutilized during evenings, weekends, and academic breaks. Even during regular class timetables, underuse is frequently observed (Cornforth; Atkinson; Spennemann, 2006). As consumer electronics have become more affordable, students increasingly rely on their own devices for internet access and coursework,

further decreasing demand for on-campus computing resources. Moreover, the widespread adoption of online teaching during the recent pandemic is expected to continue, potentially leading to even lower usage of campus computing laboratories.

Repurposing these laboratories to serve both student needs and research computing can boost their utility and maximize the university's return on investment in infrastructure.

The pioneering work on clustering PCs was conducted by Sterling *et al*. (1995), who introduced the concept under the project name Beowulf. Since then, these configurations have been known as Beowulf clusters. Dongarra *et al*. (2005) later defined Beowulf-class clusters as those built using mass-market hardware and software components to achieve optimal performance. Following Sterling's work, broader investigations explored the implementation and parallel processing capabilities of such clusters (Sterling *et al*., 1999). Studies assessing the feasibility of using Beowulf clusters for solving engineering problems with in-house codes were also conducted (Còsul *et al*., 2004; McMillan *et al*., 1999; Noack; Jolly, 2000; Sonzogni *et al*., 2002). The problem sizes in these early studies were aligned with the hardware limitations of their time. Today, the same problems can be run on an ordinary laptop.

More recently, educational institutions have adopted Beowulf clusters as a means of repurposing underutilized or obsolete computers. David *et al*. (2019) assembled a cluster using computers scheduled for disposal and used it to teach parallel computing. Fonseca (2022) constructed a cluster with 12 underused lab computers but faced difficulties operating it with more than four nodes. Gomes *et al*. (2023) created a cluster using mini-computers seized by Brazilian customs and repurposed them for use in high-performance computing (HPC) education. Souza Filho, Nascimento, and Barros (2024) developed AutoBeo, a script that reduces cluster setup time by 86%, facilitating broader adoption without requiring dedicated IT staff.

While these studies demonstrate the feasibility of Beowulf clusters, none evaluate their performance when used with CFD software.

Currently, parallel processing capabilities are readily accessible in general-purpose CFD codes, even to inexperienced users. OpenFOAM, an open-source CFD software, is widely adopted in both academic and commercial settings. Its parallel processing functionality is easily explored and can scale to several thousand cores (Bnà *et al*., 2020). Customized versions of OpenFOAM have been used to perform large-scale simulations on meshes containing up to 100 billion cells using supercomputers (Phuc; Chiba; Minami, 2016). In 2019, a Technical Committee was established to address the challenges of massively parallel high-performance computing (HPC) architectures, with a review provided by Bnà *et al*. (2020).

Recent studies evaluating OpenFOAM's performance on Beowulf clusters remain limited, particularly with respect to the number of computers (nodes) used. Wei and Sha (2013) employed a seven-node Beowulf cluster but reported only the performance evaluation results, omitting crucial details such as mesh size. In a more comprehensive study, Keough (2014) evaluated a cluster comprising eight nodes and observed that performance, measured in terms of speedup, increased with the number of nodes, although efficiency declined. The simulations involved meshes with up to 64 million cells.

Govorukhin *et al*. (2020) utilized a three-node cluster to simulate mine ventilation and reported reduced simulation times, but did not provide a detailed performance assessment. Paytan and Martínez (2022) observed low performance when increasing the number of nodes for a hydraulic jump simulation, likely due to a disproportionate node count relative to the mesh size (only 18,000 cells). Manica (2023) employed two out of 23 available nodes in a Beowulf cluster for battery cooling simulations, without evaluating the cluster's performance.

ISSN 2447-9187

Except for Keough (2014), few studies assess the performance or optimal configurations of OpenFOAM on Beowulf clusters, particularly with larger numbers of nodes. While Keough (2014) utilized only eight nodes, the present study extends the analysis to a Beowulf-type cluster comprising up to 60 nodes. As the number of nodes increases, inter-node communication becomes a critical factor, potentially introducing significant performance bottlenecks.

In addition to evaluating scalability and overall cluster viability, this study also examines solver configurations for pressure and velocity using OpenFOAM to simulate steady-state external incompressible airflow over a motorcyclist in a wind tunnel. This computationally demanding scenario highlights the importance of optimizing cluster performance.

The subsequent sections of this paper are structured as follows: Section 2 presents the methodology for evaluating the cluster; Section 3 discusses the performance results; and Section 4 offers recommended parameters for configuring simulations on Beowulf clusters using OpenFOAM.

2 Methods

This section describes the hardware configuration of the cluster, the performance metrics adopted, and the test case used for performance assessment. The cluster is named Bacurau (*Nyctidromus albicollis*), a small nocturnal bird commonly found in rural areas of Brazil.

2.1 The Bacurau cluster

The Bacurau cluster was assembled using 60 desktop computers from a general-purpose undergraduate computer laboratory at the Universidade Federal dos Vales do Jequitinhonha e Mucuri (UFVJM), Brazil. This laboratory is primarily used in classes from various engineering programs (Figure 1). Consequently, each computer hosts a broad range of software to meet both teaching and learning needs, including office suites and CAD, CAE, and CFD applications. These machines are configured with dual operating systems: Windows 10 and Linux.

Figure 1 ▶
Computers in the general purpose laboratory used to build the cluster.
Source: authors' archive

Each computer, hereafter called a cluster node, was equipped with an Intel Core i7-4770 processor featuring four processing units (PUs) running at 3.40 GHz, 8 GB of DDR3 RAM, 1 TB of HDD storage, and a 1 GB/s Ethernet connection. The operating system for the cluster simulations was Ubuntu 18.04 Minimal. Although OpenFOAM can be installed on both Windows and Linux platforms, the minimal Linux distribution was selected to maximize available RAM for simulations. This choice did not introduce any difficulties during the cluster setup.

The Beowulf cluster in this study was configured using a master-slave architecture, in which a central computer (master node) manages distributed processing among the other nodes (Pedras; Horta; Fonseca, 2019). A virtual machine was initially created with the Ubuntu 18.04 Minimal OS, selected for its low resource footprint and compatibility with the necessary cloning tools. On this machine, essential packages were installed, including the SSH server, the OpenMPI library, OpenFOAM, and, most notably, DRBL (Diskless Remote Boot in Linux). DRBL enabled the slave nodes to boot over the network (PXE boot), loading the operating system directly from the master node without requiring local storage. It was also configured to function as a DHCP server and to provide base system images to the client nodes.

The parallelization process was managed through the automated generation of a host file containing the IP addresses of the active nodes and the number of available PUs per node, as required by OpenFOAM. Bash scripts were employed to automate the copying of simulation files, the proportional distribution of OpenFOAM subdomains across nodes, and the execution of parallel processes using the mpirun command. Although the use of such scripts is not essential for basic cluster operation, their role becomes increasingly valuable as the number of nodes increases, helping to ensure consistency, reduce manual workload, and minimize the likelihood of configuration errors[1]. Domain decomposition and parallel execution were performed in accordance with OpenFOAM's recommended practices, with appropriate modifications to the decomposeParDict file to ensure compatibility across the distributed system.

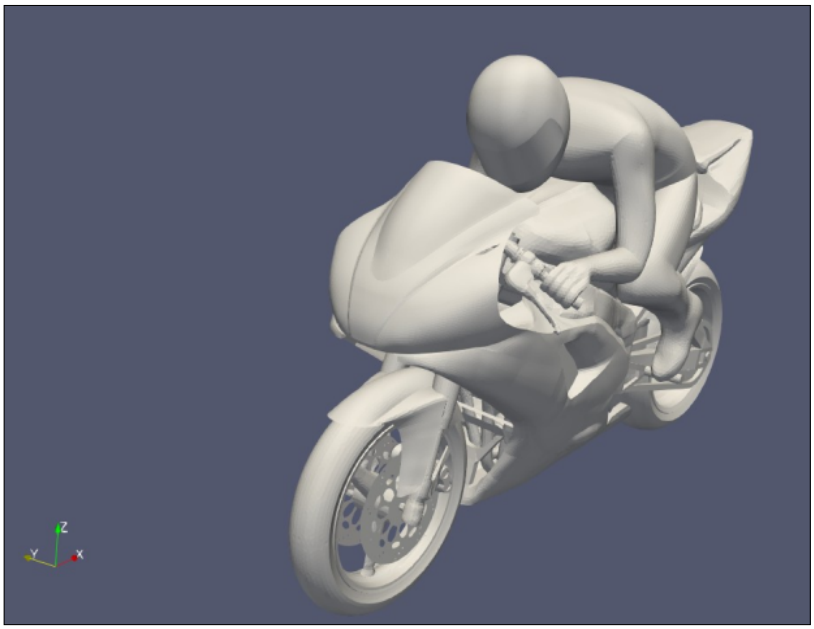
[1] The scripts are available upon request from the corresponding author.

2.2 Test case configuration

The cluster's performance was evaluated using the motorBike tutorial case[2] included with OpenFOAM. This case simulates airflow over a motorcyclist (Figure 2) and solves the Reynolds-averaged Navier-Stokes (RANS) equations for a three-dimensional, turbulent, and incompressible flow with constant physical properties in steady-state conditions. The simulation computes velocity and pressure fields, along with force coefficients. In OpenFOAM version 7, this type of flow is modeled using the simpleFOAM solver. The motorBike tutorial is widely recognized within the OpenFOAM community and serves as a standard, high-complexity engineering simulation. It is particularly suitable for parallel execution and benchmarking cluster performance. In this study, version 7 of OpenFOAM (distributed by the OpenFOAM Foundation) was used. All configuration parameters were maintained in accordance with the original tutorial, except for those specifically modified for this investigation.

[2] Available at: https://github.com/OpenFOAM/OpenFOAM-7/tree/master/tutorials/incompressible/simpleFoam/motorBike.

Figure 2 ▶
Geometry of the motorBike tutorial case.
Source: authors' archive



To assess performance, three mesh sizes were employed: 1, 8, and 64 million cells. These were created by modifying the background mesh, which is generated using OpenFOAM's blockMesh utility. The background mesh dimensions were set to 31×13×13, 70×28×28, and 150×60×60 cells, respectively. After generating the background mesh, the final mesh was constructed using the snappyHexMesh (sHM) utility. All other mesh parameters were maintained constant across tests.

Additionally, the study evaluated the three available domain decomposition methods in OpenFOAM: simple, hierarchical, and scotch. Different solver-preconditioner combinations were also assessed for solving the velocity and pressure systems. Table 1 lists all configurations tested. During pressure solver testing, the velocity solver configuration remained unchanged, and vice versa.

Table 1 ▶
Combinations of preconditioner/solver and solver/smoother assessed.
Source: elaborated by the authors

Pressure (preconditioner/solver)	Velocity (solver/smoother)
diagonal/GAMG	smoothSolver/GaussSeidel
diagonal/PBiCGStab	smoothSolver/DILU
diagonal/PCG	smoothSolver/DILUGaussSeidel
DIC/GAMG	DILU/PBiCG
DIC/PBiCGStab	DILU/PBiCGStab
DIC/PCG	GAMG
FDIC/GAMG	smoothSolver/nonBlockingGaussSeidel
FDIC/PBiCGStab	smoothSolver/symGaussSeidel
GAMG	–

Once the case was configured, the sequence of OpenFOAM applications listed in Table 2 was executed. The table also indicates whether each application was run in serial or parallel mode.

Table 2 ▶
Sequence of applications executed in each test.
Source: elaborated by the authors

Application	Execution
surfaceFeatures	Serial
blockMesh	Serial
decomposePar -copyZero	Serial
snappyHexMesh -overwrite -parallel	Parallel
patchSummary -parallel	Parallel
potentialFoam -parallel	Parallel
simpleFoam -parallel	Parallel

2.2.1 Performance measures

The performance of the cluster was evaluated based on RAM usage, RAM-to-cell ratio and three primary metrics: speedup, efficiency, and total computational time. The total available RAM corresponds to the sum of the RAM from all individual nodes. Likewise, RAM usage refers to the aggregate of peak memory consumed by each node during a simulation run. The RAM-to-cell ratio, calculated as the amount of available or used RAM divided by the number of cells in the mesh, serves as a normalized metric for comparing simulations of different mesh sizes.

The speedup (S) is defined by Equation 1:

$$S(p) = \frac{T(1)}{T(p)} \tag{1}$$

Here, $T(1)$ denotes the total simulation time using one processing unit (PU), and $T(p)$ represents the simulation time using p PUs in parallel. Speedup quantifies the reduction in computational time achieved through parallelization. Ideally, speedup should scale linearly with the number of PUs, doubling the number of PUs would, in an ideal scenario, halve the computational time.

The efficiency e is defined in Equation 2:

$$e = \frac{S(p)}{p} \tag{2}$$

Efficiency indicates how closely the actual speedup approaches the ideal linear behavior. High speedup achieved with a small number of PUs implies high parallelization efficiency, whereas low speedup, with a large number of PUs, indicates poor scalability.

The third performance metric is total computational time, defined as the time required for the simpleFoam application to complete 500 iterations. This iteration count was maintained to ensure consistency with the original tutorial configuration. The time required for mesh generation and pre-processing steps was excluded, as OpenFOAM users often rely on external software for meshing, even when using open-source CFD solutions.

To compute the performance metrics, total computational times for single-core (serial) simulations were needed for the meshes with 1, 8, and 64 million cells. However, due to memory limitations, serial simulations for the 8- and 64-million-cell meshes could

not be executed on a standard cluster node. These simulations were thus performed on a high-performance workstation equipped with an Intel Xeon Gold 6140 processor featuring 72 PUs at 2.30 GHz and 190 GB of RAM.

Since the clock speed of the workstation (2.30 GHz) differs from that of the cluster nodes (3.40 GHz), the measured times from the workstation were adjusted using a calibration equation. To derive this correlation, simulations with meshes of 350,000 and 1 million cells were executed on both machines. The resulting linear regression yields Equation 3:

$$t_{3.4\,GHz} = 0.844\, t_{2.3\,GHz} - 43.138 \quad (3)$$

where $t_{3.4GHz}$ and $t_{2.3GHz}$ are the simulation times (in seconds) on processors with 3.4 GHz and 2.3 GHz clock speeds, respectively.

This equation was then used to calculate the total computational times for serial runs on a 3.4 GHz processor, based on measurements taken from the 2.3 GHz machine. The measured and estimated total times are reported in Table 3. The results derived from Equation 3 should be regarded solely as an estimate of the total computational time for serial runs on the 3.4 GHz processor, since other factors – such as differences in cache size, bus width, and RAM type between the two computers – may also affect execution time. The estimated times (third column) were subsequently used in the calculation of speedup values.

Table 3 ▶
Measured and estimated total computational times for serial simulations.
Source: research data

Mesh size (million cells)	Execution time (s)	
	2.3 GHz processor	3.4 GHz processor
0.350	842	668
1	2897	2402
8	28198	23756 (by Equation 3)
64	266670	225026 (by Equation 3)

3 Results and discussion

This section presents and discusses the results of the cluster assessment, as well as the optimal configuration for the test case. Cluster performance is evaluated in terms of RAM usage, speedup, efficiency, and total computational time. Different decomposition methods and preconditioner/solver combinations were tested with the objective of minimizing computational time.

3.1 RAM usage and RAM-to-cell ratio

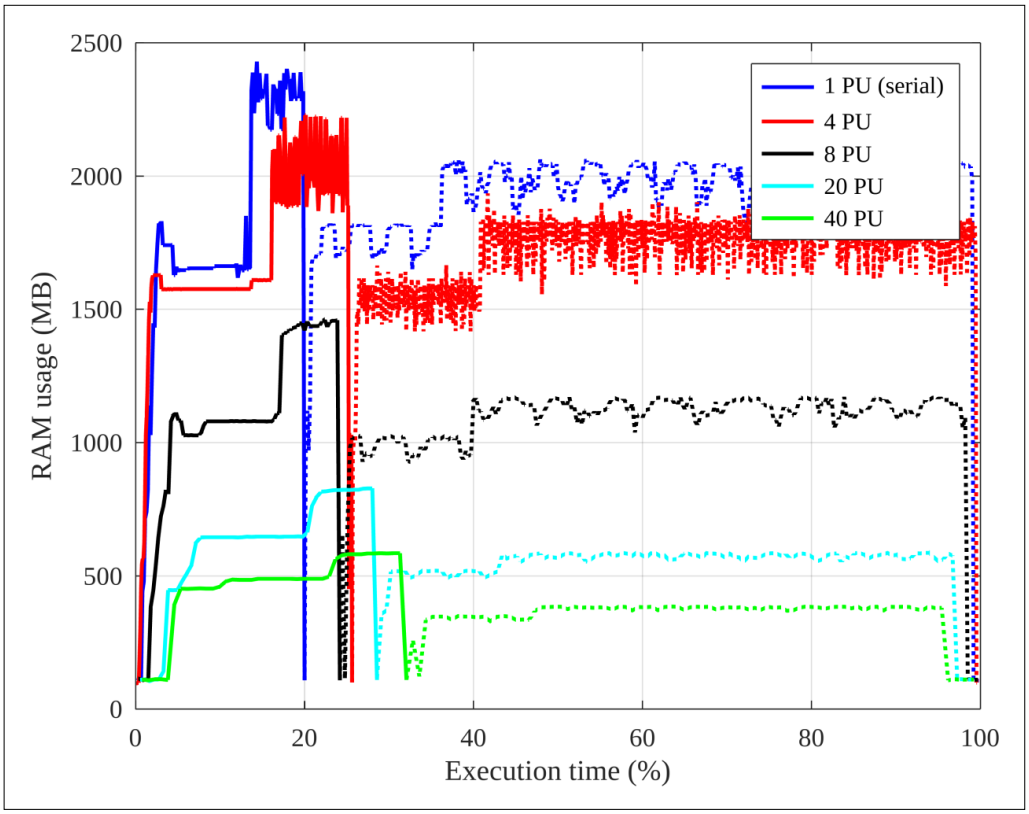
Simulations involving a large number of cells, particularly in external flow problems, typically impose two primary hardware demands: processing capacity and RAM availability. While limited processing power can lead to long and sometimes impractical execution times, insufficient RAM may cause simulations to fail when

memory usage reaches the system limit. The main objective of a Beowulf cluster is to expand both resources via parallel computing.

To assess this capability, a script was developed to monitor RAM usage throughout the execution of OpenFOAM applications.

Figure 3 illustrates RAM usage on a single cluster node during the complete execution of the test case, including mesh generation and solver execution, for a mesh containing 1 million cells. The solid lines represent memory usage by snappyHexMesh (sHM), whereas dashed lines indicate memory consumption by simpleFoam.

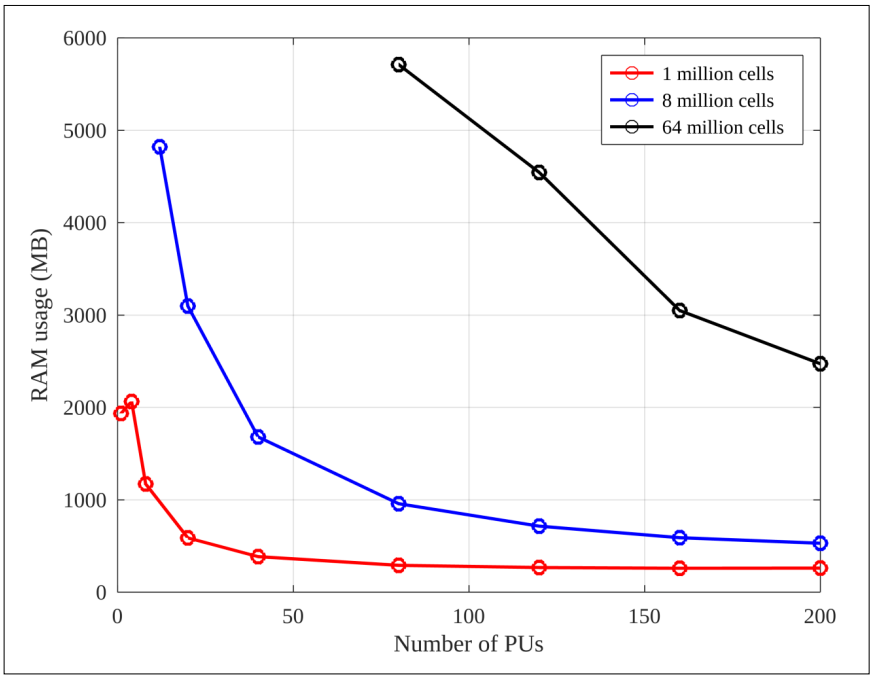
Figure 3 ▶
RAM usage on a single cluster node during full execution of the test case, including mesh generation and solver run: solid lines represent memory usage by sHM and dashed lines indicate memory consumption by simpleFoam.
Source: research data



As the number of nodes increases, peak memory usage per node decreases. This behavior enables the cluster to accommodate simulations with high memory requirements. Notably, peak usage occurs during mesh generation with sHM, reaching levels approximately 15% higher than those during the execution of simpleFoam. It is reasonable to expect that memory consumption increases with the number of mesh cells. This factor must be considered when configuring the cluster for simulations that include both meshing and solving stages.

Figure 4 presents the reduction in RAM usage per node as the number of processing units (PUs) increases, for meshes containing 1, 8, and 64 million cells. This analysis considers only the simpleFoam application.

Figure 4 ▶
Peak RAM usage per cluster node during simpleFoam execution for meshes with 1, 8 and 64 million cells.
Source: research data



As the number of PUs increases, the decreased RAM usage enables the execution of larger meshes. For instance, simulations involving meshes with 8 and 64 million cells were only feasible when 12 and 80 PUs were employed, respectively. These results underscore the extent to which the availability of computational resources can limit research capabilities. Typical meshes for wind tunnel simulations involving vehicle aerodynamics using RANS turbulence models range between 50 and 150 million cells (Amiri *et al*., 2020; Jacuzzi; Granlund, 2019). In turbomachinery simulations using DES models, mesh sizes may reach several hundred million cells (Tyacke *et al*., 2019). Without appropriate computational infrastructure, such studies remain inaccessible to many research groups.

Moreover, Figure 4 shows that RAM usage by each node can be reduced to under 1 GB. This low demand potentially enables the simultaneous use of the lab for both cluster operation and regular teaching activities without mutual interference. Utilizing the cluster during office hours also facilitates maintenance and supervision, eliminating the need for overtime work by students, technicians, or researchers.

Table 4 summarizes RAM usage and the RAM-to-cell ratio during full execution for mesh sizes of 1, 8, and 64 million cells. In the most demanding scenario (64 million cells), RAM usage reached up to 82% of the total available memory. Simulations exceeding this threshold failed during mesh generation stage due to insufficient memory.

Table 4 ▶
RAM availability and usage in full test case execution.
Source: research data

Parameters	**Mesh size (cells)**		
	1 million	**8 million**	**64 million**
Available RAM per node (GB)	8	8	8
Maximum RAM usage per node (GB)	2.23	5.71	6.56
Number of PUs	1	12	80
Number of nodes	1	3	20
$RAM_{available}$ (GB/million cells)	8.0	3.0	2.5
RAM_{used} (GB/million cells)	2.23	2.14	2.05

The values reported in Table 4 correspond to the highest RAM usage per node that still allowed successful execution. Assuming that simulations require at least 82% of total RAM available per node to proceed, a RAM-to-cell ratio of approximately 2.61 GB per million cells per node may be used to estimate the minimum RAM-to-cell ratio available necessary to a successful execution. This estimate supports the planning and sizing of Beowulf clusters for specific simulation workloads similar to steady-state incompressible turbulent flows. Further studies are required to determine whether this reference value is case-sensitive.

3.2 Cluster performance

In a parallel execution using OpenFOAM, the computational mesh is partitioned and distributed across processing units (PUs) through a process known as domain decomposition. OpenFOAM offers three decomposition methods: simple, hierarchical, and scotch. While the simple and hierarchical methods require the user to manually define the number of partitions along the Cartesian axes, the scotch method automatically distributes the mesh to minimize the number of processor boundaries.

Figure 5 presents the total computational time, speedup, and efficiency for the 1-million-cell mesh using each decomposition method. Figures 6 and 7 show corresponding results for the 8-million and 64-million cell meshes, respectively. Across all tested configurations, the scotch algorithm yielded consistently the best performance. Keough (2014) reported differing performance results for decomposition methods when simulating the motorBike case with meshes of 350,000 and 1.8 million cells on a Beowulf cluster with 8 nodes and 128 processing units. The results from Keough (2014) suggest that the performance of decomposition methods is sensitive to mesh topology – a behavior that was not observed in the present study.

Figure 5 ▶
Performance of the cluster for the 1-million-cell mesh using different domain decomposition methods.
Source: research data

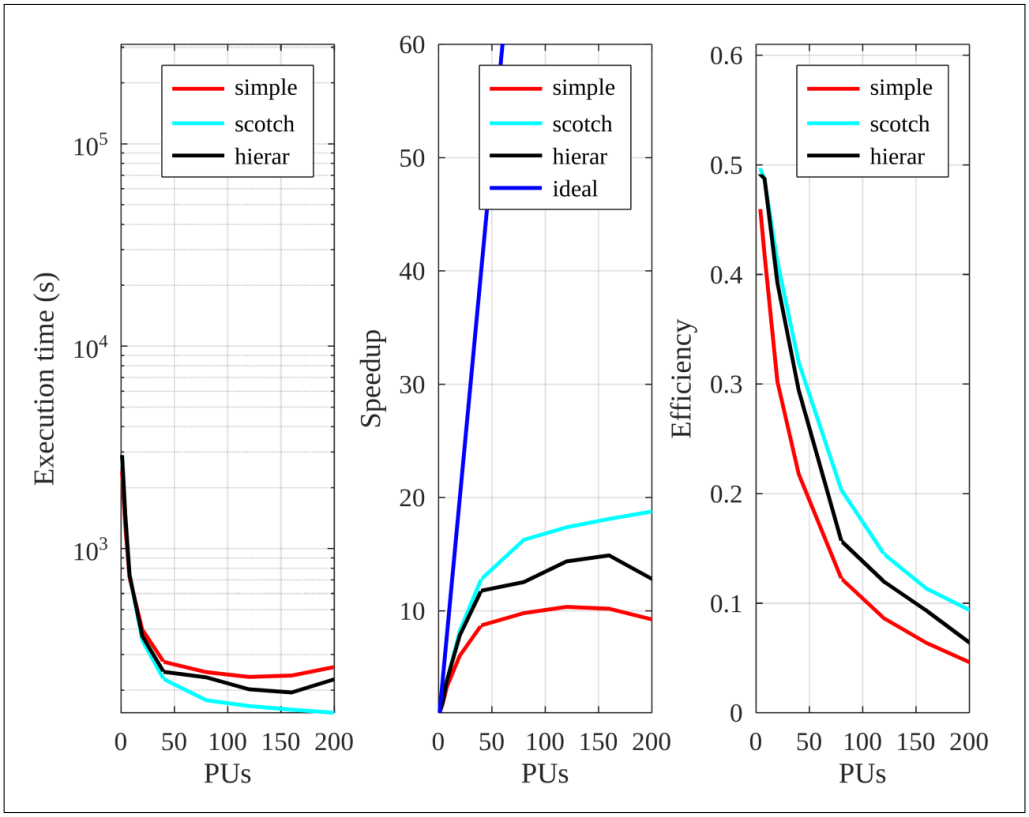


Figure 6 ▶
Performance of the cluster for the 8-million-cell mesh using different domain decomposition methods.
Source: research data

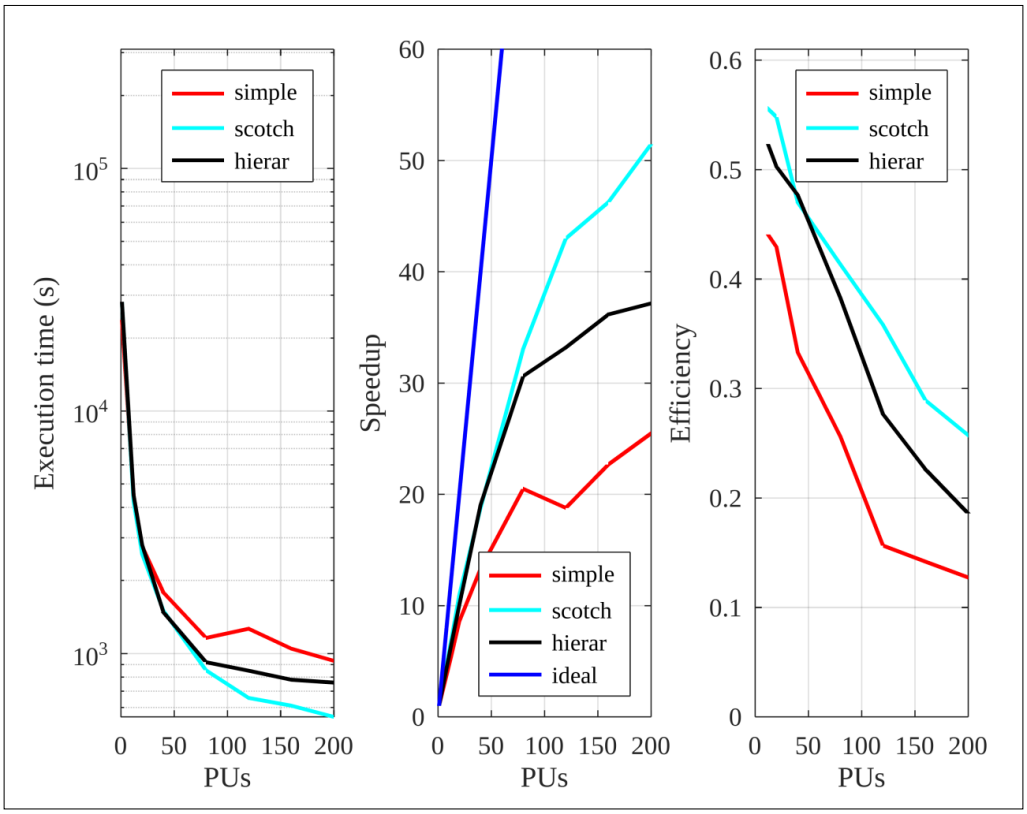
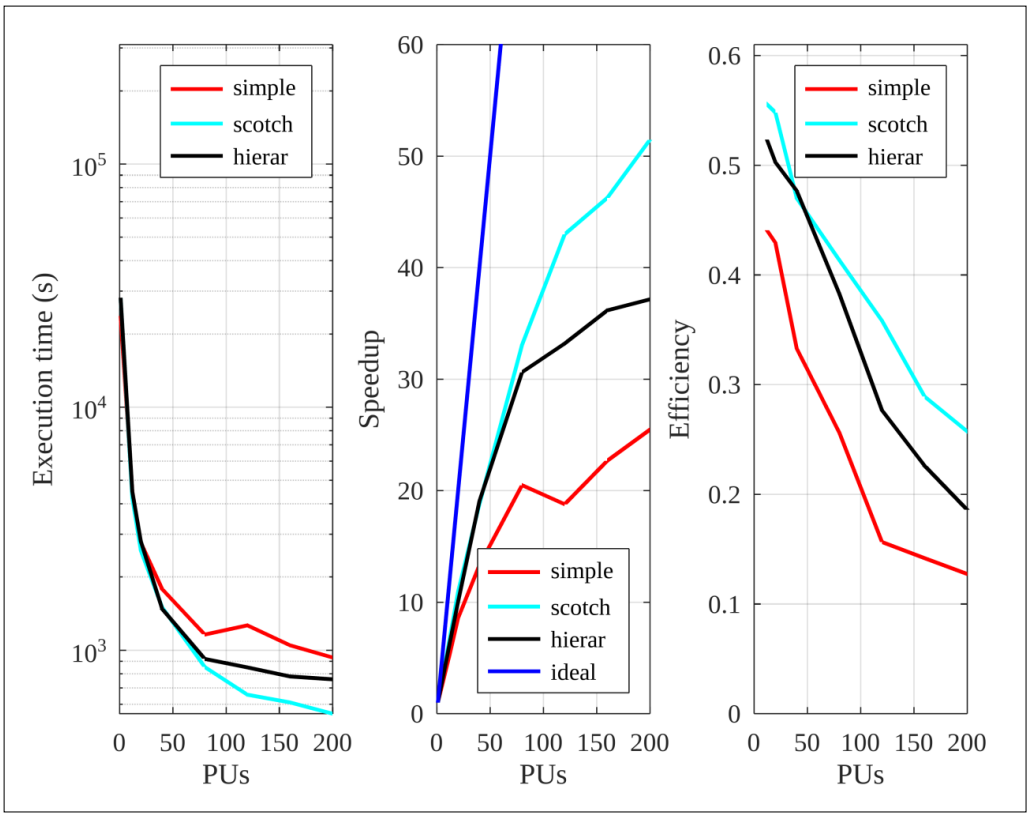


Figure 7 ▶
Performance of the cluster for the 64-million-cell mesh using different domain decomposition methods.
Source: research data



As the number of PUs increases, the total computational time flattens suggesting that the increase in nodes, and consequently in network communication between them, begins to play a significant role in the parallelization process decreasing its

ISSN 2447-9187

performance. A maximum speedup of 55 was achieved by interconnecting 60 computers comprising a total of 240 processing units (PUs) when using the 64-million-cell mesh. By comparison, Keough (2014) reported a speedup of 28 on a Beowulf cluster with 8 nodes and 128 PUs when simulating the motorBike case with a 1.8-million-cell mesh. This value is higher than that obtained in the present study for the 1-million-cell mesh using a similar number of PUs. This result is expected, as Keough (2014) reached that number of PUs using only 8 nodes, whereas the configuration in the present study required 32 nodes, thereby increasing the number of network connections between nodes. For the 1, 8, and 64 million-cell meshes, no significant improvement in performance were observed beyond 40, 120, and 160 PUs, respectively. Still, increasing PU count may be desirable to lower RAM usage per node, thus facilitating shared use of the laboratory space for both teaching and computational research.

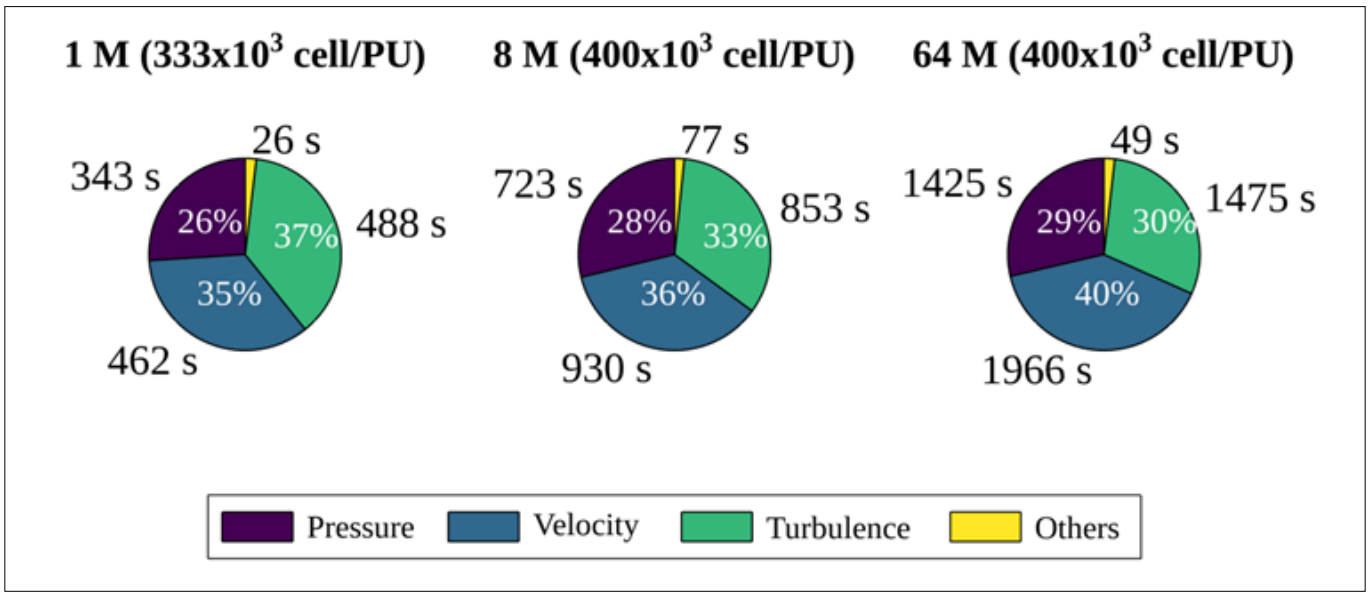
The greatest speedup and efficiency gains were observed for the 64-million-cell mesh (Figure 7). However, all experiments showed a decline in efficiency with increasing PU count. This finding highlights the importance of considering energy consumption and cost when configuring high-performance simulations.

3.3 Evaluation of pressure and velocity solvers

The assessment of pressure and velocity preconditioner/solver configurations was carried out with the goal of reducing total computational time. Initially, the portion of time spent solving the pressure and velocity equation systems was monitored during the execution of simpleFoam. The pressure equations were solved using GAMG, while the velocity equations and turbulence quantities were handled by smoothSolver. Mesh decomposition was performed using the scotch algorithm.

Figure 8 shows the proportion of computational time spent by the pressure and velocity solvers, as well as other tasks, for the three mesh sizes, while maintaining approximately the same cells-to-PU ratio. The time distribution among solvers remains relatively consistent across all mesh sizes, with the velocity and turbulence solvers accounting for the largest share of the computational time.

Figure 8 ▼
Distribution of solver execution time across mesh sizes with constant cells-to-PU.
Source: research data



ISSN 2447-9187

In Figure 9, solver execution times were monitored while keeping the number of processing units (PUs) constant, which led to an increasing cells-to-PU ratio. The results indicate that the pressure solver dominates the total computational time when the cells-to-PU ratio is low. Similarly, Bnà *et al*. (2020) reported that the pressure solver accounted for 67% of the computational time when running the OpenFOAM lid-driven cavity flow tutorial on a mesh with 1 million cells in a supercomputing environment. In both cases, a low cells-to-PU ratio reflects the use of high computational capacity to solve a relatively small problem, where the pressure solver becomes the main bottleneck to reducing execution time. However, as shown in Figure 9, this pattern changes with a higher cells-to-PU ratio, where the velocity solver becomes the primary contributor to the total computational time. This scenario corresponds to cases where limited computational resources are used to solve larger problems, an often-encountered situation in academic research groups. In such contexts, special attention should be given to optimizing the configuration of the velocity solver to improve performance.

Figure 9 ▼
Solver time distribution with increasing cells-to-PU ratio.
Source: research data

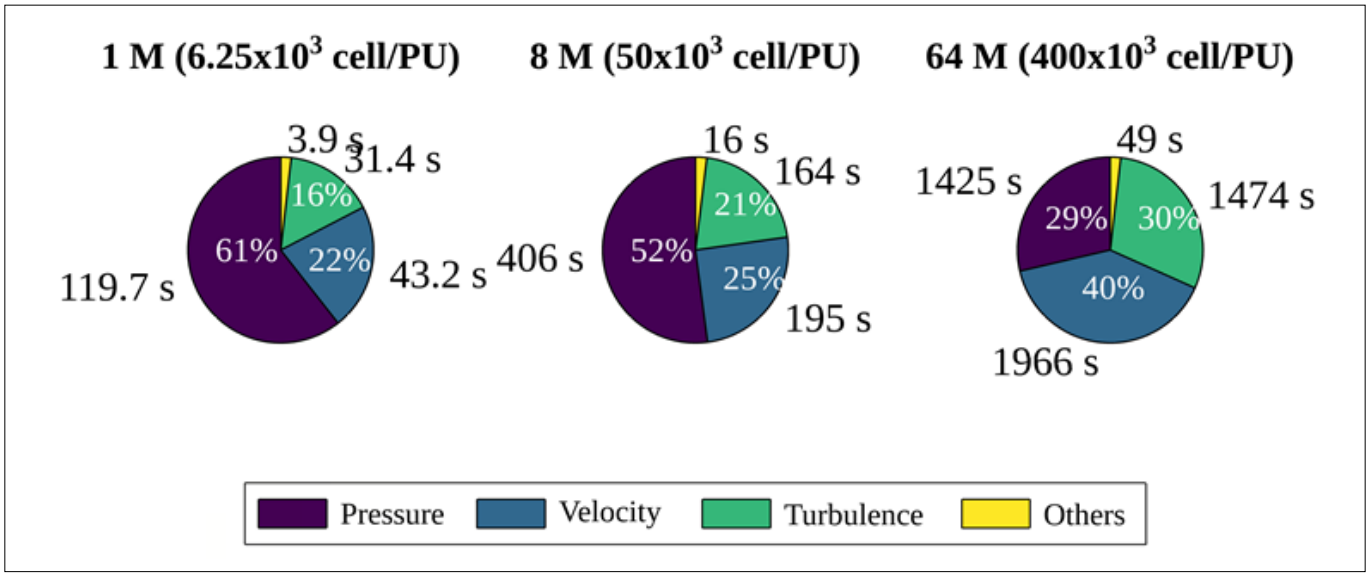
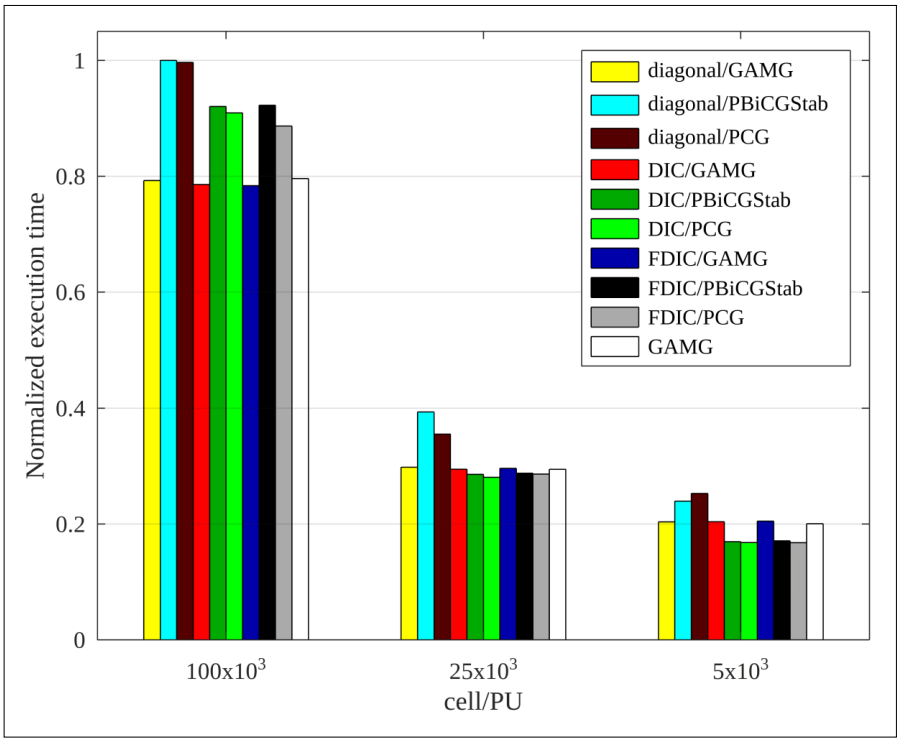


Figure 10 presents an assessment of the preconditioner/solver combinations for pressure that was performed with the aim of reducing overall computational time of the simulation. The assessment was performed in a 1 million cell mesh with a 100, 50 and 25 thousand cells to PU ratio. The total computational time reported in Figure 10 is normalized by the maximum time reached during the tests. The results show different behaviors depending on the cells-to-PU ratio. For a 100 thousand cells-to-PU ratio, the best performance was achieved using the GAMG solver. The different preconditioners used along with the GAMG solver showed only a marginal variation in the total computational time, the best combination being the FDIC/GAMG. A reduction of 22% in the total computational time was achieved when comparing the FDIC/GAMG combination with the diagonal/PbiCGStab which presented the worst performance.

ISSN 2447-9187

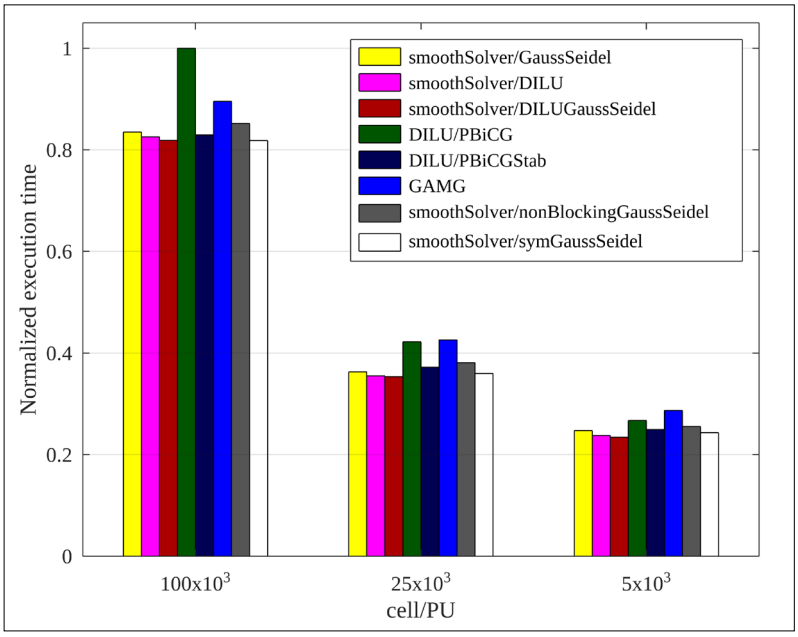
Figure 10 ▶
Normalized total simulation time for various pressure solver and preconditioner combinations.
Source: research data



As the cells-to-PU ratio decreases in Figure 10, the solvers PCG and PBiCGStab, when used with FDIC or DIC preconditioners, tend to outperform GAMG. At a ratio of 5,000 cells per PU, the DIC/PCG combination achieved the best result, reducing total computational time by 4% compared to GAMG, and by 11% when compared to diagonal/PCG. Based on these findings, it is recommended to use FDIC/GAMG for simulations with a cells-to-PU ratio greater than 25,000. For lower ratios, DIC/PCG or FDIC/PCG configurations are more suitable.

As previously shown in Figure 9, the influence of the velocity solvers becomes increasingly relevant as the cells-to-PU ratio increases. Therefore, a similar analysis was conducted to assess their performance. Figure 11 presents the normalized total simulation times for different velocity solvers and preconditioner combinations across three cells-to-PU ratios.

Figure 11 ▶
Normalized total simulation time for various velocity solver and preconditioner combinations.
Source: research data



Unlike the pressure solvers, the velocity solver performance showed little variation across the different ratios. The best result was achieved by the smoothSolver/DILUGaussSeidel combination, which consistently outperformed the others regardless of the ratio. This configuration is therefore recommended for solving the velocity equations under any cells-to-PU condition. Adopting this optimal combination led to an 18% reduction in total computational time.

In absolute terms, the optimal pressure solver configuration saved 2.8 minutes out of a total of 12.8 minutes, while the optimal velocity solver combination saved 1.9 minutes from a 10.4-minute simulation. It is important to note that all test cases were limited to the first 500 iterations, although similar proportional time savings are expected for larger simulations involving comparable physical models.

4 Conclusions

The Beowulf cluster, implemented using general-purpose computer laboratory resources, demonstrated a substantial increase in computational performance. This configuration represents a viable alternative for research groups with limited financial resources or restricted access to high-performance computing facilities. Additionally, it presents an opportunity to reduce the underutilization of such laboratories during idle periods.

RAM usage per node was observed to decrease as the number of processing units (PUs) increased. In certain configurations, memory consumption was as low as 1 GB per node, suggesting that the cluster could potentially operate concurrently with regular classroom activities during standard office hours.

In the simulations conducted, a speedup of 55 was achieved by interconnecting 60 computers comprising a total of 240 PUs. This enabled the execution of simulations involving meshes of up to 64 million cells. Due to the fact that all machines are connected to the same local network, the cluster setup and operation proved to be straightforward and did not require the continuous presence of IT professionals.

The cell to PU ratio must be considered when choosing the best preconditioner/solver to reduce the total computational time. For pressure equations, the FDIC/GAMG configuration is recommended when the cell-to-PU ratio exceeds 25,000. For lower ratios, DIC/PCG or FDIC/PCG combinations are more efficient. Regarding velocity equations, the smoothSolver/DILUGaussSeidel consistently delivered the best performance across all tested ratios.

Funding

This research received no external funding.

Conflict of interest

The authors declare no conflict of interest.

Note

This article is based on an undergraduate thesis in Mechanical Engineering titled "*Avaliação de um cluster Beowulf utilizando OpenFOAM*", authored by Wilson Ramalho Caldeira Filho and conducted at the Universidade Federal dos Vales do Jequitinhonha e Mucuri (UFVJM), originally written in Portuguese.

Contributions to the article

CALDEIRA FILHO, W. R.: conception or design of the study/research; data analysis and/or interpretation. **FONSECA, A. R.:** conception or design of the study/research; final revision with critical and intellectual contribution to the manuscript. **LIMA, T. P.:** conception or design of the study/research; final revision with critical and intellectual contribution to the manuscript. All authors contributed to the writing, discussion, reading, and approval of the final version of the manuscript.

References

AMIRI, H. A.; SHAFAGHT, R.; ALAMIAN, R.; TAHERI, S. M.; SHADLOO, M. S. Study of horizontal axis tidal turbine performance and investigation on the optimum fixed pitch angle using CFD: a case study of Iran. **International Journal of Numerical Methods for Heat & Fluid Flow**, v. 30, n. 1, p. 206-227, 2020. DOI: https://doi.org/10.1108/HFF-05-2019-0447.

BNÀ, S.; SPISSO, I.; OLESEN, M.; ROSSI, G. **PETSc4FOAM**: a library to plug-in PETSc into the OpenFOAM framework. Bruxelles: Partnership for Advanced Computing in Europe, 2020. (PRACE White Paper). Available at: https://prace-ri.eu/wp-content/uploads/WP294-PETSc4FOAM-A-Library-to-plug-in-PETSc-into-the-OpenFOAM-Framework.pdf. Accessed on: 27 Feb. 2025.

CÒNSUL, R.; RODRÍGUEZ, I.; PÉREZ-SEGARRA, C. D.; SORIA, M. Virtual prototyping of storage tanks by means of three-dimensional CFD and heat transfer numerical simulations. **Solar Energy**, v. 77, n. 2, p. 179-191, 2004. DOI: https://doi.org/10.1016/j.solener.2004.04.009.

CORNFORTH, D.; ATKINSON, J.; SPENNEMANN, D. H. R. Configuration and management of a cluster computing facility in undergraduate student computer laboratories. **Campus-Wide Information Systems**, v. 23, n. 1, p. 15-23, 2006. DOI: https://doi.org/10.1108/10650740610639705.

DAVID, N.; ALMEIDA, Y.; SILVA NETO, J.; ALMEIDA, D. D. Cluster Beowulf de baixo custo para o estudo de sistemas distribuídos. *In*: ESCOLA REGIONAL DE INFORMÁTICA DE MATO GROSSO, 10., Cuiabá. **Anais** [...]. Cuiabá: SBC, 2019. p. 136-138. DOI: https://doi.org/10.5753/eri-mt.2019.8612. In Portuguese.

DONGARRA, J.; STERLING, T.; SIMON, H.; STROHMAIER, E. High-performance computing: clusters, constellations, MPPs, and future directions. **Computing in Science & Engineering**, v. 7, n. 2, p. 51-59, 2005. DOI: https://doi.org/10.1109/MCSE.2005.34.

FONSECA, G. E. A. S. **Implementação de um cluster com computadores subutilizados dos laboratórios do Instituto Federal de Goiás – Câmpus Uruaçu**. Undergraduate thesis (Bachelor of Technology in Systems Analysis and Development) – Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Uruaçu, 2022. Available at: http://repositorio.ifg.edu.br/handle/prefix/1710. Accessed on: 18 Jun. 2025. In Portuguese.

GOMES, J. P. T.; SILVA, P. H. B.; INNOCENTINI, M. D. M.; FORMIGONI, C. E. Análise de desempenho de um cluster Beowulf com utilização de TV Box. **Jornada Científica e Tecnológica**, v. 15, n. 3, 2023. Available at: https://josif.ifsuldeminas.edu.br/ojs/index.php/anais/article/view/761. Accessed on: 27 Feb. 2025. In Portuguese.

GOVORUKHIN, Y.; KRIVOLAPOV, V.; PALEEV, D.; PORTOLA, V. Numerical studies of the aerodynamic features of dead-end entries with side junction. **E3S Web of Conferences**, v. 174, 01057, 2020. DOI: https://doi.org/10.1051/e3sconf/202017401057.

JACUZZI, E.; GRANLUND, K. Passive flow control for drag reduction in vehicle platoons. **Journal of Wind Engineering and Industrial Aerodynamics**, v. 189, p. 104-117, 2019. DOI: https://doi.org/10.1016/j.jweia.2019.03.001.

KEOUGH, S. **Optimising the parallelisation of OpenFOAM simulations**. Victoria, AU: Defence Science and Technology Organisation, 2014. Available at: https://apps.dtic.mil/sti/tr/pdf/ADA612337.pdf. Accessed on: 27 Feb. 2025.

MANICA, L. A. **Análise numérica de sistemas de gerenciamento térmico de baterias para veículos elétricos**. 2023. Undergraduate thesis (Bachelor of Automotive Engineering) – Universidade Federal de Santa Catarina, Joinville, 2023. Available at: https://repositorio.ufsc.br/handle/123456789/248347. Accessed on: 18 Jun. 2025. In Portuguese.

MCMILLAN, W.; WOODGATE, M.; RICHARDS, B. E.; GRIBBEN, B. J.; BADCOCK, K. J.; MASSON, C. A.; CANTARITI, F. Demonstration of cluster computing for three-dimensional CFD simulations. **The Aeronautical Journal**, v. 103, n. 1027, p. 443-447, 1999. DOI: https://doi.org/10.1017/S0001924000028037.

NOACK, R.; JOLLY, B. Fully time accurate CFD simulations of JDAM separation from an F-18C aircraft. *In*: AEROSPACE SCIENCES MEETING AND EXHIBIT, 38., 2000, Reno. **Proceedings** […]. Reno: AIAA, 2000. DOI: https://doi.org/10.2514/6.2000-794.

PAYTAN, J. R.; MARTÍNEZ, J. P. V. **Desempeño del sistema cluster para la solución numérica tridimensional del resalto hidráulico**. Undergraduate thesis (Bachelor in Civil Engineering) – Universidad Nacional de Huancavelica, Huancavelica, 2022. Available at: https://hdl.handle.net/20.500.14597/5036. Accessed on: 27 Feb. 2025. In Spanish.

PEDRAS, M. B.; HORTA, E. G.; FONSECA, A. R. RedBlue: um cluster para simulações computacionais. **Revista de Sistemas e Computação**, v. 9, n. 1, p. 75-95, 2019. Available at: https://revistas.unifacs.br/index.php/rsc/article/view/5850. Accessed on: 18 Jun. 2025. In Portuguese.

PHUC, P. V.; CHIBA, S.; MINAMI, K. Large scale transient CFD simulations for buildings using OpenFOAM on a World's top-class supercomputer. *In*: OPENFOAM USER CONFERENCE, 4., 2016, Cologne. **Proceedings** [...]. Cologne: Keysight, 2016.

STERLING, T.; SALMON, J.; BECKER, D. J.; SAVARESE, D. F. **How to build a Beowulf**: a guide to the implementation and application of PC clusters. Cambridge, Massachusetts: MIT Press, 1999.

SONZOGNI, V. E.; YOMMI, A. M.; NIGRO, N. M.; STORTI, M. A. A parallel finite element program on a Beowulf cluster. **Advances in Engineering Software**, v. 33, n. 7-10, p. 427-443, 2002. DOI: https://doi.org/10.1016/S0965-9978(02)00059-5.

SOUZA FILHO, E. L.; NASCIMENTO, V. H.; BARROS, R. C. AutoBeo: uma solução para construção de um cluster Beowulf. **Revista Contemporânea**, v. 4, n. 11, e6520, 2024. DOI: https://doi.org/10.56083/RCV4N11-056. In Portuguese.

STERLING, T.; BECKER, D. J.; SAVARESE, D.; DORBARND, J. E.; RANAWAKE, U. A.; PACKER, C. V. Beowulf: a parallel workstation for scientific computation. *In*: INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, 1995, Pennyilvania. **Proceedings** […]. Boca Raton: CRC Press, 1995. v. 1, p. I-11–I-14. Available at: https://egscbeowulf.er.usgs.gov/geninfo/Beowulf-ICPP95.pdf. Accessed on: 27 Feb. 2025.

TYACKE, J.; VADLAMANI , N. R.; TROJAK, W; WATSON, R.; MA, Y.; TUCKER, P. G. Turbomachinery simulation challenges and the future. **Progress in Aerospace Sciences**, v. 110, 100554, 2019. DOI: https://doi.org/10.1016/j.paerosci.2019.100554.

WEI, Y.; SHA, F. The performance of OpenFOAM in Beowulf clusters. *In*: INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE AND ELECTRONICS ENGINEERING (ICCSEE 2013), 2., 2013, Hangzhou. **Proceedings** […]. Paris: Atlantis Press, 2013. p. 1683-1685. Available at: https://www.atlantis-press.com/article/4852.pdf. Accessed on: 27 Feb. 2025.