Microservice-Enabled Simulation Platform for Industry-4.0 Dynamic Systems: A Computational Fluid Dynamics Case Study for Underground Coalmine Ventilation Networks

Alexey Cheptsov
High Performance Computing Center Stuttgart,
University of Stuttgart
Stuttgart, Germany
e-mail: cheptsov@hlrs.de

Oleg Beljaev
Donetsk National Technical University,
Computer Science Department
Pokrowsk, Ukraine
e-mail: oleg69@ukr.net

Abstract—**Industry 4.0 is a state-of-the-art methodology of complex industrial systems development, which aims to improve the industrial processes by applying the digitalization and computer processing to the technology-level of production objects. The ability to process streaming data from numerous digital sensors, as enabled by Industry 4.0, endorses a potential for creation of many innovative applications. One of the major classes of industrial applications that are enabled by Industry 4.0 is the simulation – the technology that is frequently used for the "offline" optimization of technological processes. In the context of Industry 4.0, the simulation will be benefitting from the "online" integration with the digitalized industrial infrastructure, e.g., for the realization of real-time support scenarios. However, the existing simulation approaches and tools are not able to support highly dynamic, hierarchically organized industrial systems due to a monolithic design of those tools, motivated by the other essential requirements, such as efficiency, user- and developer-friendliness, etc. This article proposes an approach that is based on microservices – the functionally-decoupled, interconnected composition blocks of a hierarchically organized, modular simulation application, that implements a specific part of the simulation logic for the targeted physical phenomena. The use of the microservice-based approach is demonstrated on the implementation of a simulation framework for underground coalmine ventilation networks – complex technological objects that impose challenging tasks, such as conduction of Computational Fluid Dynamics studies.**

Keywords-microservices; dynamic systems; real-time simulation; computational fluid dynamics; cyber-physical automation; ventilation network.

I. INTRODUCTION

Modelling and Simulation are essential technologies for design and support of complex industrial and technological objects, which can be methodologically considered as **Dynamic Systems (DS)**. DS form a special class of simulation applications that are based on the (mathematical) models that are built according to the functional and structural decomposition approaches. The essence of those approaches lies in the application of a numerical decomposition method (e.g., domain decomposition, functional decomposition, structural decomposition, etc.) in order to i) break a complex system down into the elementary sub-systems (sub-models), which can be represented by means of the standard mathematical techniques (e.g., systems of ordinary and/or differential equations), as well as ii) define a methodology to combine those sub-models to solve the original big problem.

A typical DS example is a **Ventilation Network (VN)** of an underground coalmine, as discussed in our earlier publication [1] and also taken as the basis case study in this article (see a further description in Section II). VNs are challenging objects in terms of design, operational support, and control – they expose several safety-critical tasks, which cannot be solved in the practice without an extensive support, provided by the simulation tools. One of such tasks is the analysis of the **aerodynamics**, i.e., of the distribution of the air and gases in underground mining areas – the major factor of the coalmine production safety. The classic simulation approach for such tasks is served by the **Computational Fluid Dynamics (CFD)** technique – a numerical computation method based on complex models.

The quality of the CFD models prediction (the model data accuracy) can be considerably improved by incorporation of the data that are measured at the object (i.e., the physical measurements data) into the simulation – the strategy that is often referred as a part of the "cyber-physical system" or "internet-of-things" concept (see a survey of Bordel Sanchez and Alcarria [2]). Nowadays, these concepts are considered as essentials of the **Industry 4.0** methodology, aiming to maximize the digitalization and automation of industrial systems (see overview of Wollschlaeger et al. [3]). For this purpose, the modern industrial technological objects are being increasingly equipped with "smart sensors" – light-weight measurement devices with rich communication interfaces (like Ethernet) that allow retrieval of physical data, e.g., in form of the real-time data streams. The new-generation smart sensors are characterized by a wide portability and low power consumption and hence can be installed in a very broad spectrum of industrial technological objects, also including the production areas of coalmines and their VNs. The data that are obtained from the sensors can be used not only to "calibrate" the simulation experiments results, but also to support the models in the situations when they cannot be reliably applied to some specific parts of the modelled physical problem or phenomena, e.g., those that include unpredictable processes, such as the gas emission from dynamic sources of VNs, such as the goaf – a cavity in the production areas of the underground coalmining works. In particular, the published work of Stewart et al. [4] has shown that the value of VN models can be considerably increased for the use in real, productional cases by incorporating the sensors data. However, integration of sensor data into the general-purpose simulation packages is not easy, in particular, due to the monolithic design of the former.

This paper evaluates the major problems with which the simulation experts might face during the development of sensor-aware simulation application scenarios by using wide-spread programming models and tools. As an efficient and, at the same time, developer-friendly alternative to the traditional approaches, a decentralized, microservice-oriented simulation framework *MS-Sim* is proposed and described.

The remainder of the paper is organized as follows: Section II gives an overview of VNs and modelling techniques for their analysis. Section III introduces and discusses the methodology of a real-time simulation of VNs. Section IV is dedicated to the microservice architecture for development of simulation applications. Section V presents the evaluation use cases and discusses the validation and benchmarking results. Section VI concludes the paper and discusses the main outcomes.

II. Overview of Ventilation Networks as Objects of Modelling and Simulation

Despite of the revolutionary development in the domain of renewable and regenerative energy technologies, as presented in the last decade, (black) coal remains one of the most important sources of energy for industrial and urban objects world-wide, along with the gas, oil and other fossil fuels. The total discovered reserves of *1 Tera-ton* along with the actual annual production of *5,5 Giga-ton* (according to the International Energy Program [5]) makes the coal to one of the most essential energy sources of the world's industry, especially in the regions of East Europe, Asia, and Australia. At the same time, the coal industry is one of the most unsafe industrial sectors in terms of the work safety – the specific of the underground production (carried out at the depths of up to many kilometers under the surface), such as a limited ability of the natural air circulation through the underground mining fields or a frequent location of dynamic sources of hazardous gases, cause in the practice a considerable percentage of victims among the work labour due to technological accidents, such as the explosions of methane and oxygen mixtures.

Ventilation is the most important technology that is used for the safety ensuring in underground production areas of coalmines – it aims to provide mining fields with the fresh air conducted from the surface (in the amount that is necessary for carrying out the mining processes by the underground workers), as well as to drain the hazardous gases (mainly, CH_4 – methane) that are emitted during the mining and coal transporting from the underground production areas. The real VNs have to cover a widely distributed network of diverse interconnected underground airways (see Fig. 1) that span over many mining sections (e.g., longwalls) with diverse elements with the total length of up to many thousands of kilometers, as shown in Fig. 2. The ventilation is enforced by the *ventilation fans* (along with smaller local booster fans), mounted on the surface, that ensure an uninterrupted air circulation through all the elements of the VN sections: *main gates* (leading the fresh air from the surface), *faces* (areas of coalmining*), tail gates* (offtakes of air-gas mixtures), etc.

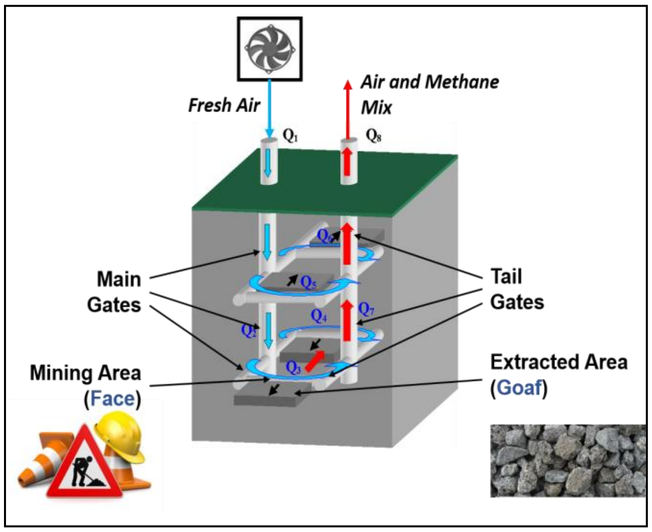


Figure 1. General structure of mine ventilation section.

The engineering analysis of the VN ventilation aims to solve a multi-level (including the airway, section, or network levels) regulation problem in order to determine operation modes of the major and booster fans (i.e., the depression values), values of the local regulators (e.g., the positions of *sluice valves*), etc., in order to saturate every VN airway with the airflow-rates required for the safe coalmining production. In practice, this is frequently performed with the help of static models, having the following general form:

$$P = RQ^2, \qquad (1)$$

where P – the air-pressure across the airway, Q – the air-flow rate, R – the air-flow resistance.

The air-flow model in the general form (1), applied to each and every VN element, results in the equation system that allows calculation of static ventilation properties for the whole VN. However, dealing with more complex tasks like the ventilation on demand (ensuring optimal energy consumption in constantly changing VN configurations due to gradual progress of the coalmining process), or solving any task that requires analysis of the gas emission from the coal-bed implies the analysis of dynamic behavior and makes the static model (1) inapplicable. Instead, the dynamic models are required at this place.

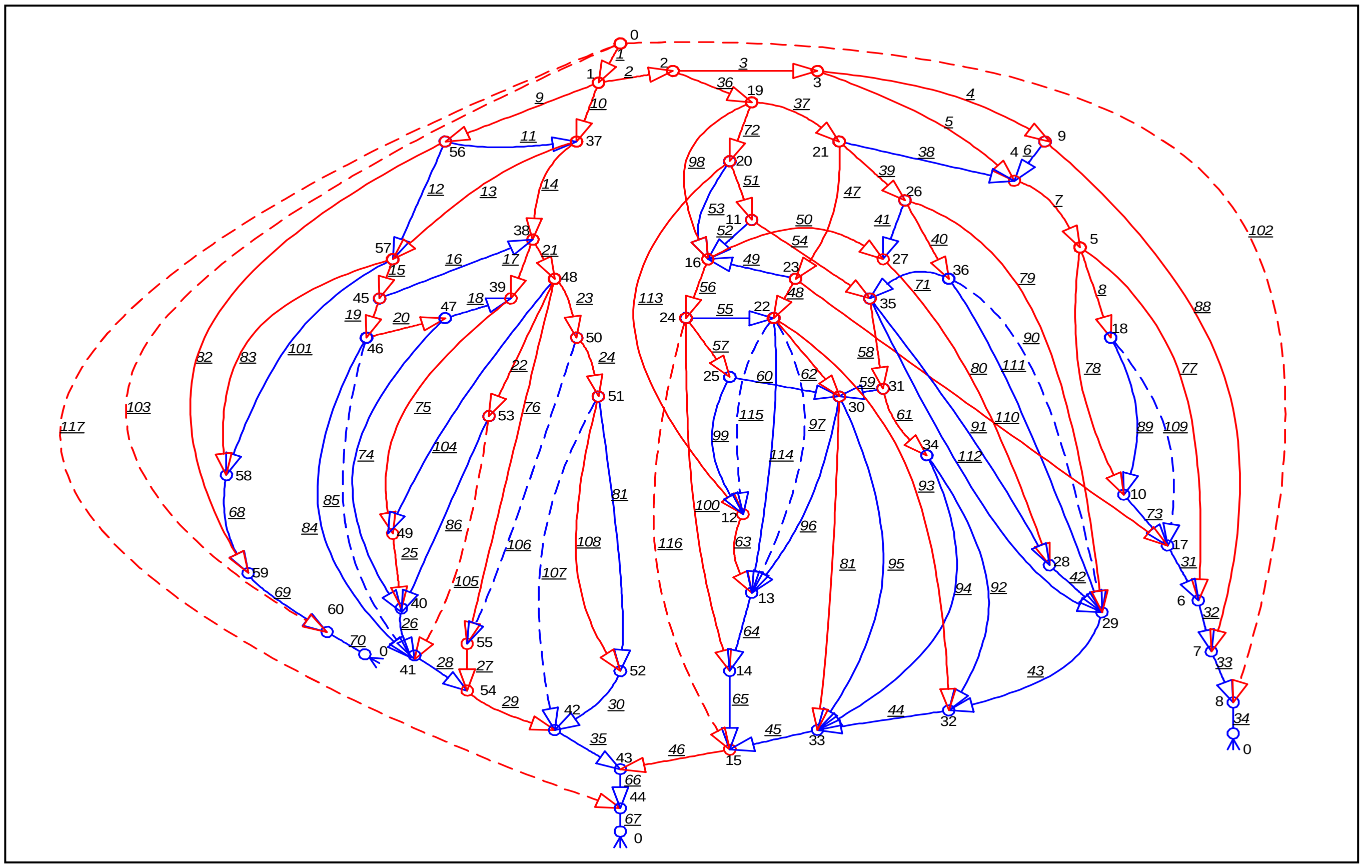


Figure 2. Illustration of real-complexity ventilation network with 117 branches and 61 connection nodes – coalmine South-Donbass-3, located in Ukraine. *The coloring of airways is used here just for better readibility purpose*.

The latter analysis task – gas-dynamics – is of a special importance for the coalmine safety, as most of the emergency situations in the coalmining industry happen due to combustion of the gases preserved in the coal veins [6]. Once past the *face*, the intake air that travels up the *main gate* and leaves the section through *the tail gate* is no longer fresh, but contaminated by mine gases, such as methane. Methane may be emitted from diverse dynamic sources in the section. The major source is, however, the *goaf* – a cavity between the *main* and the *tail gates*, which preserves the sealed methane that is carried out by the air-flows bleeding through it (parallel to the *face*, as depicted in Fig. 3). The concentration of the methane in the mixture with oxygen carried away by the *main gate's* bleeding flows through the *goaf* may relatively quickly reach the upper limit (of approx. 5%), which can cause spontaneous combustions of gases and coal in the *goaf*. Therefore, the gas-dynamics models are of a special importance for the coalmine safety, for the reason that they allow a prediction of potentially unsafe situations.

The gas-dynamics analysis relies on the models that consider dynamic aspects of the unsteady, multiphase (air- and gas-) flows, which are a way more complex than the static model (1). The general approach to the analysis of the dynamic problems involving fluids is the use of **Navier-Stokes (NS)** equations for a compressible flow (e.g., see a detailed explanation in Chorin [7]), covering different aspects of the fluids' macroscopic properties, such as the free-streaming and interaction with complex-geometry surfaces. Since the NS equations are non-linear and have no analytical solution, numerical techniques have to be applied. The common practical way used in the numerical CFD simulation is to apply an approximation and decomposition in order to simplify the original mathematical model whilst keeping the most essential fluid properties (e.g., flow rates, viscosity, pressures, etc.) sufficiently represented. A widely used VN aerodynamic model is a 1-D NS-approximation proposed by Svjatnjy [8], in the form of the equation system (2), consisting of a transport (impulse) and a continuity equation.

$$\begin{cases} -\dfrac{\partial P}{\partial \xi} = -\dfrac{2\rho}{F^2} Q \dfrac{\partial Q}{\partial \xi} + \dfrac{\rho}{F}\dfrac{\partial Q}{\partial t} + rQ^2 + r'(t)Q^2 \\ -\dfrac{\partial P}{\partial t} = \dfrac{\rho a^2}{F}\dfrac{\partial Q}{\partial \xi} - \dfrac{\rho a^2}{F} q \end{cases}, \qquad (2)$$

where t – the time, ξ – the spatial coordinate, and other values – diverse aerodynamic properties and coefficients.

The NS-based equations can also describe the state and transport properties of gas fluids. One of the "classic" models that describe a methane emission from the *goaf*'s filtration space was proposed by Feldmann [9]:

$$\int_V m \frac{\partial(\rho C)}{\partial t} dV = G_0 - G, \qquad (3)$$

where G – the gas-mass flow that is carried out by filtration air-flow from the goaf, G_0 – the gas-mass flow in the filtration volume of the goaf, C – the gas concentration in the filtration flow through the goaf, m – the coefficient of the goaf porosity, as well as other aerodynamic properties.

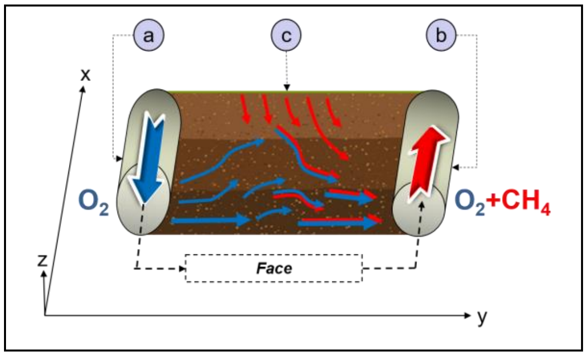


Figure 3. Methane emission schema in a U-type (longwall) section ventilation: a) main gate, b) tail gate, c) goaf.

The Feldmann's model (3) of the methane emission from the *goaf* can be approximated (in order to simplify the further numerical solution) by a 1-D representation, e.g., in the following form, as elaborated by Svjatnjy [8]:

$$\frac{mV}{q}\frac{dQ_m}{dt} + Q_m = Q_{m0} + \frac{mV}{q^2} Q_m \frac{dq}{dt}, \qquad (4)$$

where Qm – the methane-flow rate and q – the air-flow rate in the filtration space of the goaf.

The methane transportation by the flow in the section's *tail gate*, surrounded by the *goaf*, as well as in the further VN airways along the air-flow contour direction can be described, for example, by the following model built according to the Stewart's discrete cells approach [4]:

$$Q_{mt}^{i} = \sum_{j=1}^{N} Q_{mt}^{j}, \qquad (5)$$

where Q_{mt}^{i} – the rate of the methane-flow that is transported by the i-discretization element of the airway, N – the number of discretization elements that act as sources of the methane transported with the incoming air-flow.

The overall dynamic model of the complete VN can be built up from the models of each of its elements, i.e.:

- airways with only air-flow (e.g., the *main gates* leading fresh air to the section), such as represented by model (2),
- airways that contain active methane sources (e.g., the *goaf*), such as represented by models (2) and (4),
- airways that are transporting both air- and methane-flows from the adjoining airways (e.g., the *tail gates* carrying out the used air and methane from the section), such as represented by model (2) and (5),

as well as by the connective equations that describe the boundary conditions between the elements, as defined at the phase of discretization. Discretization can be performed by means of several methods, such as the Finite Differences (FDM), Finite Volume (FVM), Discontinuous Galerkin (DM), etc., see more details at [10]. Fig. 4 shows an example of a 1-D FDM discretization for a simple mining section containing a *main gate* (air-flow Q_{MG}), a *face* (air-flow Q_F), a *goaf* (air-flow Q_G and emitted gas-flow Qm_G), and a *tail gate* (air-flow Q_{TG} and gas-flow Qmt_{TG}) – see more details on FDM discretization for NS-based equations at [11].

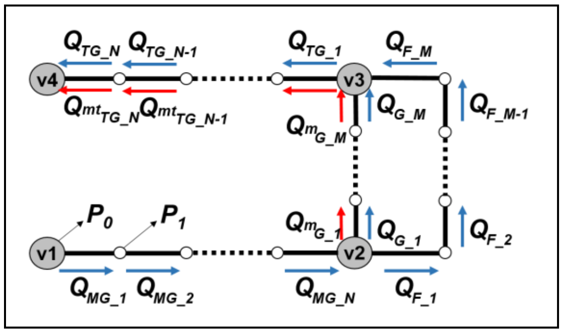


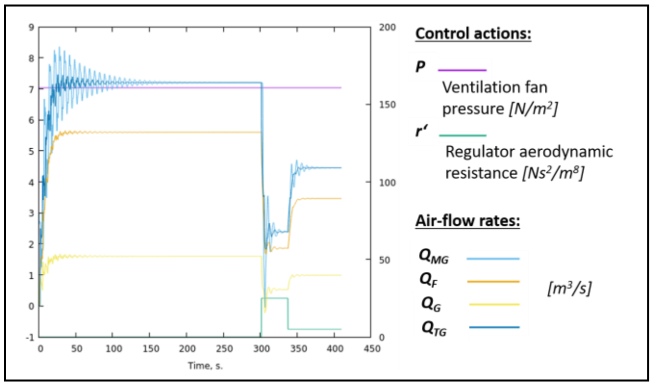
Figure 4. 1-D discretization schema of a mining section with a main gate (MG), face (F), goaf (G), and tail gate (TG).

The resulting 1-D model of a coalmining section can thus be shaped into the following form:

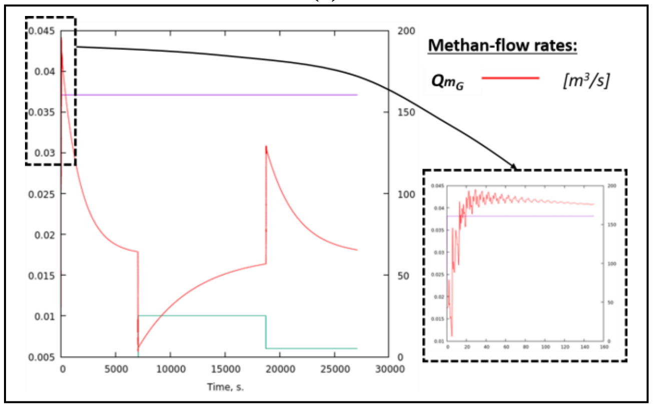
$$\begin{cases}
\frac{\partial Q_{MGj}}{\partial t} = -\frac{S_j}{\rho}\frac{(P_i - P_{i-1})}{\Delta\psi} - rQ_{MGj}^2 - r'Q_{MGj}^2 \\
\frac{\partial Q_{Fj}}{\partial t} = -\frac{S_j}{\rho}\frac{(P_i - P_{i-1})}{\Delta\psi} - rQ_{Fj}^2 - r'Q_{Fj}^2 \\
\frac{\partial Q_{Gj}}{\partial t} = -\frac{S_j}{\rho}\frac{(P_i - P_{i-1})}{\Delta\psi} - rQ_{Gj}^2 - r'Q_{Gj}^2 \\
\frac{\partial Q_{TGj}}{\partial t} = -\frac{S_j}{\rho}\frac{(P_i - P_{i-1})}{\Delta\psi} - rQ_{TGj}^2 - r'Q_{TGj}^2 \\
\frac{mV}{Q_{Gj}}\frac{dQm_{Gj}}{dt} + Qm_{Gj} = Qm_{0Gj} + \\
\qquad \frac{mV}{Q_{Gj}^2} Qm_{Gj} \cdot \left(-\frac{S_j}{\rho}\frac{(P_i - P_{i-1})}{\Delta\psi} - rQ_{Gj}^2 \right) \\
\forall_{j=1}, Qmt_{TGj} = Qm_{Gm} \\
\forall_{j=2..N}, Qmt_{TGj} = Qmt_{TG(j-1)} \\
\frac{\partial P_i}{\partial t} = -\frac{pa^2}{S_i}\frac{(Q_{j+1} - Q_j)}{\Delta\psi}
\end{cases}, \qquad (6)$$

where *Δψ* – the length of a spatial discretization element along the air-flow contour direction.

Application of a numerical solution method (such as Euler, Runge-Kutta, etc. – see more at Feldmann's survey [12]) leads to a resolution of the equation, in its general form (6), with the aim to obtain the evolution of the dynamic processes in time. Fig. 5 shows an example of the aerodynamics processes development by applying different control actions in a test section, controlled by a *ventilation fan* and a *local regulator*, installed in the *tail gate*.



(a)



(b)

Figure 5. Example of aerodynamics analysis by means of models: (a) air-flow in langwall, (b) methane-flow in mining section's goaf.

III. State of the Art in Simulation Technologies for Ventilation Networks Analysis

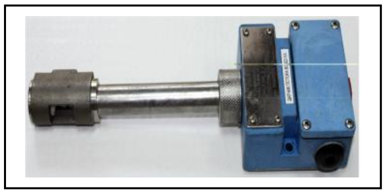
Implementation of the VN aerodynamics equations, such as needed for the ventilation control, happening on the basis of equations in the general form (6), can be performed in several ways. The simplest, in terms of developments efforts, is the usage of a wide-spread simulation package, such as *ANSYS-CFX* or *OpenFOAM*. Those packages provide a rich library of diverse solvers that can be applied on the target model equations. The other alternatives include:

- development of the own simulation platform (including discretization approaches, numerical solvers, visualization software, etc.) with more problem-specific customization options, however, requiring much more development and support actions, and
- use of the mixed (heterogeneous) environments with the customization of some specific functions of the standard simulation package.

However, regardless of the chosen implementation strategy and technology, the simulation applications have to deal with one critical problem – the model data actuality. Since the VNs are highly dynamic objects, meaning their parameters are continuously changing, at the latest after every production cycle (e.g., after an extension of the section, redirection of the air-flow contour), it is important to keep the models being constantly updated, in order to ensure that they can still be applicable on the production processes in their actual state and deliver trustful results.

In the reality, it has always been very difficult, if not even impossible, to refresh the data frequently. The so-called "ventilation surveys" – physical measurements of the aerodynamics parameter in the VN – could only be conducted for every 2 to 4 weeks, for the reasons of technological complexity. However, the parameter values between the last and the new measurements might change considerably, most notably – for the gas-dynamics. Therefore, the dynamic models have not found a "critical mass" application in the coal industry up to nowadays.

However, the things have been changing in the last years rapidly – the modern coalmines rely extensively on the **Cyber-Physical Automation (CPA)**, which offers, among other advantages, an advanced data acquisition and monitoring technology, supported by the Industrial Ethernet technologies like *PROFINET*, *Modbus*, etc., see some examples by Lee at [13] or Kay at [14]. One of the CPA systems that has successfully been taken in production is *UTAS*, widely utilized at the East-European coalmining industry space, for details look at Ignatovych [15]. The sensor network of UTAS provides both air- and methane-flow detectors, see in Fig. 6.



(a)

(b)

Figure 6. Example of UTAS smart sensors: (a) Methane detector TX3263D, (b) Air-flow rate measurement station TX1322D.

The key advantage of the CPA organization for the implementation of ventilation control is an immediate (i.e., in the real-time) availability of sensors data, which might be used for the fine-grain control of technological processes. Moreover, many CPA systems serves at the same time as a decentralized computation platform for "in situ" sensor data processing, i.e., when the sensor data are processed locally on the available digital resources – the so-called "sensor processors". The interconnection of all sensor processors allows the realization of global control operations, i.e., for the whole VN. On the other hand, the distributed, heterogeneous CPA infrastructure, which is preferable to be used for conduction of CFD simulation studies straightforwardly in the industrial environment, poses a major problem for the development of the simulation software, which cannot rely on the general-purpose development environments of the simulation software, such as the above mentioned *ANSYS-CFX* or *OpenFOAM*, designed for densely-organized, massively-parallel infrastructures – the High Performance Computers (HPC) or Cloud. Even for the custom simulation frameworks that partially rely on standard packages, there are still numerous adaptations and optimizations necessary, which are very difficult to implement, e.g., as the interaction between the sensor and modelling parts of the simulation workflow.

Development of the simulation software "from scratch" would allow to overcome the general compatibility problems of the standard simulation packages. However, it is a complex and non-trivial process. Moreover, the following CPA-specific infrastructure factors have to be considered:

- Heterogeneity of the CPA distributed infrastructure – many sensor devices are provided on the basis of a host system, whose architecture might differ from the typical HPC, Cluster, or Cloud environment but still requires a seamless integration within the distributed application workflows. However, the standard parallelization approaches require a uniform infrastructure with the compute nodes of the same hardware architecture and performance class.
- Limited flexibility of the mainstream parallelization approaches to support distributed application scenarios – many parallel applications rely on the Single Instruction Multiple Data (SIMD) technique, which mainly targets densely built compute systems like HPC. However, the applications that are running on the truly distributed infrastructures (HPC + Cloud + remote embedded systems) have to be developed according to the MIMD approach, in order to allow different functionalities to be executed on different types of systems.
- Composition complexity of the CFD models – models are often organized according to the applied mathematical decomposition technique to the initial physical problem, which have to consider different hierarchical levels with complex informational and control dependencies between the composition blocks of the models (see Fig. 7). The hierarchical physical connections between the objects have to be reflected in their models, accordingly.

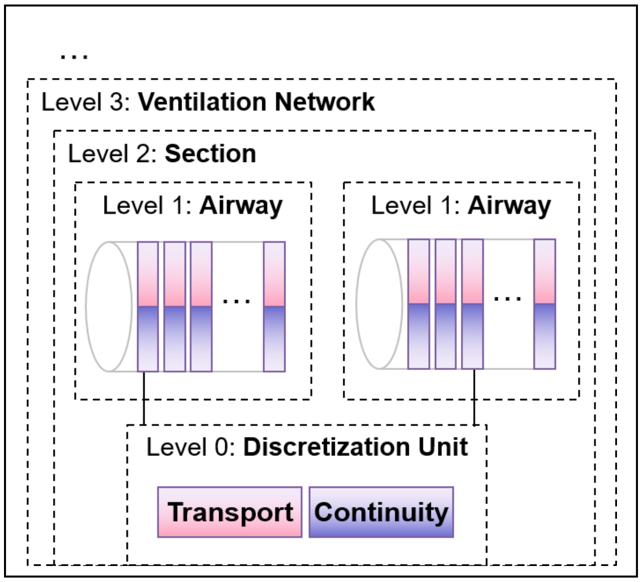


Figure 7. Hierarchical structure of composition blocks of ventilation network CFD models.

The simulation software needs to be developed in a modular way, where the modules would correspond to a part of the modelled physical system (according to the applied decomposition strategy), with the possibility of reuse of the topologically-assembled modules to implement the functionality of the higher-level models, i.e., in a bottom-up development approach. Therefore, novel development approaches are required for the implementation of the portable, horizontally and vertically scalable, as well as energy-efficient simulation software.

IV. MICROSERVICE ARCHITECTURE FOR DEVELOPMENT OF SIMULATION APPLICATIONS

In the last decade, **Service-Oriented Architectures (SOA)** have been successfully established in software development area as a more efficient (in terms of the required development efforts but also in terms of performance) alternative to the monolithic software design. The key concepts of SOA, such as the component-based application design, remote access of components via standardized communication protocols, etc., make the service-oriented development concept attractive for use in the simulation applications as well. However, due to the required peak performance on the highly optimized infrastructures, such as HPC, which could not be guaranteed by the standard SOA technologies, like the web services, the SOA concept has not found a wide application in the simulation applications domain. However, with i) the proliferation of the less strict, in terms of performance requirements, Cloud technology into the simulation domain, ii) appearance of heterogeneous Cloud-HPC testbeds, as well as iii) optimization of the SOA communication technology stack, the interest of the community in using the SOA technology started to grow again. With respect to the requirements discussed in previous Section III, it is proposed to use a **Microservice** (**MS**) architectures for the realization of the VN simulation framework (see Fig. 8).

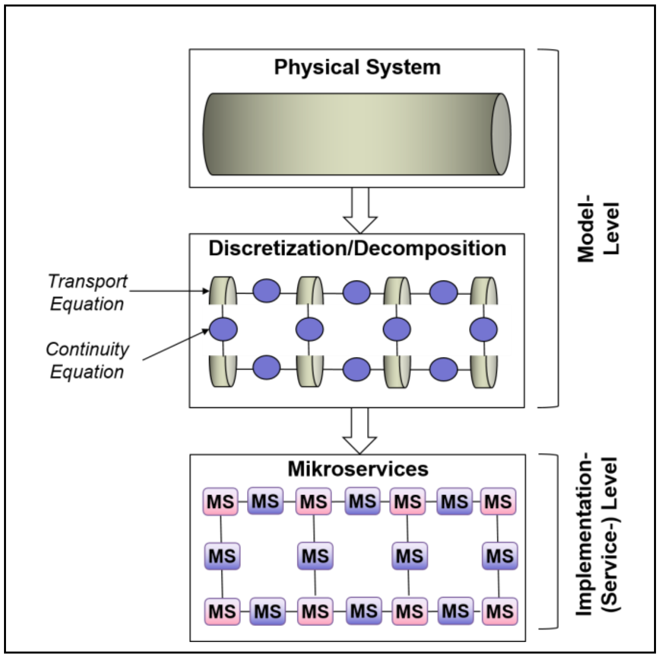


Figure 8. Composition of services for realisation of simulation application.

In this approach, a simulation application is arranged as a collection of independent, isolated, portable software modules (i.e., services), each of those implementing a composition block of the complex system, composed according to the functional, spatial, or any other decomposition. Each MS follows in its implementation a locality principle, i.e., bearing the responsibility for the assigned part of the complex system. In order to reflect the physical or informational connections of the real object, MS can be interconnected by means of a common data and/or control flow, which can be implemented with the help of a light-weight communication library, such as the technology-agnostic *HTTP*.

In order to evaluate the usability of the MS concept to carry out the VN simulation studies, as previously described in Section II, a reference framework *MS-Sim* was developed, available at the open-source platform Github [16]. The *MS-Sim* key concepts that are relevant to the simulation applications developments are as follows:

- Microservice model – specifies the development scenario for independent MS. The collection of different MS can be combined in a "market place" – library of domain-related services, which can be integrated into a common (simulation) application.
- Application model – defines the implementation strategy for MS-based (simulation) applications.
- Communication model – provides a realization of data exchange (e.g., control and data flow) between the loosely-coupled over the distributed infrastructure resources MS, as well as data storage.
- Hierarchy model – defines the implementation of hierarchically-dependent MS at the different granularity levels.

The *MS-Sim microservice model* provides a specification for every individual MS-module. According to it, MS are asynchronous and independent software processes, which are identified by a unique *Identifikator*. The MS can be data-synchronized with the other services by means of *ports* – special buffers in which the service can put the data that need to be communicated to the other services, or also obtain the incoming data (see Fig. 9). *MS-Sim* provides API to read/write data from/to buffers as well as to flush (by the sender) and synchronize (by the receiver) the buffer content.

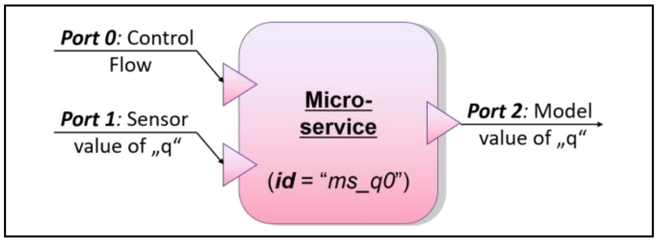


Figure 9. High-level microservice archtiecture.

At the time of service development, the connections between the MS, i.e., sources and destinations of data exchanged by means of *ports*, as well as the size of the ports' buffer, are irrelevant and can be established in the phase of creating a microservice application, i.e., by the MS-Sim *application model*. The application model specifies all the MS that are included into the application as well as connections between their corresponding ports, see Fig. 10. The MS are normally implemented in accordance to a pro-active logic, i.e., they are steered by a dedicated "master" service by means of a command flow – messages coded corresponding to a user-defined protocol, acknowledged across all MS, see example in Fig. 11. The application model distinguishes 2 categories of ports: normal buffers and *proxies*. The latter are used for internal communication with the dependent MS (e.g., between the master and the workers) and are not visible to the other application components interfacing this MS.

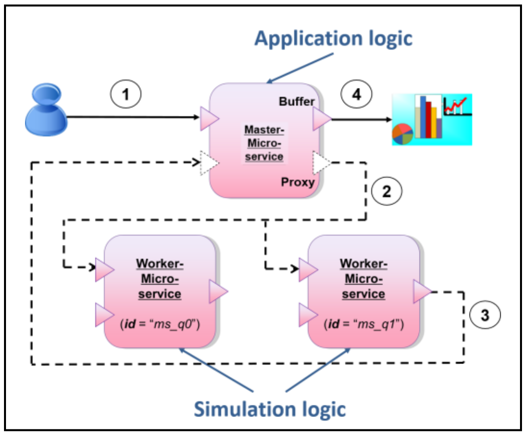


Figure 10. High-level microservice archtiecture.

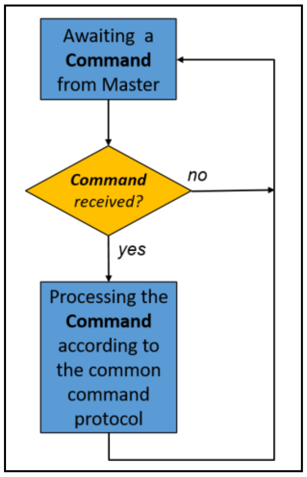

Awaiting a Command from Master
Command received?
no
yes
Processing the Command according to the common command protocol

Figure 11. Implementation of command flow by worker microservices.

The typical application workflow contains instructions to initiate some important for the simulation logic operations, such as the initialization of model's parameters (e.g., by the value obtained from the sensor), initiation of the simulation, control of the simulation experiment convergence, output of results, etc.

The data exchange between the master and the workers (e.g., for the propagation of the command flow or gathering results), as well as among the workers at the time of performing the simulation (e.g., for the implementation of the boundary conditions exchange), happens asynchronously and supported at the runtime of the (simulation) application execution by the MS-Sim *communication model*. In the communication model, the connection pairs (i.e., *communication links*) between the sender and the recipient MS are specified, as shown in Fig. 12. The information about the amount (number of entities) of data transmitted between the sender's and the recipient's ports may be added to the communication link as well (or assumed as a single data value, otherwise). At the physical network level, the data transport is performed by a special communication library, supplied to the microservice execution framework.

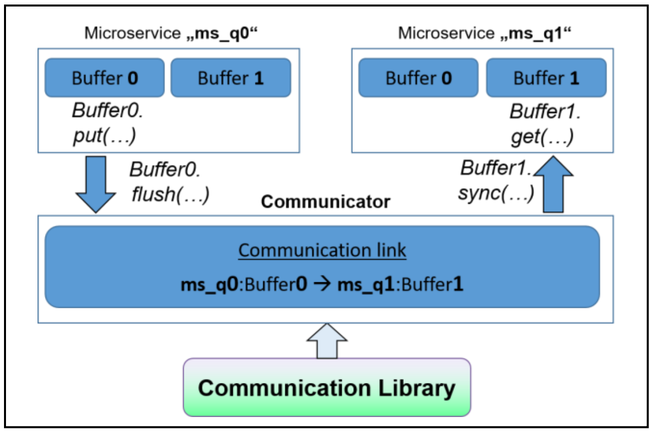


Figure 12. Realization of indirect communication between microservices.

The major dilemma, which nearly all microservice applications are facing during their design, is the definition of the granularity level of the initial problem that shall be implemented as a microservice. For example, in our case of the CFD applications, the lowest granularity level is served by a discretization element, and the highest – by a VN, see Fig. 7 above. In general, the lower is the chosen granularity level of the individual MS, the more customization options can be applied to the design of the MS-based application. However, the disadvantages of the very low chosen granularity level are getting very clear when the number of microservices in the MS-application is getting fairly big and the application has to spent most of the time in struggling with the management and coordination issues arising across the huge amount of very small MS. The "golden middle" strategy, which is used by some MS-based applications, is, however, not always applicable. On the contrary to the existing approaches, the MS-Sim design allows the developer (in our case – the simulation expert) to incorporate MS of different levels granularity in a single application, depending on the imposed customization and usability requirements. This hierarchical view is becoming possible to achieve by using the MS-Sim *hierarchy model* – a special methodology for creation of compound, hierarchically structured services. The MS-Sim hierarchical service concept allows not only to use the microservices of the different granularity in a common application, but also to create and reuse new, higher-level services on the base of the already existing ones. The developer will obtain the possibility to reuse services of any granularity level from a common "service market place". Fig. 13 shows an example of a microservice-based implementation of the U-formed coalmining section (depicted in Fig. 3) that combines the services of: i) discrete units (q, p) and ii) airways (Q, P) granularity level.

A prototype implementing the basic functionality of the *MS-Sim* framework was implemented with *OpenMPI* (see Gabriel et al. [17]) – an implementation of the Message-Passing Interface (MPI) standard, serving in our implementation as both a run-time environment and a communication library. *OpenMPI* is currently one of the most widely used solutions for designing parallel applications, mainly for HPC and cluster infrastructures. However, it is also useful for the implementation of MS-based applications: Same as the typical MPI applications, the SOA-based applications consist of a large number of components (microservices, in our terminology, or ranks, in the MPI terminology), which are running on distributed resource in isolated environments (as a software process). The components (regardless of whether they are microservices or MPI ranks) need to occasionally communicate between a couple of those (sender and receiver) or involving a group of components (i.e., collectively). In *MS-Sim*, every microservice is wrapped into an MPI process and is identified, in addition to its own Identifikator, by an MPI rank. The *communication library* (see Fig. 12) substitutes the data transfer calls with a corresponding MPI library calls, such as asynchronous point-to-point or collective communication.

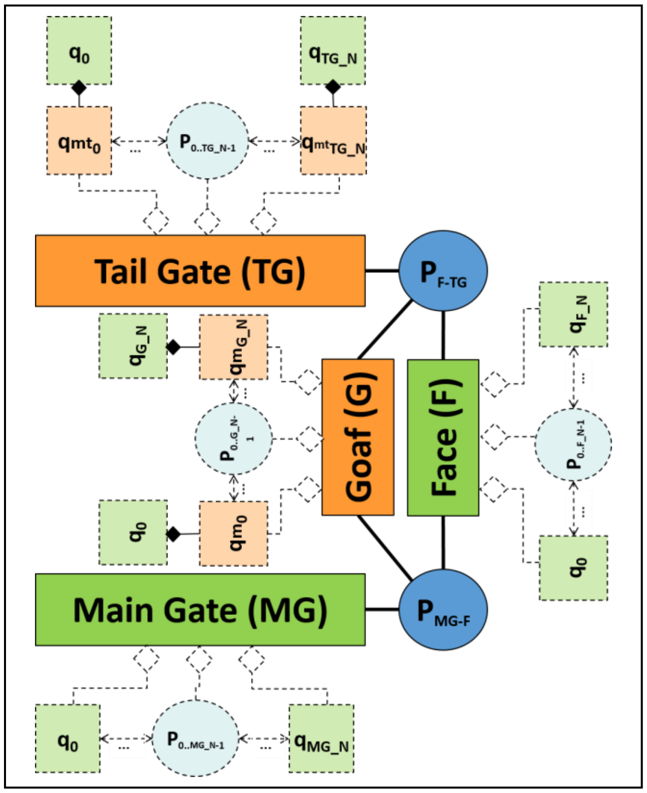


Figure 13. Implementation of ventilation section model with microservices of different granularity levels.

V. Validation and Benchmarking Results

The ventilation section model, depicted in Fig. 12, which includes the models of the air-flow through the *main gate* and *face*, air leakage into the filtration cavity of the *goaf* and methane emission, as well as transport of the air-methane mix through the *tail gate* was evaluated for a serious of typical changes that might be applied to the operational mode of the VN regulation elements – the main *ventilation fan* and local regulator (a *sluice valve* installed in the *tail gate*). The discretization of the physical domain was performed with the help of method of lines (NMOL), as served by the equation (6), with the approximation element length of $\Delta\psi$=50m. The numerical solution (time discretization) was performed with Runge-Kutta (RK4) method with the time stamp of 0.1 s. The service-based application workflow was steered by a dedicated supervising service – a manager, which performs the following functions:

- Initialization of microservices with the parameters that correspond to their respective physical object.
- Initiation of iterative numerical solution process.
- Control of the solution readiness by means of polling the status of every individual “worker service”.
- Instruct the services to store results in a local storage (the local disk).

Goal of the evaluation was: i) to validate the functionality of the models as compared with the real measurements from the test objects and ii) to benchmark the performance of simulation services in order to evaluate the applicability for real-time and HPC/Cloud usage scenarios.

The **functional validation** was performed on the basis of a test object (a section of the coalmine South-Donbass-3) for three subsequent regulation scenarios (*A*, *B*, *C*):

- substantial increasing of the ventilation fan pressure *H* from the value *0* to the full capacity *160,69 N/m²* (scenario A),
- decreasing of the air throughput by means of raising the regulation element’s value *r* from *0* to *25 Ns²/m⁸* (scenario B), and
- returning of the regulator in the initial position (scenario C).

The values of the airflows that resulted from simulation (see Table 1) were validate across the practical measurements data that were obtained by Svjatnyj [8]. The validation has not found any deviations between the modelling experiments and real measurements.

The **non-functional performance benchmarking** was performed for two following scenarios:

- Performance on a range of non-parallel systems: an embedded device (a low-power ARM-based embedded system Odroid-XU4 [18]), a generic desktop PC (Intel Core i5 CPU), and a server system equipped with a high-performance Intel Haswell CPU. The evaluated figure of merit was the ability to meet real-time requirements, i.e., $T_{sim} < T_{phys}$, where T_{sim} – the simulation time (duration of simulation experiment execution), T_{phys} – duration of dynamic process in real-time of the physical process. Only *main gate* model was used in this benchmark.
- Performance on a parallel (HPC/Cloud) infrastructure. The evaluated figure of merit was the parallel efficiency, measured as $T_{Sim}/(N_{Cores}*T_{Ncores=1})$, where N_{Cores} – the number of parallel processing units (e.g., CPU cores), $T_{Ncores=1}$ – the simulation time on a single processing unit. The evaluated system was served by a cluster with Intel Haswell CPU and Infiniband interconnect.

The performance results on non-parallel systems are shown in Fig. 14. In order to test the scalability on different hardware architectures, the evaluation was repeated for a doubled set of microservices (also involving more communication between them), which was achieved through refining the discretization mesh from $\Delta\psi$=50m. (corresponds to 8 microservices representing the modelled main gate) to $\Delta\psi$=25m. (i.e.,16 microservices, accordingly).

Benchmarking for the test case have revealed, that all systems were able to satisfy the real-time requirements (by the measured physical duration of the dynamic process of approximately 3 hours). As expected, the worst scalability results (decrease of performance by increased simulation’s computation load) were shown by the low-performance embedded system, however, without any implication on the usability in real-time scenarios.

TABLE I. VALIDATION RESULTS FOR TEST OBJECT

Parameter	**Experiment (A)**		**Experiment (B)**		**Experiment (C)**	
	Start	Finish	Start	Finish	Start	Finish
Regulation Actions						
Local regulator resistance **r´** *[Ns²/m⁸]* → r' *[Ns^2/m^8]*	0	0	0	**25**	25	**5**
Ventilator depression **ΔP** *[N/m^2]*	0	**160.69**	160.69	160.69	160.69	160.69
Measurement Values						
Airflow **Q** *[m^3/s]*						
- Main Gate	0	7.2	7.2	2.38	2.38	4.45
- Face	0	5.6	5.6	1.86	1.86	3.45
- Goaf	0	1.6	1.6	0.52	0.52	1.00
- Tail Gate	0	7.2	7.2	2.38	2.38	4.45
Simulation Results Values						
Airflow **Q** *[m^3/s]*						
- Main Gate	0	7.199	7.199	2.39	2.39	4.45
- Face	0	5.599	5.599	1.86	1.86	3.46
- Goaf	0	1.599	1.599	0.53	0.53	0.99
- Tail Gate	0	7.199	7.199	2.39	2.39	4.45

The important outcome of the nonfunctional properties evaluation is the fact that despite of being extremely compute time consuming and communication intensive, the CFD models of ventilation networks can be executed even on relatively small and with weak performance low-power systems, which are interconnected by means of commodity networks, such as Ethernet. Of course, imposing the necessary level of approximation (as in our case that deals with 1D-models) is the major prerequisite for the ability of such small systems to carry out the simulation successfully; however, the precision of the obtained simulation results allows the simulations to be used for the real cases in production without any considerable restrictions.

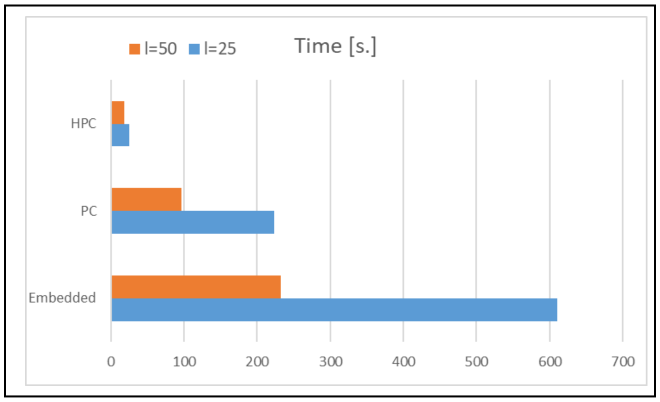


Figure 14. Performance results on diverse hardware architetures.

As for a highly optimized parallel high-performance system, the scalability of the applications is one of the major criteria for the usability of parallel applications. This characteristic was tested on a parallel cluster with Infiniband interconnect and Intel Xeon processors. The parallel efficiency that was obtained on the cluster is shown in Fig. 15. The implemented benchmark shows very positive results (efficiency over 100%) for up to 72 cores. Afterwards, the efficiency starts lowering, due to undersubscription of the CPU cores, as the number of microservices remains unchanged. Overall, according to the evaluation results, the pilot implementation of the *MS-Sim* platform, supported by *OpenMPI*, was able to achieve good performance and can be applied in both real-time and HPC simulation cases.

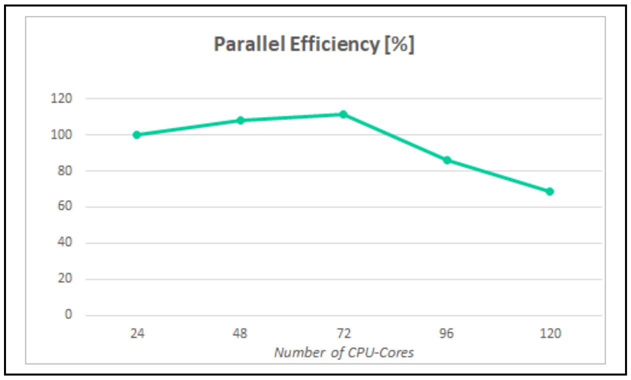


Figure 15. Parallel efficiency on cluster.

VI. Conclusion

The simulation technology is facing the challenges of application for new real-time scenarios that require a high flexibility of modelling tools in terms of the broader usage of the available infrastructure (data acquisition, storage, and processing devices). The rapid development of sensor networks has made possible a number of new innovative scenarios, for which the monolithic design of the existing simulation tools and workflow solutions on their top might be of a big obstacle. Service-oriented platforms offer a promising vision of the future development of simulation tools by offering benefits of on-demand distribution and parallelization, which might be well supported the underlying management platforms. Microservices are the technology that can if not fully replace the workflow-based scenarios but have the potential to support and bring them to the principally new level of usability. The effort that was done on implementation of the ventilation scenario has revealed a high potential of microservices architectures in geoscience and other domains of science and technology.

The dynamic approach, in which the services act as independent interactive components that are continuously running on the dedicated hardware and can be steered by a remote controller according to the specific application logic is particularly interesting for real-time control scenarios. In such scenarios, the services can incorporate the sensor data, make prediction for the future situation development, and instruct the control system about the probability of any potential risks appearance. On the other hand, the models can be optimized by adapting their parameters to best fit the actual mode of the controlled complex dynamic system.

The further research related to the application of the microservice simulation concept to the ventilation networks (and other dynamic systems with topological structure) will concentrate on the following activities:

- elaboration of functional composition strategies to development of complex, assembled service for hierarchically-organized systems,
- evaluation of the energy consumption of microservices on the embedded systems and elaboration of strategies to its minimization,
- extension of the communication library to support other commodity data transfer libraries, such as any REST-based communication frameworks, such as *RabbitMQ*, etc.,
- development of interfaces to communicate with the industrial sensors (field bus),
- simplification of API towards automation of complex microservices creation,
- modularization of the numerical solvers' library, in order to allow a seamless selection of the required numerical method to be applied on the models.

The other important direction of work will be the adoption of the microservice programming concept and the implemented MS-Sim framework by the further application scenarios in the HPC community. The advantages that are offered by the SOA style of programming MIMD parallel applications offers advantages for many other application domains in the engineering and other scientific computing fields. The developed microservice-based programming technology is expected to be useful for the design of environmental control systems, biomedicine applications, traffic prediction and control systems, as well as many others.

Acknowledgment

The work presented in this paper has become possible thanks to the support of the EU project *ChEESE*, which has received funding from the European Union's Horizon 2020 research and innovation program under the grant agreement Nº 823844.

References

[1] A. Cheptsov and O. Beljaev, "A Microservices Approach for Parallel Applications Design: A Case Study for CFD Simulation in Geoscience Domain", In proc. GEOProcessing 2020: The Twelfth International Conference on Advanced Geographic Information Systems, Applications, and Services, pp. 25-30, 2020.

[2] B. Bordel Sánchez, R. Alcarria, and T. Robles, "Cyber-physical systems: Extending pervasive sensing from control theory to the Internet of Things", Pervasive and Mobile Computing, 2017.

[3] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the Internet of Things and Industry 4.0", IEEE Industrial Electronics Magazine, vol. 11, no. 1, pp. 17-27, 2017.

[4] C. Stewart, S. Aminossadati, and M. Kizil, "Use of live sensor data in transient simulations of mine ventilation models", Mining Report 153 (4/2017), pp. 356-363, 2017.

[5] Internationales Energy Programm (IEA), "World Energy Balances 2019", [Online]. Available from: https://www.iea.org/statistics/coal/. Last access on 5.12.2020.

[6] Wikipedia. Statistics of major catastrophes in coal mines, [Online]. Available from: https://de.wikipedia.org/wiki/ Liste_von_Ungl%C3%BCcken_im_Bergbau/ 2020.03.24. Last access on 5.12.2020.

[7] A. Chorin and J.-E. Marsden, "A Mathematical Introduction to Fluid Mechanics", Springer Verlag, 2000.

[8] V. Svjatnyj, "Simulation of aerodynamic processes and development of control system for underground mine ventilation", PhD thesis 1985, (in Russian).

[9] L. Feldmann, "Dynamics analysis and automated control systems synthesis for coalmine ventilation", PhD thesis, 1974, (in Russian).

[10] H.J. Bungartz, S. Zimmer, M. Buchhoz, and D. Pflüger, "Model creation and simulation. A practise-oriented guide", Springer Verlag, 2013, (in German).

[11] R. Rannacher, "Finite element methods for Navier-Stokes equation", Lecture notes on theoretical and numerical fluid mechanics in Vancouver, 2019.

[12] L. Feldmann, V. Svyatnyy, M. Resch, and M. Zeitz, "Research domain: Parallel simulation technology", In proc. ASIM: The 3. International Conference on Simulation and Computer Graphic, pp. 139-168, 2009.

[13] E. Lee, "The past, present and future of cyber-physical systems: A focus on models", Sensors 15(3), pp. 4837-4869, 2015.

[14] J. A. Kay, R. Entzminger, and D. Mazur, "Industrial Ethernet - overview and best practices", Conference Record of 2014 Annual Pulp and Paper Industry Technical Conference, Atlanta, GA, pp. 18-27, 2014.

[15] N. Ignatovych, “Perspectives of implementation of automated contol system UTAS in coal-mines”, Problems of work safety in Ukraine, 17(2009), pp. 3-16, 2009, (in Ukrainian).

[16] MS-Sim reference framework implementationat Github, [Online]. Available from: https://github.com/alexey-cheptsov/MS-Sim. Last access on 5.12.2020.

[17] E. Gabriel et al., “Open MPI: Goals, concept, and design of a next generation MPI implementation”, In proc. 11th European PVM/MPI Users' Group Meeting, pp. 97-104, 2004.

[18] Pollin, Odroid-XE4 short specification, [Online]. Available from: https://www.pollin.de/p/odroid-xu4-einplatinen-computer-samsung-exynos-5422-2-gb-2x-usb-3-0-810409. Last access on 5.12.2020.