Computing and Informatics, Vol. 39, 2020, 695–723, doi: 10.31577/cai_2020_4_695

PERFORMANCE EVALUATION OF PARALLEL HAEMODYNAMIC COMPUTATIONS ON HETEROGENEOUS CLOUDS

Oleg Bystrov, Arnas Kačeniauskas, Ruslan Pacevič
Vadimas Starikovičius, Algirdas Maknickas
Eugeniuš Stupak, Aleksandr Igumenov

Vilnius Gediminas Technical University
Saulėtekio 11, Vilnius 10223, Lithuania
e-mail: {oleg.bystrov, arnas.kaceniauskas, ruslan.pacevic,
vadimas.starikovicius, algirdas.maknickas, eugenius.stupak,
aleksandr.igumenov}@vgtu.lt

Abstract. The article presents performance evaluation of parallel haemodynamic flow computations on heterogeneous resources of the OpenStack cloud infrastructure. The main focus is on the parallel performance analysis, energy consumption and virtualization overhead of the developed software service based on ANSYS Fluent platform which runs on Docker containers of the private university cloud. The haemodynamic aortic valve flow described by incompressible Navier-Stokes equations is considered as a target application of the hosted cloud infrastructure. The parallel performance of the developed software service is assessed measuring the parallel speedup of computations carried out on virtualized heterogeneous resources. The performance measured on Docker containers is compared with that obtained by using the native hardware. The alternative solution algorithms are explored in terms of the parallel performance and power consumption. The investigation of a trade-off between the computing speed and the consumed energy is performed by using Pareto front analysis and a linear scalarization method.

Keywords: Cloud computing, parallel computing, haemodynamic flows, parallel performance analysis, energy consumption, bi-objective optimization problem

Mathematics Subject Classification 2010: 68M14, 68M20, 65Y05

1 INTRODUCTION

In spite of noticeable recent achievements in medicine and technology, cardiovascular diseases are one of the leading causes of death in the world [1]. The heart is a complex system governed by haemodynamics [2], structural dynamics and electromagnetics [3, 4]. The efforts to create fully coupled holistic models of the heart valves have not been applied to a clinical patient-specific base yet. Present-day in-vivo measurement techniques can only resolve large-scale features of the haemodynamic cardiovascular flows [2]. Despite the progress in the numerical methods and constantly increasing power of modern computers, the considered problem is still highly challenging, owing to complex moving geometries, intrinsic flow unsteadiness, very intense velocity gradients, simulation divergence and mesh dependent numerical solutions. Most of the performed haemodynamic analyses have been restricted to non-physiological flow regimes, simplified solution domains, laminar flow simulations and relatively coarse space discretization due to computational challenges [3]. Computational fluid dynamics (CFD) simulations [2, 5] of blood flow in geometries extracted from medical images seem to be well suited for the patient-specific analysis and are compatible with clinical routine [6]. The required level of detail makes the patient-specific haemodynamic simulations of heart chambers computationally very expensive [3]. Naturally, to establish quantitative links between the aortic valve flow patterns and cardiac disease, parallel computations have become an obvious option for significantly increasing computational capabilities. Software for complex biomechanical and haemodynamic computations is usually deployed as an HPC solution [7].

Cloud computing is a distributed computing paradigm that has recently gained great popularity as a platform for on-demand, high-availability and high-scalability access to resources. Generally, clouds provide three levels of services [8]: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). Deployment of the software services (SaaS) for data preparation, high performance computation and visualization on the cloud infrastructure increases the mobility of users and achieves better exploitation because clouds feature flexible management of resources. Thus, flexible cloud infrastructures and software services are perceived as a promising avenue for future advances in the multidisciplinary area of haemodynamic computations, such as numerical analysis of the patient-specific aortic valve flows. However, cloud computing still lacks case studies and quantitative comparison of performance in the case of specific applications. Most of evaluations of virtualization overhead and performance of cloud services have been based on standard benchmarks [8], therefore, the impact of the numerical issues and algorithmic aspects of haemodynamic applications on the performance of parallel computations remains unclear. The growing demand for cloud services and modern computational needs results in the development of large-scale IT infrastructures, which cause a considerable increase in power consumption [9]. Power efficiency is a crucial factor in the cloud computing environment. Green cloud computing is cloud computing with the efficient use of power, which helps to reduce power consumption and carbon

emissions. Energy consumption varies significantly depending on the application, workload, scheduling strategy and virtualization overhead. Power efficiency can be reduced, when virtualization is used, contrary to physical resource deployment, to grant all application requests [10]. In cloud computing, the problem of energy-efficiency is still a challenge mainly because of the variety of applications that need to be processed on cloud infrastructures and the demand for high performance.

The present article describes the efficiency analysis of parallel numerical algorithms for performance- and energy-aware computations of the haemodynamic aortic valve flows carried out on Docker containers of the heterogeneous cloud infrastructure. The aim of the presented research is to demonstrate that efficient computations on heterogeneous clouds require the elaborate selection of numerical solution algorithm and domain decomposition method that are highly dependent on the considered application. Information provided by the synthetic benchmarks usually performed on clouds does not include all important factors and it is not sufficient for finding the best hardware setup. Therefore, the application specific tests need to be performed before the production runs to optimize the parallel and energy efficiency of computationally demanding applications. Other parts of the article are arranged as follows: in Section 2, the related works are overviewed and discussed, Section 3 describes the patient-specific aortic valve problem and Section 4 presents the hosted cloud infrastructure and the developed software services. The parallel performance analysis, energy consumption and solution of a bi-objective optimization problem are discussed in Section 5, while the concluding remarks are presented in Section 6.

2 THE RELATED WORKS

Cloud computing is becoming a natural solution to the problem of expanding computational needs due to its on-demand nature, low-cost and offloaded management [8]. For haemodynamic analysis, cloud computing and Linux containers can offer a convenient, scalable alternative to traditional methods of managing computational resources.

There are different implementations of cloud software that organizations can utilize for deploying their own private cloud. OpenStack is an open source cloud management platform delivering an integrated foundation to create, deploy and scale a secure and reliable public or private cloud [11]. Another popular cloud computing framework, Eucalyptus [12], implements infrastructure services enabling users to run and control virtual machine (VM) instances across a variety of physical resources. Cloud computing makes the extensive use of virtual machines (VM) because they allow workloads to be isolated and the resource usage to be controlled. Xen is primarily a bare-metal, type-1 hypervisor which can be directly installed in the computer hardware without the need for a host operating system [13]. Kernel Virtual Machine (KVM) [13] is a feature of Linux that allows Linux to act as a type 1 hypervisor, running an unmodified guest operating system inside a Linux process. Containers

present an emerging technology for improving the productivity and code portability in the cloud infrastructures. Container-based virtualization was initially viewed as a lightweight alternative to virtual machines. Rather than running a full operating system on virtual hardware, container-based virtualization modifies an existing operating system to provide extra isolation. Container runtimes, such as LXC [14] and Docker [15], largely abstract away the differences between the many operating systems that users run. In many cases, Docker container images require less disk space and I/O than the equivalent VM disk images due to the layered file system. Thus, Docker has emerged as a standard runtime, image format, and build system for Linux containers. HPC vendors have also begun integrating native support for Docker. For example, IBM has added Docker container integration to Platform LSF to run the containers on an HPC cluster [16]. Container computing has revolutionized the way groups are developing, sharing, and running software and services. Recently, container computing has gained traction in the HPC community through enabling technologies like Docker, Charliecloud [17] and Singularity [18]. Container runtimes, such as Singularity and Charliecloud, allow end-users to run containers in environments, where standard Docker tools would not be feasible. Charliecloud [17] employs the Linux user and mount namespaces to run industry-standard Docker containers with no privileged operations or daemons on center resources. Singularity [18] is a novel containerization technology that proposes quickly deployable and transferable containers without encapsulating an entire OS inside the containers images. Sauvanaud et al. [19] have investigated performance of big data applications based on Hadoop, performing scenarios on Singularity and Docker instances. UberCloud application software containers have provided ANSYS Fluids and Structures software [20]. However, it is hardly possible to provide precise guidelines regarding the optimal cloud platform and virtualization technology for each type of research and application [8]. Moreover, the performance is a critical factor in deciding whether or not containers are viable for scientific software.

The performance of virtual machines and lightweight containers has already received some attention in the academic literature because they are crucial components of the overall cloud performance. Seo et al. [21] have compared the performance of containers and virtual machines for non-scientific software stacks deployed in the cloud. In the research performed by Kačeniauskas et al. [22], the performance of the private cloud infrastructure and virtual machines of KVM has been assessed testing CPU, memory, hard disk drive, network and the software services for medical engineering. The measured performance of the virtual resources has been close to the performance of the native hardware measuring only the memory bandwidth and disk I/O. Di Tommaso et al. [23] have compared the performance of some commonly used genetics analysis software running natively and inside a container. Production load for scientific experiments carried out on Docker containers has been investigated by Mazzoni et al. [24]. Estrada et al. [25] have executed genomic workloads on the KVM hypervisor, the Xen para-virtualised hypervisor and LXC containers. Xen and Linux containers exhibited near-zero overhead. In the previous work of the authors [26], the performance of the developed software services for haemodynamic

computations was measured on Xen hardware virtual machines, KVM virtual machines, Docker containers and compared with the performance achieved by using the native hardware. Most of the discussed performance studies [23, 24, 25, 26] have found negligible performance differences between a container and a native hardware.

Han et al. [27] have performed MPI-based NAS benchmarks on Xen and showed that the measured overhead became higher when more cores were added. Strong and weak scalability of the alternative parallel solvers for the aortic valve flows has been examined in Rocks cluster [28], but virtualisation overhead and alternative domain decomposition methods have not been considered. A study [29] on the use of container-based virtualisation in HPC has revealed that Xen VM was slower than LXC container by roughly of the factor of 2, while a native server and LXC container had near-identical performance. Hale et al. [30] have shown that the performance of Docker containers, when using the system MPI library for parallel solution of the Poisson's equation carried out by FEniCS software, was comparable to the native performance. Mohammadi and Bazhirov [31] have performed the High Performance Linpack benchmark on cloud computing infrastructures managed by Amazon Web Services, Microsoft Azure, Rackspace, IBM SoftLayer and demonstrated that the performance per single computing core on public cloud could be comparable to modern traditional supercomputing systems. The authors of the present article have performed the initial MPI-based benchmarks [32] on virtualized resources of homogeneous OpenStack cloud infrastructure.

The pervasive use of cloud computing and the resulting rise in the number of hosting centres have brought forth many concerns including the cost of power, as well as peak power dissipation and cooling. One of the great challenges of cloud infrastructures is to manage system resources in an energy-efficient way. In computer infrastructures, energy-efficiency can be enhanced at three different levels [33], such as energy-efficient applications, energy-efficient hardware and power-aware resource management. Energy-efficient applications are developed using the energy-efficient algorithms [34], special lower level programming techniques, including dynamic voltage and frequency scaling [35], data reuse methodology [36], etc. However, manual tuning of application for higher energy-efficiency remains a time consuming and challenging task. Low-power CPU, memory and other components of energy-efficient hardware [37] can effectively support the static power management. The dynamic power management employs load balancing techniques [38] and power-scalable hardware components [35] to optimize energy consumption. Load balancing methodologies can be characterized as solving a trade-off between power supply and system performance. Tseng and Figueira [39] have investigated power consumption of multithreaded processes on multicore machines and found that energy-optimal configuration was usually the most efficient solution in the case of CPU-bounded tasks. Computations on several multicore nodes, communicating by MPI means, have not been investigated. Pan et al. [40] have investigated power consumption and execution time of applications from NAS parallel benchmark suite on a power-scalable cluster. However, the influence of virtualization layer has not been considered. In virtualized cloud infrastructures, server consolidation and load balancing are some

of those techniques that have gained premier importance for power-aware resource management [41]. Guo et al. [42] proposed heuristic algorithm for dynamic consolidation of heterogeneous VMs based on the analysis of the historical data.

Power models and power management algorithms are necessary for system designers to ensure that an application execution does not exceed the power constraints of a system. Moreover, performance and energy models are required for application developers to optimize time and energy consumption. The performance and energy profiles of real-life scientific applications on modern parallel architectures are not smooth or monotonous and may deviate significantly from the shapes that allowed traditional and state-of-the-art load balancing algorithms to minimize their computation time. Lastovetsky and Manumachu [43] have proposed new model-based methods and algorithms for minimization of time and energy of computations for general shapes of performance and energy profiles of data parallel applications observed on the homogeneous multicore clusters. Zhang et al. [44] introduced an analytical hierarchy process based model to perform the decision-making for virtual machine migration towards green cloud computing. The bi-objective optimization problem for performance and energy for data-parallel applications on homogeneous clusters has been formulated in [45]. Finally, the survey [46] concludes that there exists no predictive model today truly and comprehensively capturing performance and energy consumption of the highly heterogeneous and hierarchical architecture of the modern HPC node. Moreover, computational performance and energy efficiency of any non-trivial application is highly dependable on its specific features and the selection of the best suitable numerical methods [47, 48, 49].

To the best of our knowledge, there are no reports in the literature on attempts to optimize the parallel performance of real-life application on heterogeneous cloud in terms of both consumed time and energy. The novelty of the presented research is the extensive efficiency analysis, which evaluates the parallel speedup on heterogeneous cloud resources, virtualization overhead, a trade-off between the computing speed and the consumed energy, performance of the applied domain decomposition methods and application specific issues of the haemodynamic flows. The presented study supports and advances the idea that time and energy performance models need to be built as discrete functions of problem size, approximating the measured data from application tests on the particular computing architecture. Such discrete functions can be used as input for the performance and energy optimization of the considered application on the particular architecture.

3 A TARGET APPLICATION

The aortic valve has a complex 3D geometry, which is composed of three leaflets and Valsalva sinuses connected together through the commissures. The patient-specific aortic valve geometry was represented by the developed 3D geometric model [50] constructed from the parametric curves according to the obtained patient-specific geometric parameters. The 3D images of the aortic valve of a human subject were

obtained by using the computer tomography equipment GE LightSpeed VCT in the Cardiology and Angiology Centre of Vilnius University Hospital "Santaros Klinikos". The Medical Imaging Interaction Toolkit (MITK) [51] was employed to extract the geometric parameters of the aortic valve from the obtained DICOM images (Figure 1). Finally, the geometric model constructed according to the obtained patient-specific geometric parameters was imported into the ANSYS Workbench for mesh generation.

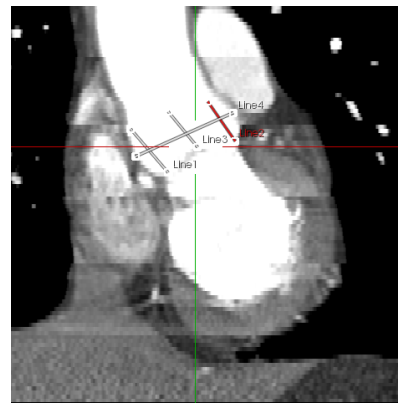
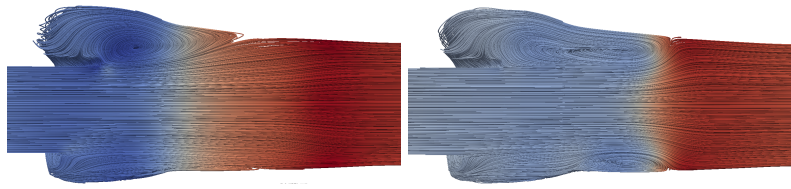


Figure 1. Extraction of the geometric parameters from DICOM images

The pulsatile flow of viscous incompressible fluid is described by the Navier-Stokes equations [52] solved by the finite volume method. The systolic phase of the cardiac cycle was simulated by applying at the inlet a time-dependent velocity of the plug flow based on the clinical Doppler measurements. The measured velocity reached the maximal value of 1.44 m/s during the phase of the peak systole $t = 0.14$ s, while the simulation time interval of 0.36 s was considered. All simulations started from a zero initial condition and the prescribed inflow was accelerated according to the measured waveform. The no-slip boundary conditions were prescribed for velocity on the aorta walls and leaflet surfaces. On the outlet, the prescribed pressure and zero velocity gradient normal to the boundary were applied. The turbulence intensity of 5 % and the hydraulic diameter equal to 0.018 m were specified on the inlet. The density of the blood was set to $\rho = 1\,060\,\mathrm{kg/m^3}$. The dynamic viscosity coefficient was $\mu = 0.004028\,\mathrm{kg/ms}$. Other details of the applied numerical model and references can be found in [53, 54].

Figure 2 shows flow vortices, illustrating the complexity of the 3D flow pattern at $t = 0.1636$ s in the aortic sinuses. Figure 2 a) presents the results obtained by using the $k - \epsilon$ turbulence model [53], while Figure 2 b) shows the data obtained by using the $k - \omega$ turbulence model [53]. The velocity field was visualized by using the streamlines coloured according to the pressure field. The developed vortices are detached and move downstream in the ascending aorta as the flow starts to decelerate after passing the phase of the peak systole. It is worth noting that different



a) $k-\epsilon$ turbulence model b) $k-\omega$ turbulence model

Figure 2. Flow vortices at $t = 0.1636\,\mathrm{s}$ in the aortic sinuses

vertex patterns can be obtained by using different turbulence models. The previous research [54] had shown that the high Reynolds number $k-\epsilon$ turbulence model significantly smoothed the vortex field. Thus, the shear-stress transport (SST) $k-\omega$ model [53] is considered for the developed software service because it is more suitable for the simulation of relatively low Reynolds number turbulent flows past aortic valve. Moreover, it can be concluded that the considered benchmark is rather complex from the numerical point of view.

The way of coupling between the velocity and pressure is an essential part of any numerical scheme for the solution of Navier–Stokes equations [52]. There are two main strategies to perform the velocity-pressure coupling, either a segregated or a coupled approach. In the segregated approach, equations for all variables in the system are decoupled by using the fixed known values from the last iteration of the other independent variables. The linear systems obtained after the discretization can be solved separately for all variables. This approach has the advantage of yielding small storage requirements and systems amenable to solution by classical iterative methods because of the standard structure and properties of system matrices. Unfortunately, suppressed interlinkage between the partial differential equations results in serious drawbacks, such as convergence deterioration and the need for under-relaxation. The drawbacks of segregated schemes and tremendous increase in the available computer memory have stimulated the search for coupled solution algorithms [55]. The idea is to solve discretized momentum and pressure-based continuity equations together in one system of linear equations. Retaining the coupling between the momentum and pressure equations promotes the stability and accelerates the convergence rates. Application of the coupled scheme is advised when the quality of the mesh is poor, non-linear iterations are very expensive due to the time-consuming physical models for constitutive relations, or if larger time steps are required. However, for significantly reduced number of outer iterations of coupled scheme, we pay a price with a solution of four times larger systems of linear equations with non-standard matrices. There is a danger that the advantage of the higher convergence rate will be countered by the increase in computational time incurred in the solution of the enlarged system of equations. Thus, the coupled [55, 5] and PISO [56] schemes were considered as the alternative algorithms to investigate the performance of the developed software service.

Simulations of the turbulent aortic valve flows are time consuming, therefore, parallel computations have become an obvious option for significantly reducing computing time. Domain decomposition approach and message passing technology were used for parallel computations performed by ANSYS Fluent. The default option of MPI library employed by ANSYS Fluent was IBM Platform MPI 9.1.4.2. Communication pattern was highly influenced by the applied domain decomposition method because each process, working on its subdomain, mostly communicates with processes, working on neighbouring subdomains. The most often the domain decompositions were performed by ParMETIS library. However, the alternative Cartesian axes and cylindrical z-coordinate methods were also considered. Solving the largest benchmark problem of 3.2 million cells on 5 nodes (20 cores) and performing the domain decomposition by ParMETIS, 227 MB and 405 MB were transferred during one iteration in the case of the coupled and PISO solution algorithms, respectively. 55 388 MB and 58 320 MB were transferred during the whole test run in the case of the coupled and PISO schemes, respectively. The peak memory consumption of the largest benchmark problem solved on one node reached 11.5 and 9.0 GB, in the case of the coupled and PISO schemes, respectively. In the case of the largest benchmark problem of 3.2 million finite volumes solved on 5 nodes (20 cores), the averaged CPU utilization was equal to 99.84 % and 99.25 % for the coupled and PISO solution algorithms, respectively. Thus, the initial investigation showed that performance of haemodynamic computations might depend on the numerical solution algorithm, the number of used nodes (cores) and the applied domain decomposition method.

4 CLOUD INFRASTRUCTURE AND THE DEVELOPED SOFTWARE SERVICES

The university private cloud infrastructure based on OpenStack Stein version [11] is hosted in Vilnius Gediminas Technical University. The deployed capabilities of the OpenStack cloud infrastructure include Compute Service Nova, Networking Service Neutron, Image Service Glance, Identity Service Keystone, Object Storage Service Swift and Block Storage Service Cinder. The Ubuntu 18.04.3 LTS release was installed in the host nodes. Linux containers were managed with Docker 19.03.2, which created an abstraction layer between computing resources and the services using them. The containers had the following characteristics: 4 cores, 31.2 GB RAM, 80 GB HDD and Ubuntu 18.04.3 LTS release. The cloud infrastructure is composed of several different types of nodes connected to 1 Gbps Ethernet LAN. Hardware characteristics of faster nodes hosting the containers are listed below: Intel®Core i7-6700 3.40 GHz CPU (4 cores), 32 GB DDR4 2 133 MHz RAM and 1 TB HDD. Hardware characteristics of slower nodes are listed below: Intel Core i7-4790 3.60 GHz CPU (4 cores), 32 GB DDR3 1 866 MHz RAM and 1 TB HDD.

The OpenStack cloud IaaS provides the platforms to develop and deploy software services called SaaS (Figure 3). In the developed cloud infrastructure, only

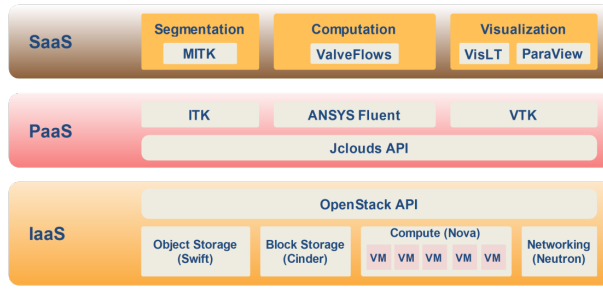


Figure 3. The layers of cloud services

one virtualization layer is used to limit virtualization overhead. In the considered implementation, OpenStack executes and manages Docker containers on bare metal nodes. The cloud infrastructure is managed by using jclouds API [57] providing unified access to EC2 services. The jclouds API binds HTTP/REST services to synchronous and asynchronous Java APIs. Moreover, jclouds API supports 30 cloud providers and cloud software stacks offering high portability. A user-friendly Cloud Manager [26] has been developed by using jclouds. Monitoring of the running instances, volumes and images is available for the user. SaaS jobs can be run by the developed launchers and monitored by the Cloud Manager. The platform as a service was provided on the basis of the popular numerical modelling software ANSYS Fluent [53], which was widely used by researchers and engineers to solve various CFD applications. ITK [58] was deployed on cloud infrastructure as PaaS for developing medical image processing applications. ITK is a cross-platform, open-source application development framework widely used for the development of image segmentation and image registration software. ITK employs leading-edge algorithms for registering and segmenting multidimensional data. A lot of software tools and environments have been developed by using ITK to segment structures in 3D medical images. A Visualization Toolkit (VTK) [59] is deployed as the platform for developing visualization software. The academic numerical software developed by the university researchers usually lacks the required visualization capabilities, therefore, a wide variety of visualization algorithms provided by VTK can fill this gap in the cloud infrastructure. The applications of the discussed toolkits are platform independent, which is very attractive for heterogeneous cloud architectures.

The SaaS layer contains software services developed on top of the provided platforms (Figure 3). Software services were provided for performing medical image segmentation to obtain the patient-specific geometry and to prepare the patient-specific model of the aortic valve. The medical image segmentation was performed and geometric parameters were obtained by using the MITK [51], which was based on the ITK platform, but also used VTK. The software service ValveFlows was developed by using ANSYS Fluent for computations of the patient-specific aortic

valves [26]. Computational results were visualized using the open-source ParaView software [60] and the cloud visualization service VisLT [61] deployed on top of the VTK platform. VisLT was supplemented with the developed middleware component, which could reduce the communication between different parts of the cloud infrastructure.

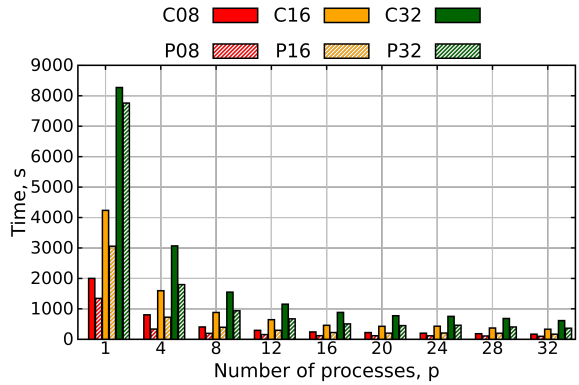
5 THE ANALYSIS OF PARALLEL PERFORMANCE AND ENERGY CONSUMPTION

The presented analysis aims at investigating the parallel performance and energy consumption of the developed software services for haemodynamic flow computations on Docker containers managed by the OpenStack cloud infrastructure. Initially, the considered benchmarks were solved on 5 nodes with 4 cores each, which resulted in homogeneous virtual architecture of 5 Docker containers using 20 cores. 3 slower nodes with 4 cores each were added to perform computational experiments on heterogeneous virtual resources, which resulted in 8 containers using 32 cores. The following subsections present the results of the performed research on virtualization overhead, parallel performance, the influence of the applied domain decomposition methods, power consumption, energy-efficiency and a trade-off between the computing time and the consumed energy.

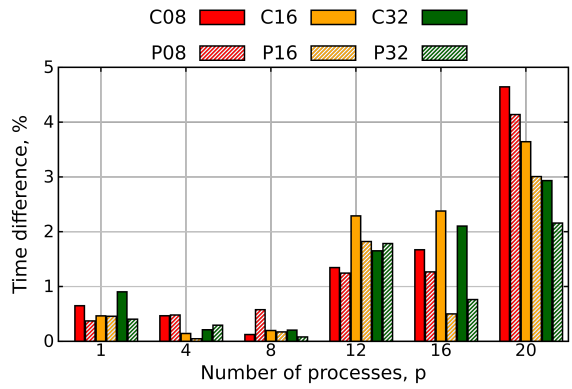
5.1 Execution Time and Virtualization Overhead

First, the computational performance of the considered numerical algorithms and overhead induced by the virtualization were investigated. The benchmark problem was solved on 8 heterogeneous containers (8 nodes, 32 cores) using discrete meshes of the increasing size of 0.8, 1.6 and 3.2 million cells. In Figure 4 a), execution times obtained solving the benchmark on the OpenStack cloud are presented. In accordance with our previous findings [28], the solution times obtained by using the PISO numerical algorithm (P08, P16, P32) were shorter than those attained by using the coupled numerical algorithm (C08, C16, C32).

In the present research, the size of virtualization overhead was also investigated. In Figure 4 b), the percentage difference in performance between the native hardware and the Docker containers is presented. It can be observed that the relative time difference is smaller than 1 % for tests on one and two nodes, i.e. using 1, 4 and 8 processes. These findings are consistent with the results reported in the literature [23, 24, 26] and confirm the low overhead of the employed Docker containers. However, the growth of the virtualization overhead can be observed, when the number of nodes for the solution of the fixed size problem is increased. It is worth noting that the overhead obtained by using the coupled numerical algorithm is consistently bigger. These effects were caused by the increasing part of the communication time in the overall solution time. For the increasing number of parallel processes, the latency of the network communication becomes more and more important, especially,



a) execution time



b) virtualization overhead

Figure 4. The performance of the developed cloud software service

for smaller size problems. On average, the measured latency of the native network was 17.5 μs, while that of the virtual network was 23.0 μs, which revealed 24 % latency increase. The increased latency of a virtual network has also been reported in the literature [22].

5.2 Parallel Performance

At the next stage, the parallel scalability and efficiency of the considered algorithms were studied by performing computations on heterogeneous cloud infrastructure.

In Figure 5, the parallel scalability results obtained on 5 faster homogeneous nodes (20 cores totally) are presented. The parallel scalability is evaluated by using parallel speedup values $S_p = T_1/T_p$, where T_1 and T_p are execution times measured

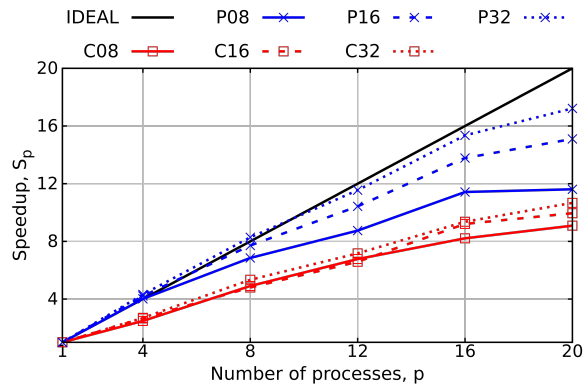
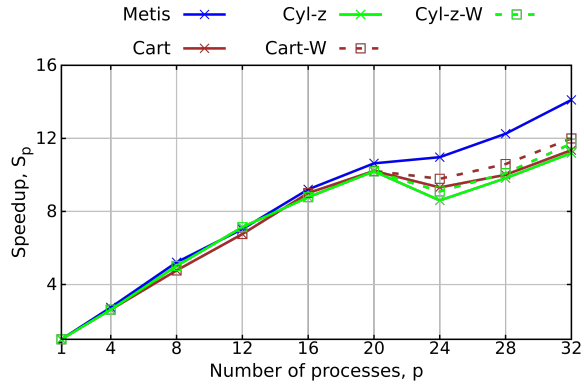


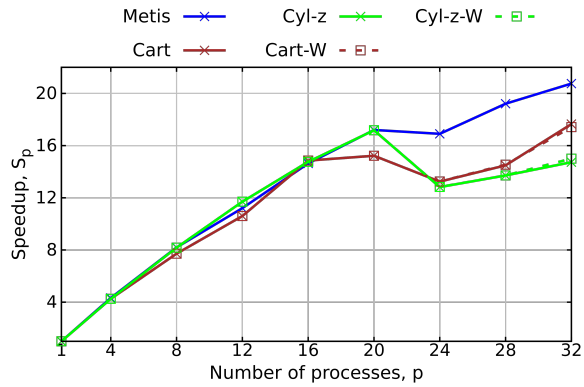
Figure 5. Speedup of parallel computation on homogeneous resources

by solving the benchmark problem with 1 and p processes, respectively. It can be observed that the parallel PISO algorithm demonstrated significantly higher parallel scalability and efficiency than the parallel coupled algorithm. As could be expected, the parallel performance metrics (speedup and efficiency) were greater for a bigger problem. This is in accordance with the theory of parallel algorithms. Noticeably, the difference is more significant for the PISO algorithm, as the performance of the coupled algorithm is bounded by the capacity of the network even for the problems of a larger size. The parallel PISO algorithm shows even super-linear speedup due to the cache effects associated with smaller working data sets for a fixed size problem and the increasing number of the employed cores. It is interesting to note that, in this study, the measured parallel performance metrics are much better than those obtained in the previous research on OpenFOAM-based parallel solvers [47, 62].

In the present research, parallel computation tests were performed using virtualized heterogeneous resources by adding up to 3 slower nodes. The performance of the slower nodes was 12 % and 20 % lower according to Linpack and ValveFlows benchmarks, respectively. Figure 6 presents the parallel speedup values measured using heterogeneous cloud resources (8 nodes, 32 cores). The sequential time measured on one faster core is considered for calculation of speedup values. The largest problem with 3.2 million cells was solved by using the coupled (Figure 6 a)) and PISO (Figure 6 b)) algorithms. It is well-known that the parallel performance of this type of numerical algorithms largely depends on the quality of the domain decomposition. The domain decomposition method should not only ensure the load balance between the parallel processes, but also minimize the amount of communications between the neighbouring regions. In this work, the performance of three domain decomposition methods was investigated. The applied method from the well-known ParMETIS library (the curve "Metis" in Figure 6) is based on the parallel multilevel multiconstraint k-way graph partitioning [63]. The Cartesian axes method (the curve "Cart" in Figure 6) uses the bisection of the domain per-



a) The coupled algorithm



b) The PISO algorithm

Figure 6. Speedup of parallel computation on heterogeneous resources

pendicular to all coordinate axes [53]. The cylindrical z-coordinate method (the curve "Cyl-z" in Figure 6) bisects the domain only along the z cylindrical coordinate [53].

It is worth noting that the performance results of all three domain decomposition methods were quite similar up to 20 cores (5 homogeneous nodes). However, the addition of the first slower node was not beneficial. On the contrary, it caused the decrease in the speedup, i.e. the increase in the solution time, which was significant in most of the cases. This observation once more illustrated the critical dependence of considered parallel application on the network performance. It also should be noted that slower nodes are not only computationally slower, but have 42.5 % higher virtual network latency (40.0 μs instead of 23.0 μs) as well. However, overall performance increase can be achieved by adding 2 or 3 slower nodes. In

this case, the heterogeneous nodes made 37.5 % of all resources, and a considerable increase in speedup values was achieved.

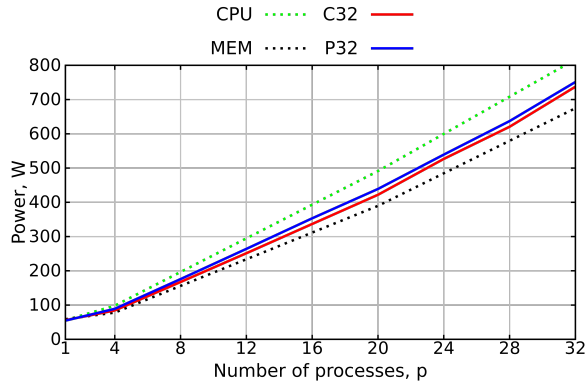
The heterogeneous load was evaluated, assigning the relative weights to the slower cores in the cases of using the Cartesian axes and the cylindrical z-coordinate methods. The parallel performance was improved, but the observed increase in the speedup was not significant (the curves "Cart-W" and "Cyl-z-W" in Figure 6). Parallel computations, using the domain decomposition performed by Metis, revealed higher speedup values. Thus, the most important factor was the ability of the multilevel graph partitioning method to reduce the amount of communication between the neighbouring regions in the case of a larger number of processes, the increased data transfer and higher network latency.

5.3 Power Consumption

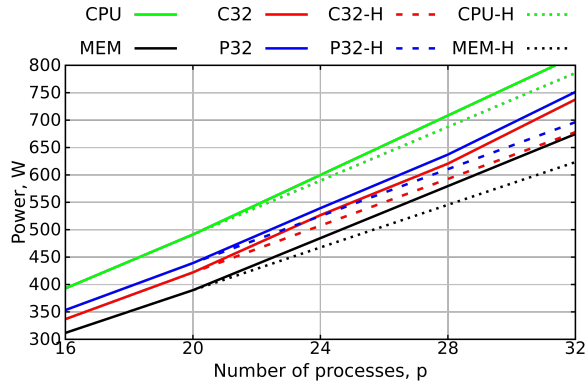
Figure 7 presents the averaged values of the consumed power in watts measured by performing a synthetic CPU benchmark (the curve "CPU"), a synthetic memory benchmark (the curve "MEM"), the computations based on the coupled algorithm (the curve "C32") and the computations based on the PISO algorithm (the curve "P32"). The power was measured by using the EnerGenie energy meter EGM-PWM-LAN. The representative computational experiments were performed 10 times, and the standard deviations were computed. The results obtained by solving the benchmark problem with 3.2 million cells on 8 heterogeneous nodes (32 cores totally) are shown in Figure 7. In agreement with our previous findings, power measurements demonstrated that the PISO algorithm was better in utilizing the CPU cores, i.e., it was more CPU-intensive than the coupled algorithm. Consequently, the software service ValveFlows based on the PISO algorithm consumed more power per fixed time interval, but solved the considered benchmark faster (Figure 4 a)). The power difference became noticeable, when 8 processes (2 nodes) were employed. However, it stabilized at around 20 W on average and did not grow further, increasing the number of the employed nodes, as could be expected from the growing differences in computational performance (Figures 5 and 6).

In Figure 7 b), the effects of heterogeneity are highlighted by showing the extrapolated power consumption levels for homogeneous hardware (the curves "CPU-H", "MEM-H", "C32-H" and "P32-H"). These results show that slower nodes are also less energy-efficient, which is in agreement with the values of thermal design power provided by the CPU manufacturer. All these findings raise the question, whether the observed increase in power consumption for the PISO algorithm and heterogeneous setup is compensated by the obtained computational performance gains, i.e. the reduction in the execution time. Such questions will be addressed in terms of energy-efficiency in the next section.

Another important problem associated with the influence of virtualization on the changes in power consumption was also addressed in the present research. Figure 8 shows the relative difference in power consumption between the native hardware and the OpenStack cloud, running the considered software based on the coupled



a) The global view



b) The zoomed view with highlighted effects of heterogeneity

Figure 7. Power consumption

algorithm. Up to 0.6 % difference could be observed in the results of the performed benchmark. It is worth noting that the differences declined, when the number of employed cores was increased. As can be seen from Figure 7 a), utilization of cores decreased in comparison with the CPU benchmark and, consequently, the influence of the virtualization on power consumption also declined. The standard deviation was equal to 0.11 % and 0.08 % of the presented averaged values in the case of 1 node (4 cores) and 8 nodes (32 cores), respectively. Thus, the standard deviation was smaller than the observed difference in power consumption between the native hardware and OpenStack cloud in all considered cases.

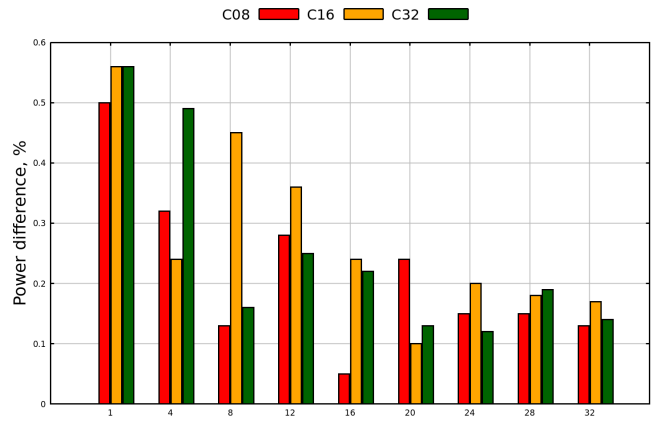


Figure 8. The percentage difference in power consumption between the native hardware and the OpenStack cloud

5.4 Energy-Efficiency

Using the results given in the previous sections, the best solution algorithm and hardware setup in terms of the overall energy consumption can be found. The consumed energy E, required to solve the considered problem, was computed as

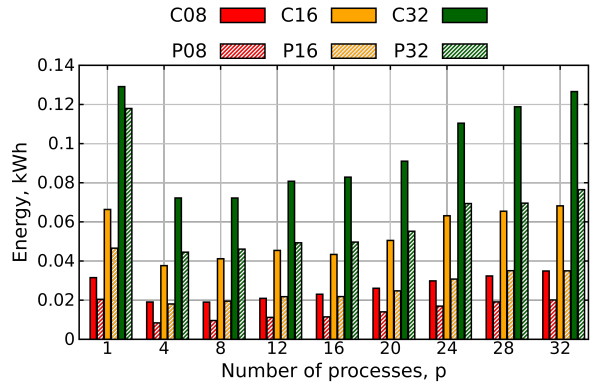
$$E = P \cdot T_{exec} \tag{1}$$

where P is the average power and T_{exec} is the solution time. The consecutive energy change ΔE, increasing the number of the employed cores ($p_i = 1, 4, \cdots, 32$), was calculated by the formula

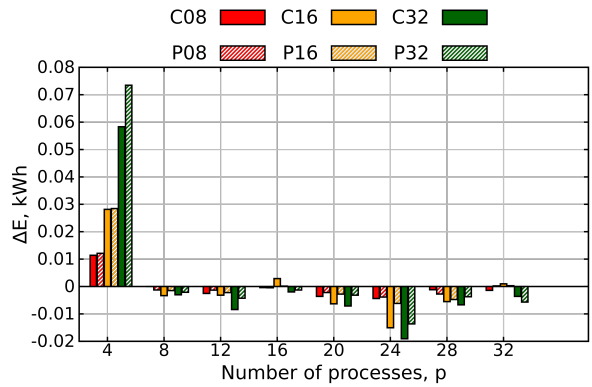
$$\Delta E = E_{p_{i-1}} - E_{p_i} \tag{2}$$

where $E_{p_{i-1}}$ is the energy required for the solution of the considered problem on p_i cores. The values of the consumed energy (1) are shown in Figure 9 a). It can be observed that the parallel PISO algorithm consumed significantly less energy than the parallel coupled algorithm. Thus, the PISO algorithm is superior in terms of computational performance and energy-efficiency because less energy is required in spite of a slightly higher instant power consumption.

As concerns the best hardware setup, the situation is more complicated. Figure 9 b) presents the values of the consecutive energy change calculated by the formula (2). In terms of energy-efficiency, the most significant consecutive change in the consumed energy occurs, when all 4 cores of a single node are used. The observed energy reduction is equal to 45.1 % and 62.3 %, in the case of the largest problem with 3.2 million cells, solved by using the coupled algorithm and the PISO algorithm, respectively. This finding is in agreement with the current trends in the design of modern processors, which is motivated by higher energy-efficiency. Further



a) energy



b) consecutive energy change

Figure 9. Energy consumption

analysis of the presented results reveals a significant jump in energy consumption for the heterogeneous setup with one slower node, which is caused by the above-discussed performance degradation (Figure 6). However, in most of the cases, the increase in energy consumption for 8, 12, 16, and even 20 processes, is not significant. The question arises, whether it is rational to choose the optimal hardware setup as a case of minimal energy consumption, when it is known that the additional nodes cause a significant reduction in the solution time. To answer this question, a bi-objective optimization problem needs to be considered.

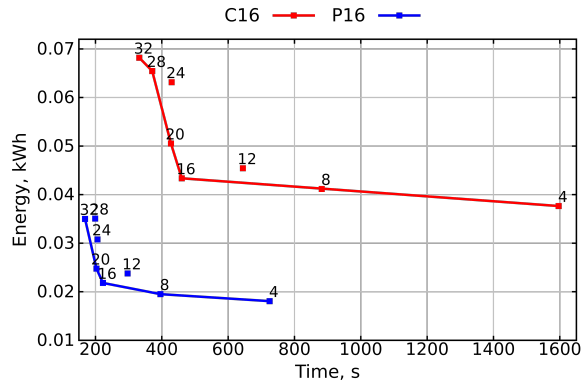
5.5 The Solution of a Bi-Objective Optimization Problem

The choice of the optimal hardware setup needs to be taken in the presence of two conflicting objectives or criteria: the solution time T and the consumed energy E.

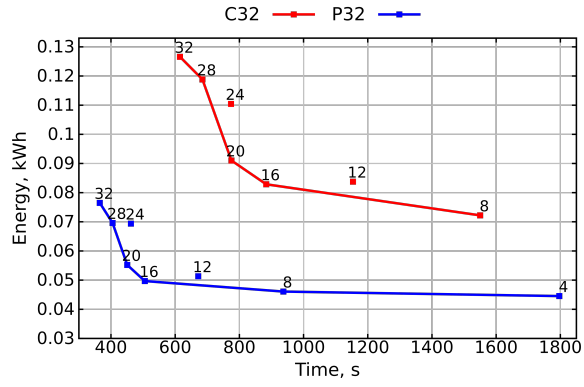
This bi-objective optimization problem can be formulated as follows:

$$\min_{p_i \in X} (T(p_i), E(p_i)) \tag{3}$$

where $X = \{1, 4, 8, 16, 20, 24, 28, 32\}$ is the set of feasible solutions. There are many different approaches to deal with multi-objective optimization problems. For non-trivial problems, no single solution exists, which simultaneously minimizes each objective. A common approach is to find the Pareto optimal solutions, i.e., the solutions that cannot be improved in any of the objectives without degrading at least one of the objectives. The set of the Pareto optimal solutions is often called the Pareto front. For the formulated bi-objective optimization problem (3), the Pareto optimal solutions can be found from the scalar plot shown in Figure 10.



a) the problem with 1.6 million cells



b) the problem with 3.2 million cells

Figure 10. Pareto fronts, considering energy and computing time as objectives

Figure 10 demonstrates a clear domination of the parallel PISO algorithm over the parallel coupled algorithm. The hardware setup with just one slower node ($p_i = 24$) is not the Pareto optimal in all presented cases. This result can be expected from the previous analysis. It is worth noting that the homogeneous setup with three fast nodes ($p_i = 12$) is not the Pareto optimal in all four cases either. This finding can be explained by a relatively less successful domain decomposition produced by Metis for this number of processes. In terms of Pareto optimality, all other hardware configurations cannot be excluded in all the considered cases either. Other approaches, including subjective preferences of a decision maker, should be considered to find a single solution to the formulated problem. Scalarization can be considered as a popular approach to solve a multi-objective optimization problem. The idea is to convert the original problem with multiple objectives to a single-objective optimization problem, which is referred to as a scalarized problem. A proper scalarization method ensures the Pareto optimality of the obtained solutions. In the case of the considered bi-objective optimization problem, linear scalarization can be defined as follows:

$$\min_{p_i \in X} (\omega_T \widehat{T}(p_i) + \omega_E \widehat{E}(p_i)) \tag{4}$$

where ω_T and ω_E are the weights of the normalized solution time objective $\widehat{T}(p_i)$ and the normalized consumed energy objective $\widehat{E}(p_i)$, respectively. The parameters of the scalarization, ω_T and ω_E, are set by a decision maker, but should satisfy simple conditions: $\omega_T + \omega_E = 1$, $1 \geq \omega_T \geq 0$ and $1 \geq \omega_E \geq 0$.

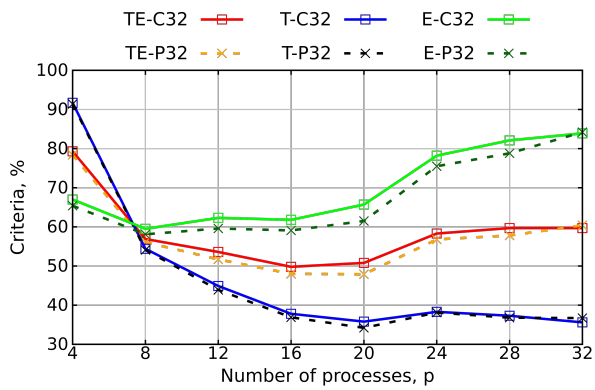


Figure 11. The application of linear scalarization method with various weights

The results of linear scalarization of the considered bi-objective optimization problem (4) are shown in Figure 11. In the case of the largest problem with 3.2 million cells solved by using the coupled algorithm, the curves TE-C32, T-C32 and E-C32 represent the objective functions, with equal, solution time- and energy-oriented weights, respectively. Other curves, TE-P32, T-P32 and E-P32, represent

the objective functions of the same problem solved by the PISO algorithm. Thus, three representative sets of weights were considered in the work. The equal weights ($\omega_T = \omega_E = 0.5$) resulted in optimal configurations based on 16 and 20 cores for the coupled algorithm and the PISO algorithm, respectively. The energy-oriented weights ($\omega_T = 0.2, \omega_E = 0.8$) gave the optimal hardware configuration based on 8 cores for both numerical algorithms. On the contrary, the solution time-oriented weights ($\omega_T = 0.8, \omega_E = 0.2$) revealed the optimal configurations based on 20 homogeneous and 32 heterogeneous cores for the PISO algorithm and the coupled algorithm, respectively. Various optimal hardware configurations obtained for different numerical algorithms proved that the considered bi-objective optimization problem was not trivial even for a simple set of hardware configurations and revealed some challenges for decision-makers.

6 CONCLUSIONS

In this article, parallel performance and energy consumption analysis of the haemodynamic computations performed using Docker containers of the heterogeneous OpenStack cloud infrastructure is presented. Based on the performed investigation, some observations and concluding remarks may be drawn as follows:

- Virtualization layer reduced computational performance of the developed software services by less than 1 % in the case of one or two nodes used. Increasing the number of the employed nodes caused an increase in the virtualization overhead to 4.6 % of the benchmark time on the native hardware due to higher latency of the virtual network.
- The employed virtualization has not significant influence to power consumption of the performed computations. The largest measured difference was less than 0.6 % of the power consumed by running the benchmark on the native hardware.
- The parallel PISO algorithm demonstrated significantly higher computational performance and better parallel scalability than the parallel coupled algorithm for the computations of the considered haemodynamic flows.
- Power measurements demonstrated that the PISO algorithm was more CPU intensive and consumed more power per fixed time interval than the coupled algorithm. However, the consumed energy analysis revealed that the PISO algorithm required less energy to solve the considered problems due to higher computational performance.
- The minimal amount of energy is required to solve the considered problems using a single node with all cores employed. Energy consumption slightly increased with the increasing number of the employed nodes until the degradation of parallel performance became significant. The largest increase in the consumed energy could be observed, when the first slower node was employed.

- Bi-objective optimization based on Pareto front analysis or using the linear scalarization method could help to solve a trade-off between the computing time and the consumed energy.
- The Pareto front analysis helped to detect inefficient hardware configurations. The heterogeneous setup with a single slower node and the homogeneous setup with less successful domain decomposition to 12 parts were not the Pareto optimal in all the considered cases.
- In the case of the considered application, linear scalarization with weights suggested completely different hardware configurations for various subjective preferences of a decision-maker.
- The conducted study demonstrated great challenges to the efficient use of the heterogeneous cloud resources by the developed software service in the case of the considered application. Optimal hardware configurations for single parallel jobs were highly dependent on the network properties, the numerical algorithm and the results of the applied domain decomposition method.
- The performed research has revealed that standard benchmarks can hardly provide comprehensive information required for time- and energy-efficient scheduling of parallel haemodynamic computations. The preliminary specific benchmarks are required to evaluate the parallel performance of the developed software services and algorithmic aspects of the considered application.

REFERENCES

[1] Mendis, S.—Puska, P.—Norrving, B. (Eds.): Global Atlas on Cardiovascular Disease Prevention and Control: Policies, Strategies and Interventions. World Health Organization, World Heart Federation, World Stroke Organization, Geneva, 2011.

[2] Moosavi, M.-H.—Fatouraee, N.—Katoozian, H.—Pashaei, A.—Camara, O.—Frangi, A. F.: Numerical Simulation of Blood Flow in the Left Ventricle and Aortic Sinus Using Magnetic Resonance Imaging and Computational Fluid Dynamics. Computer Methods in Biomechanics and Biomedical Engineering, Vol. 17, 2014, No. 7, pp. 740–749, doi: 10.1080/10255842.2012.715638.

[3] Marom, G.: Numerical Methods for Fluid-Structure Interaction Models of Aortic Valves. Archives of Computational Methods in Engineering, Vol. 22, 2015, No. 4, pp. 595–620, doi: 10.1007/s11831-014-9133-9.

[4] Tumonis, L.—Kačianauskas, R.—Kačeniauskas, A.—Schneider, M.: The Transient Behavior of Rails Used in Electromagnetic Railguns: Numerical Investigations at Constant Loading Velocities. Journal of Vibroengineering, Vol. 9, 2007, No. 3, pp. 15–19.

[5] Kačeniauskas, A.—Rutschmann, P.: Parallel FEM Software for CFD Problems. Informatica, Vol. 15, 2004, No. 3, pp. 363–378, doi: 10.15388/Informatica.2004.066.

[6] Chnafa, C.—Mendez, S.—Nicoud, F.—Moreno, R.—Nottin, S.—Schuster, I.: Image-Based Patient-Specific Simulation: A Computational Modelling of the Human Left Heart Haemodynamics. Computer Methods in Biomechanics and Biomedical Engineering, Vol. 15, 2012, No. 1, pp. 74–75, doi: 10.1080/10255842.2012.713673.

[7] Bastrakov, S.—Meyerov, I.—Gergel, V.—Gonoskov, A.—Gorshkov, A.—Efimenko, E.—Ivanchenko, M.—Kirillin, M.—Malova, A.—Osipov, G.—Petrov, V.—Surmin, I.—Vildemanov, A.: High Performance Computing in Biomedical Applications. Procedia Computer Science, Vol. 18, 2013, pp. 10–19, doi: 10.1016/j.procs.2013.05.164.

[8] Sakellari, G.—Loukas, G.: A Survey of Mathematical Models, Simulation Approaches and Testbeds Used for Research in Cloud Computing. Simulation Modelling Practice and Theory, Vol. 39, 2013, pp. 92–103, doi: 10.1016/j.simpat.2013.04.002.

[9] Sehdev, G. K.—Kumar, A.: Performance Evaluation of Power Aware VM Consolidation Using Live Migration. International Journal of Computer Network and Information Security, Vol. 7, 2015, No. 2, pp. 67–76, doi: 10.5815/ijcnis.2015.02.08.

[10] Zakarya, M.—Gillam, L.: Energy Efficient Computing, Clusters, Grids and Clouds: A Taxonomy and Survey. Sustainable Computing: Informatics and Systems, Vol. 14, 2017, pp. 13–33, doi: 10.1016/j.suscom.2017.03.002.

[11] OpenStack. 2019, available at: `https://www.openstack.org`.

[12] Nurmi, D.—Wolski, R.—Grzegorczyk, C.—Obertelli, G.—Soman, S.—Youseff, L.—Zagorodnov, D.: The Eucalyptus Open-Source Cloud-Computing System. Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09), Shanghai, China, 2009, pp. 124–131, doi: 10.1109/CCGRID.2009.93.

[13] Chierici, A.—Veraldi, R.: A Quantitative Comparison Between XEN and KVM. Journal of Physics: Conference Series, Vol. 219, 2010, No. 4, Art. No. 042005, pp. 1–10, doi: 10.1088/1742-6596/219/4/042005.

[14] LXC. 2019, available at: `https://linuxcontainers.org`.

[15] Docker. 2019, available at: `https://www.docker.com`.

[16] McMillan, B.—Chen, C.: High Performance Docking. Technical Report, 2014.

[17] Priedhorsky, R.—Randles, T.: Charliecloud: Unprivileged Containers for User-Defined Software Stacks in HPC. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17), ACM, 2017, Art. No. 36, pp. 1–10, doi: 10.1145/3126908.3126925.

[18] Kurtzer, G. M.—Sochat, V.—Bauer, M. W.: Singularity: Scientific Containers for Mobility of Compute. PLoS ONE, Vol. 12, 2017, No. 5, Art. No. e0177459, pp. 1–20, doi: 10.1371/journal.pone.0177459.

[19] Sauvanaud, C.—Dholakia, A.—Guitart, J.—Kim, C.—Mayes, P.: Big Data Deployment in Containerized Infrastructures Through the Interconnection of Network Namespaces. Software: Practice and Experience, Vol. 50, 2020, No. 7, pp. 1087–1113, doi: 10.1002/spe.2793.

[20] ANSYS 18.0 Fluids and Structures. UberCloud, 2019, available at: `https://www.theubercloud.com/ansys-cloud`.

[21] Seo, K.-T.—Hwang, H.-S.—Moon, I.-Y.—Kwon, O.-Y.—Kim, B.-J.: Performance Comparison Analysis of Linux Container and Virtual Machine for Building Cloud. Advanced Science and Technology Letters, Vol. 66, 2014, pp. 105–111.

[22] Kačeniauskas, A.—Pacevič, R.—Staškūnienė, M.—Šešok, D.—Rusakevičius, D.—Aidietis, A.—Davidavičius, G.: Private Cloud Infrastructure for Applications of Mechanical and Medical Engineering. Information Technology and Control, Vol. 44, 2015, No. 3, pp. 254–261, doi: 10.5755/j01.itc.44.3.7379.

[23] Di Tommaso, P.—Palumbo, E.—Chatzou, M.—Prieto, P.—Heuer, M. L.—Notredame, C.: The Impact of Docker Containers on the Performance of Genomic Pipelines. PeerJ, Vol. 3, 2015, Art. No. e1273, doi: 10.7717/peerj.1273.

[24] Mazzoni, E.—Arezzini, S.—Boccali, T.—Ciampa, A.—Coscetti, S.—Bonacorsi, D.: Docker Experience at INFN-Pisa Grid Data Center. Journal of Physics: Conference Series, Vol. 664, 2015, No. 2, Art. No. 022029, pp. 22–29, doi: 10.1088/1742-6596/664/2/022029.

[25] Estrada, Z. J.—Deng, F.—Stephens, Z.—Pham, C.—Kalbarczyk, Z.—Iyer, R.: Performance Comparison and Tuning of Virtual Machines for Sequence Alignment Software. Scalable Computing: Practice and Experience, Vol. 16, 2015, No. 1, pp. 71–84, doi: 10.12694/scpe.v16i1.1061.

[26] Kačeniauskas, A.—Pacevič, R.—Starikovičius, V.—Maknickas, A.—Staškūnienė, M.—Davidavičius, G.: Development of Cloud Services for Patient-Specific Simulations of Blood Flows Through Aortic Valves. Advances in Engineering Software, Vol. 103, 2017, pp. 57–64, doi: 10.1016/j.advengsoft.2016.01.013.

[27] Han, J.—Ahn, J.—Kim, C.—Kwon, Y.—Choi, Y.—Huh, J.: The Effect of Multi-Core on HPC Applications in Virtualized Systems. In: Guarracino, M. R. et al. (Eds.): Euro-Par 2010 Parallel Processing Workshops (Euro-Par 2010). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6586, 2011, pp. 615–623, doi: 10.1007/978-3-642-21878-1_76.

[28] Starikovičius, V.—Kačeniauskas, A.—Maknickas, A.—Stupak, E.—Pacevič, R.—Staškūnienė, M.—Davidavičius, G.: On Efficiency of Parallel Solvers for the Blood Flow Through Aortic Valve. Mathematical Modelling and Analysis, Vol. 22, 2017, No. 5, pp. 601–616, doi: 10.3846/13926292.2017.1339642.

[29] Xavier, M. G.—Neves, M. V.—Rossi, F. D.—Ferreto, T. C.—Lange, T.—De Rose, C. A. F.: Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments. 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, IEEE, 2013, pp. 233–240, doi: 10.1109/PDP.2013.41.

[30] Hale, J.—Li, L.—Richardson, C. N.—Wells, G. N.: Containers for Portable, Productive and Performant Scientific Computing. Computing in Science and Engineering, Vol. 19, 2017, No. 6, pp. 40–50, doi: 10.1109/MCSE.2017.2421459.

[31] Mohammadi, M.—Bazhirov, T.: Comparative Benchmarking of Cloud Computing Vendors with High Performance Linpack. Proceedings of the 2nd International Conference on High Performance Compilation, Computing and Communications (HP3C), 2018, pp. 1–5, doi: 10.1145/3195612.3195613.

[32] Staškūnienė, M.—Kačeniauskas, A.—Starikovičius, V.—Maknickas, A.—Stupak, E.—Pacevič, R.: Parallel Simulation of the Aortic Valve Flows on the OpenStack Cloud. Proceedings of the Fifth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering, 2017, Art. No. 16, doi: 10.4203/ccp.111.16.

[33] Valentini, G. L.—Lassonde, W.—Khan, S. U.—Min-Allah, N.—Madani, S. A.—Li, J.—Zhang, L.—Wang, L.—Ghani, N.—Kolodziej, J.—Li, H.—Zomaya, A. Y.—Xu, C.-Z.—Balaji, P.—Vishnu, A.—Pinel, F.—Pecero, J. E.—Kliazovich, D.—Bouvry, P.: An Overview of Energy Efficiency Techniques in Cluster Computing Systems. Cluster Computing, Vol. 16, 2013, No. 1, pp. 3–15, doi: 10.1007/s10586-011-0171-x.

[34] Albers, S.: Energy-Efficient Algorithms. Communications of the ACM, Vol. 53, 2010, No. 5, pp. 86–96, doi: 10.1145/1735223.1735245.

[35] Mishra, A.—Khare, N.: Analysis of DVFS Techniques for Improving the GPU Energy Efficiency. Open Journal of Energy Efficiency, Vol. 4, 2015, No. 4, pp. 77–86, doi: 10.4236/ojee.2015.44009.

[36] Al Hasib, A.—Natvig, L.—Kjeldsberg, P.—Cebrián, J.: Energy Efficiency Effects of Vectorization in Data Reuse Transformations for Many-Core Processors. Journal of Low Power Electronics and Applications, Vol. 7, 2017, No. 1, Art. No. 5, 21 pp., doi: 10.3390/jlpea7010005.

[37] Baun, C.: Performance and Energy-Efficiency Aspects of Clusters of Single Board Computers. International Journal of Distributed and Parallel Systems, Vol. 7, 2016, No. 2-4, pp. 13–22, doi: 10.5121/ijdps.2016.7402.

[38] Pinheiro, E.—Bianchini, R.—Carrera, E. V.—Heath, T.: Dynamic Cluster Reconfiguration for Power and Performance. In: Benini, L., Kandemir, M., Ramanujam, J. (Eds.): Compilers and Operating Systems for Low Power. Springer, Boston, MA, 2003, pp. 75–93, doi: 10.1007/978-1-4419-9292-5_5.

[39] Tseng, C.—Figueira, S.: An Analysis of the Energy Efficiency of Multi-Threading on Multi-Core Machines. International Conference on Green Computing, IEEE, 2010, pp. 283–290, doi: 10.1109/GREENCOMP.2010.5598301.

[40] Pan, F.—Freeh, V. W.—Smith, D. M.: Exploring the Energy-Time Tradeoff in High-Performance Computing. 19th IEEE International Parallel and Distributed Processing Symposium, 2005, pp. 1–9, doi: 10.1109/IPDPS.2005.213.

[41] Choudhary, A.—Rana, S.—Matahai, K. J.: A Critical Analysis of Energy Efficient Virtual Machine Placement Techniques and Its Optimization in a Cloud Computing Environment. Procedia Computer Science, Vol. 78, 2016, pp. 132–138, doi: 10.1016/j.procs.2016.02.022.

[42] Guo, L.—Zhang, Y.—Zhao, S.: Heuristic Algorithms for Energy and Performance Dynamic Optimization in Cloud Computing. Computing and Informatics, Vol. 36, 2017, No. 6, pp. 1335–1360, doi: 10.4149/cai_2017_6_1335.

[43] Lastovetsky, A.—Manumachu, R. R.: New Model-Based Methods and Algorithms for Performance and Energy Optimization of Data Parallel Applications on Homogeneous Multicore Clusters. IEEE Transactions on Parallel and Distributed Systems, Vol. 28, 2017, No. 4, pp. 1119–1133, doi: 10.1109/TPDS.2016.2608824.

[44] Zhang, L.—Ma, J.—Liu, T.—Wang, Y.—Lu, D.: AHP Aided Decision-Making in Virtual Machine Migration for Green Cloud. Computing and Informatics, Vol. 37, 2018, No. 2, pp. 291–310, doi: 10.4149/cai_2018_2_291.

[45] Manumachu, R. R.—Lastovetsky, A.: Bi-Objective Optimization of Data-Parallel Applications on Homogeneous Multicore Clusters for Performance and Energy. IEEE Transactions on Computers, Vol. 67, 2018, No. 2, pp. 160–177, doi: 10.1109/TC.2017.2742513.

[46] O'Brien, K.—Pietri, I.—Reddy, R.—Lastovetsky, A.—Sakellariou, R.: A Survey of Power and Energy Predictive Models in HPC Systems and Applications. ACM Computing Surveys, Vol. 50, 2017, No. 3, Art. No. 37, pp. 1–38, doi: 10.1145/3078811.

[47] Duran, A.—Celebi, M. S.—Piskin, S.—Tuncel, M.: Scalability of OpenFOAM for Bio-Medical Flow Simulations. The Journal of Supercomputing, Vol. 71, 2015, No. 3, pp. 938–951, doi: 10.1007/s11227-014-1344-1.

[48] Kačeniauskas, A.—Kačianauskas, R.—Maknickas, A.—Markauskas, D.: Computation and Visualization of Discrete Particle Systems on gLite-Based Grid. Advances in Engineering Software, Vol. 42, 2011, No. 5, pp. 237–246, doi: 10.1016/j.advengsoft.2011.02.007.

[49] Markauskas, D.—Kačeniauskas, A.: The Comparison of Two Domain Repartitioning Methods Used for Parallel Discrete Element Computations of the Hopper Discharge. Advances in Engineering Software, Vol. 84, 2015, pp. 68–76, doi: 10.1016/j.advengsoft.2014.12.002.

[50] Staškūnienė, M.—Kačeniauskas, A.—Maknickas, A.—Starikovičius, V.—Stupak, E.—Pacevič, R.: Investigation of the Backflows and Outlet Boundary Conditions for Computations of the Patient-Specific Aortic Valve Flows. Technology and Health Care, Vol. 26, 2018, No. S2, pp. 553–563, doi: 10.3233/THC-182502.

[51] Wolf, I.—Nolden, M.—Böttger, T.—Wegner, I.—Schöbinger, M.—Hastenteufel, M.—Heimann, T.—Meinzer, H.-P.—Vetter, M.: The MITK Approach. The Insight Journal – 2005 MICCAI Open-Source Workshop, 2005. Available at: http://hdl.handle.net/1926/14.

[52] Acheson, D. J.: Elementary Fluid Dynamics. Oxford University Press, 1990.

[53] ANSYS: ANSYS Fluent Theory Guide. 2016.

[54] Stupak, E.—Kačianauskas, R.—Kačeniauskas, A.—Starikovičius, V.—Maknickas, A.—Pacevič, R.—Staškūnienė, M.—Davidavičius, G.—Aidietis, A.: The Geometric Model-Based Patient-Specific Simulations of Turbulent Aortic Valve Flows. Archives of Mechanics, Vol. 69, 2017, No. 4-5, pp. 317–345.

[55] Chen, Z. J.—Przekwas, A. J.: A Coupled Pressure-Based Computational Method for Incompressible/Compressible Flows. Journal of Computational Physics, Vol. 229, 2010, No. 24, pp. 9150–9165, doi: 10.1016/j.jcp.2010.08.029.

[56] Issa, R. I.: Solution of the Implicitly Discretised Fluid Flow Equations by Operator-Splitting. Journal of Computational Physics, Vol. 62, 1986, No. 1, pp. 40–65, doi: 10.1016/0021-9991(86)90099-9.

[57] JClouds. 2019, available at: `http://www.jclouds.org`.

[58] Johnson, H.—McCormick, M.—Ibanez, L.: The ITK Software Guide. Insight Software Consortium, 2014, 804 pp.

[59] Schroeder, W.—Martin, K.—Lorensen, B.: The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics. Kitware Inc., 2006, 528 pp.

[60] Kačeniauskas, A.—Pacevič, R.—Bugajev, A.—Katkevičius, T.: Efficient Visualization by Using ParaView Software on BalticGrid. Information Technology and Control, Vol. 39, 2010, No. 2, pp. 108–115.

[61] Pacevič, R.—Kačeniauskas, A.: The Development of VisLT Visualization Service in OpenStack Cloud Infrastructure. Advances in Engineering Software, Vol. 103, 2017, pp. 46–56, doi: 10.1016/j.advengsoft.2016.06.012.

[62] Starikovičius, V.—Čiegis, R.—Bugajev, A.: On Efficiency Analysis of the OpenFOAM-Based Parallel Solver for Simulation of Heat Transfer in and Around the Electrical Power Cables. Informatica, Vol. 27, 2016, No. 1, pp. 161–178, doi: 10.15388/Informatica.2016.80.

[63] Karypis, G.—Kumar, V.: Parallel Multilevel Series k-Way Partitioning Scheme for Irregular Graphs. SIAM Review, Vol. 41, 1999, No. 2, pp. 278–300, doi: 10.1137/S0036144598334138.

Oleg Bystrov is Ph.D. student in informatics engineering at the Vilnius Gediminas Technical University (VGTU), Lithuania, and System Administrator at the Laboratory of Security of Information Technologies. He received his M.Sc. in informatics engineering from VGTU in 2010. His research interests include distributed and cloud computing, green computing, performance evaluation, virtualization technologies, IaaS, OpenStack, Docker, LXC, operating systems and networking.

Arnas Kačeniauskas is the Director of the Institute of Applied Computer Science of Vilnius Gediminas Technical University (VGTU), Lithuania, and the Chief Researcher at the Laboratory of Parallel Computing. He also is Professor and Ph.D. supervisor at the Department of Graphical Systems at VGTU. He received his professorship in informatics engineering and Ph.D. in mechanical engineering at VGTU. He is R & D Project Manager, author and co-author of 34 scientific papers in journals indexed in Clarivate Analytics WoS database. His research interests include parallel, distributed, grid and cloud computing, high-performance computing, performance of SaaS, Linux containers, CFD, haemodynamics, coupled problems in multiphysics, GPGPU.

Ruslan Pacevič is Associated Professor at the Department of Graphical Systems of Vilnius Gediminas Technical University (VGTU), Lithuania. He also is the postdoctoral research fellow at the Department of Applied Informatics of Kaunas University of Technology. He received his Ph.D. in informatics engineering from VGTU in 2015. He is the co-author of several scientific papers and participant of several European and national research projects. His research interests include distributed, grid and cloud computing, development of SaaS and middleware components, OpenStack, Eucalyptus, visualization software, GPGPU, OpenCL.

Vadimas Starikovičius received his Ph.D. degree in mathematics from the Vilnius University, Lithuania in 2002. Currently, he is Professor at the Department of Mathematical Modelling and the Head of Laboratory of Parallel Computing at Vilnius Gediminas Technical University, Lithuania. His current research interests include parallel and distributed computing, performance evaluation, numerical methods for solution of partial differential equations. He has published over 24 refereed articles in these areas.

Algirdas Maknickas is Senior Researcher at the Institute of Mechanics and the Head of the Laboratory of Numerical Simulations of Vilnius Gediminas Technical University (VGTU), Lithuania. He also is Professor and Ph.D. supervisor at the Departments of Biomechanical Engineering and Mechanical and Material Engineering at VGTU. He received his Ph.D. in mechanical engineering at VGTU. He is author and co-author of 23 articles and 16 proceeding papers in journals indexed in Clarivate Analytics WoS database. His research interests include biomechanical engineering, haemodynamics, geometric modelling, applications of high-performance computing, linear and non-linear continuum mechanics, coupled problems in multiphysics, artificial intelligence and computational complexity, GPGPU.

Eugeniuš Stupak is Associated Professor at the Department of Applied Mechanics of Vilnius Gediminas Technical University (VGTU), Lithuania. He received his Ph.D. in Mechanical Engineering from VGTU in 2004. He is the co-author of several scientific papers and participant of several national research projects. His research interests include advanced mesh generation strategies, patient-specific modelling, solution of coupled multiphysics problems.

Aleksandr Igumenov is Lecturer in the Department of Information Technologies at Vilnius Gediminas Technical University, Lithuania. He received his Ph.D. in informatics engineering from the Vilnius University in 2012. He is the author and co-author of several scientific papers. His main research interests include green and energy efficient computing, high-performance computing, IoT and internet technologies, blockchain technologies, global optimization.