Windsurfing with APPA: Automating Computational Fluid Dynamics Simulations of Wind Flow using Cloud Computing

1st Anshul Jindal*, 2nd Benedikt Strahm†, 3rd Vladimir Podolskiy*, 4th Michael Gerndt*
*Chair of Computer Architecture and Parallel Systems, Technical University of Munich**
Independent Researcher†
Garching (near Munich), Germany
anshul.jindal@tum.de, benedikt.strahm@outlook.com, v.podolskiy@tum.de, gerndt@in.tum.de

***Abstract*—Computational fluid dynamics (CFD) can serve as a complementary approach to conventional wind tunnel testing to assess the wind flow around tall buildings. Being a clear High Performance Computing (HPC) task, CFD simulations conventionally run on supercomputers and compute clusters using specialized software such as OpenFOAM. The limited availability and high maintenance costs of supercomputers and clusters force small and medium companies to search for the cost-efficient infrastructure to conduct their simulations with the appropriate performance. The on-demand offer of compute capacity by cloud service providers are well suited this task. However, engineers and researchers require extensive expertise and experience in working with cloud computing in order to benefit from running CFD simulations on a cloud.**

The contribution of the paper to the outlined problem is two-fold: 1) a unique Automated Parallel Processing Application (APPA) tool that hides the cloud management details from the wind engineer and provides an intuitive user interface; 2) the estimation of the optimal number of cores (vCPUs) for virtual machine instances provided by AWS and Google Cloud based on average run time and total cost metrics for a given number of cells of a CFD-simulation. n1-highcpu-96 Google Cloud VM met both goals: low cost and low runtime per timestep. For the number of vCPUs below 16, the c4.8xlarge AWS VM type has the least runtime per timestep in all the cases. Google Cloud instances with high vCPUs are recommended to run the simulations if budget is a big concern.

***Index Terms*—cloud computing, wind flow simulation, computational fluid dynamics, domain-specific performance analysis, OpenFOAM, Google Cloud, AWS, containers, Docker**

I. Introduction

Wind engineering is a discipline that, amongst other, investigates the impact of wind flow on buildings in different environments. The importance of this discipline arises from wind loading on buildings being a potential cause of structural damage and collapse. An example would be the collapse of the Tacoma Narrows Bridge near Seattle in November 1940 due to wind-induced vibrations [1]. Since then, wind tunnel-based physical testing was established as the standard to evaluate the impact of wind on slender and/or long-span structures such as bridges and skyscrapers.

CFD uses numerical analysis to solve problems of fluid flows and allows to conduct simulations of turbulent wind flow around structures. These numerical methods are nowadays increasingly finding their way into the discipline of wind engineering on buildings and already play an important role in sectors such as the automotive or aircraft industry. However, such simulations require large amount of computational resources that can be provided by supercomputers and high performance compute clusters [2]. The high cost of acquiring and maintaining the required computing infrastructure, combined with a fluctuating demand for compute resources, may constrain small and medium enterprises (SME) in using CFD-simulations. Although it is possible to temporarily rent the compute resources from HPC centers and even to adapt them to the changing resource demand of the task at hand [3], the high amount of concurrent users, high costs and loose guarantees on fault tolerance make these solutions mostly unfeasible for practical use.

Cloud provides on-demand compute resources in the form of virtual machines (VMs) of different types, which vary in the amount of resources including the memory and the virtual CPUs (vCPUs) and in the price per unit of time. The on-demand provisioning of the compute resources in the cloud allows to make the infrequent data processing cost-efficient by undeploying the VMs as soon as they are not required. Along with the high guaranteed availability, this cost-efficiency makes cloud a good choice for conducting CFD simulations [4]. Albeit cloud services providers support some CFD solutions like OpenFOAM out-of-the-box by offering the corresponding VM images through their marketplace[1], still the following two challenges remain both for the individual wind engineers and for the SMEs that cannot afford having a full-time cloud computing expert: 1) general-purpose interface of cloud services providers console that is not customized to the level of skills of engineering workers; 2) unclear relation between the accuracy of the simulation, the time required to get the solution and the cost of the cloud resources used for the simulation.

The paper contributes to solving the outlined challenges in

[1]AWS, "CFD Direct From the Cloud": aws.amazon.com/marketplace/pp/B017AHYO16/

the following ways: 1) by introducing the Automated Parallel Processing Application (APPA) tool developed to make the setting and the execution of CFD simulations on AWS Cloud and Google Cloud intuitive via the though-through user interface; 2) by estimating the optimal number of virtual CPUs for VMs of AWS Cloud and Google Cloud based on the execution time and the total cost as determined for various configurations of the CFD simulations.

The rest of the paper is organized as follows. Section 2 provides the background knowledge in CFD and cloud services. Section 3 discusses the related works. Section 4 examines the architecture and implementation details of the developed APPA tool. Section 5 provides the experimental configuration used to determine the optimal number of cores. Section 6 presents the results of running the tests and discusses the selection of cloud services provider for conducting CFD simulations. Section 7 concludes the work and sketches future work directions.

II. Background

A. Computational fluid dynamics in wind engineering on tall buildings

CFD is a branch of fluid mechanics. It deals with numerical solutions of various fluid problems. The flow of air around tall buildings in the context of wind engineering and fluid mechanics is classified as an unsteady, turbulent Newtonian fluid nearly always in subsonic range.

Therefore, the appropriate mathematical model to describe the wind flow is the incompressible Navier-Stokes equation (1):

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + F_i + \frac{\mu}{\rho}\frac{\partial^2 u_i}{\partial x_j \partial x_j} \tag{1}$$

To perform numerical simulations for buildings, a so called virtual wind tunnel is used. It is a three-dimensional space that encloses the geometry of the building. The wind tunnel and the building are discretized in space into a computational cells as shown in Fig. 1. For each of these cells, numerical approximations of equation (1) are iteratively computed.

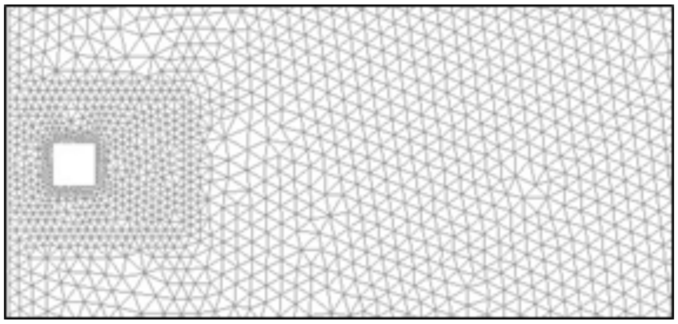


Fig. 1: Virtual wind tunnel discretized in computational cells

Since the problem is time-dependent, the geometrically discretized equations must also be discretized in time and solved at each time step.

The process of converting the flow problem from a physical statement into an approximated solution is summarized in Fig. 2.

Main difficulties in the solution of the Navier-Stokes equations arise from the turbulence of the flow. In order to correctly solve the flow problem, all turbulence scales have to be considered, leading to a practically unfeasible high computational effort [6]. Therefore typically Large Eddy Simulations (LES) are used, where small scales are not resolved and the effects of turbulence are predicted. This significantly reduces computational costs, but is less accurate than resolving all scales. Nevertheless, the solution of turbulent flow problems remains computationally expensive. This results from the large number of cells to be solved, which is amplified by the fact that typically a sufficiently large time period with fine time stepping is required.

For CFD-simulations there exists a wide range of commercial and open source software, each designed for specific flow problems. In the open source area, OpenFOAM, which has been applied in this research, is one of the most common software packages [7].

B. Cloud Service Providers

This subsection presents the Cloud Service Providers (CSP) whose compute resources were used in the scope of the research.

1) Amazon Web Services (AWS): AWS[2] is a cloud computing platform offered by Amazon to provide on-demand compute and storage resources. AWS provides Amazon Elastic Compute Cloud (Amazon EC2), a web service that offers scalable and re-sizable compute resources and allows users to launch different types of VMs varying by the amount of resources. Some EC2 instance types are optimized for specific use cases, e.g. compute-intensive or memory-intensive. The other AWS service which is used by APPA is Amazon Simple Storage Service (Amazon S3). Amazon S3[3] is an an object storage service allowing to store and retrieve any amount of data through a simple web interface. To store an object on S3, one needs to create a bucket which acts as a collection of objects [8].

2) Google Cloud Platform (GCP): The Google Cloud Platform[4] is a suit of cloud computing services offered by Google. As part of this suite, they offer Google Compute Engine (GCE), an Infrastructure-as-a-Service (IaaS) used to launch virtual machines of different types on-demand [9]. Apart from it, APPA also uses Google Cloud Storage to store the metadata and simulation results [10]. Google Cloud Storage is a RESTful web service to store and access the files through GCP infrastructure. It is also an IaaS service similar to the Amazon S3. The combination of GCE with Google Cloud Storage offers a Google-specific way to realize the CFD simulations on the cloud.

[2]https://aws.amazon.com/what-is-aws/
[3]https://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html
[4]https://cloud.google.com/gcp/

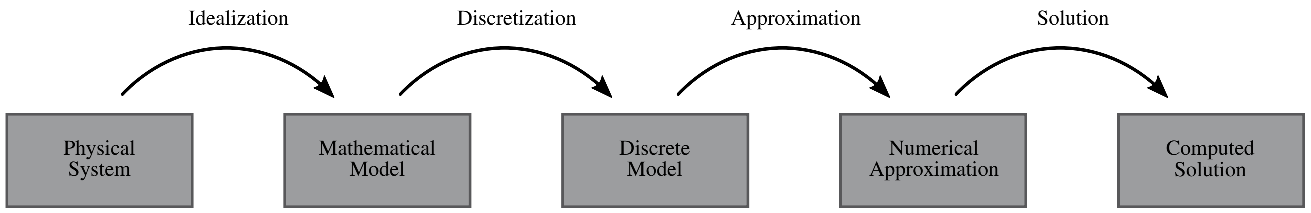


Fig. 2: Analysis process of CFD-simulations [5]

C. *Containerization*

Containerization is an OS-level virtualization method that allows to create containers which share the operating system of the underlying host [11]. Containers became a major trend in software development as an alternative to the traditional VMs. Containers wrap the application code along with its dependencies, so that it can be executed robustly on any infrastructure.

Docker is an open source containerization technology developed by Docker Inc which allows to execute application in an enclosed runtime environment without virtualizing the OS [12]. Docker containers became the de facto industry standard for containerizing the applications [13]. Docker uses images (similar to the virtual machine images) as a base for the containers. These images contain all the libraries as well as the OS kernel required for running the application. Containers are instantiated from such images similarly to VMs in order to run the application. Docker containers require the Docker software and enough resources for running them on a host. As part of this work, OpenFOAM docker image was used as the base image for building the application container [14].

III. Related Work

Slawinski, et al. [15] in their research ported two production CFD based simulations on four different platforms (one of which was Amazon EC2) and claimed that the Cloud laaS resources can be utilized for scientific CFD simulations possibly at lower cost than those incurred locally. However, they did the deployments of the simulations on the VMs manually due to cumbersome nature of the requirements and it took them up to a day for the deployment only. Ledyayev and Richter [16] benchmarked OpenFOAM on OpenStack platform and found out that OpenStack platform might not be well-suited for running OpenFOAM based simulations due to the performance degradation with respect to a reference computer of the same capabilities outside of OpenStack. However, they also claimed that with certain measures like changing the hardware and running multiple tests one can improve the performance.

OLeary, et al. developed a web-based simulation environment called as HPCCloud. The environment allows to create or define a simulation using the given input decks and then it submits to a cluster in the Cloud for processing. Their work lets the user create simulation through the web interface using the given input decks, however in our work, we have left the simulation creation to the domain expert and other stuff like deployment, running of simulation, monitoring etc. are offloaded from the user to the developed system. This allows the user to only focus on the application logic part.

IV. Automated Wind Flow Simulations in Cloud with APPA

A. *Architecture and functionality*

Automated Parallel Processing Application (APPA) tool was developed in the scope of the paper to automate the wind flow simulations on the cloud. Its implementation is in Golang and Python and consists of two major components – User Interface and Application Deployer. The high-level architecture of the tool and the communication paths between its components are shown in Fig. 3. Containerized APPA components can be individually scaled depending on the load. The tool partly automates the workflow of a CFD simulation, starting with the initialization of application requirements and ending with the collection of the final simulation results.

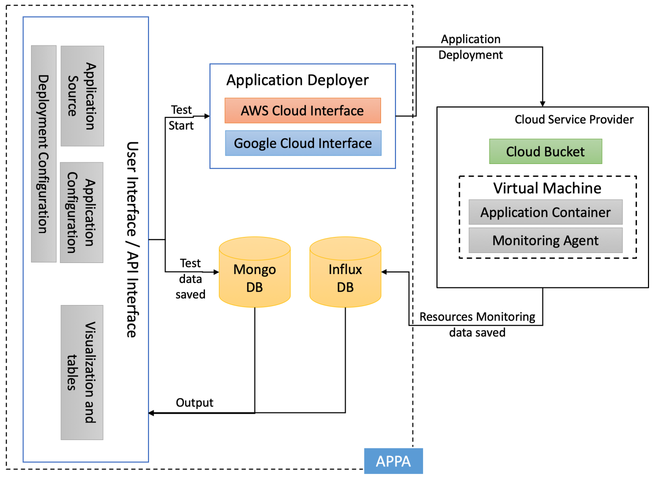


Fig. 3: Overall architecture of the APPA tool.

The automated simulation process starts with the user providing the OpenFOAM-based application source path along with the configuration parameters both for the application and for the deployment settings through the web interface or APIs.

Following, to prevent the user the hassle of preparing docker images the application is packaged into a Docker container and is deployed along with the monitoring agent in a VM of the supported cloud service provider (AWS or Google Cloud). The application metadata should be uploaded in the defined storage bucket of the selected cloud service provider beforehand using the option provided in the user interface and is automatically downloaded by the application container during its initialization process for further use.

On the completion of the preparatory steps, the simulation process starts. During the simulation, APPA continuously monitors the remaining simulation time by comparing the number of timesteps completed (stored inside the log files) to the total number of timesteps required for the simulation to be completed. Once the simulation is completed, the results are pushed to a new cloud storage bucket and the VM is terminated. The table representation of the test data information: the number of cells, VM type used, test case name, number of time steps completed, link to download the simulation results, start and stop time of the simulation, and link to the real-time resources usage monitoring is presented to the user. The user also has the option to download the simulation results. Fig. 4 shows the flowchart of the procedure of running simulations using APPA.

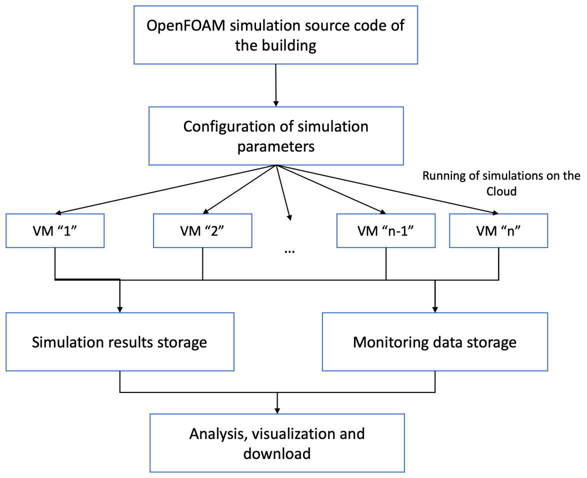

OpenFOAM simulation source code of the building
Configuration of simulation parameters
Running of simulations on the Cloud
VM "1"
VM "2"
...
VM "n-1"
VM "n"
Simulation results storage
Monitoring data storage
Analysis, visualization and download

Fig. 4: Flowchart of the procedure of running simulations using APPA.

In further subsections, we describe each component of the APPA.

B. User Interface

APPA encompasses the web user interface (UI) which allows the user to easily interact with the tool and conduct wind flow simulations using the dashboard shown in Fig. 5. Fig. 5a shows the dashboard enabling the user to configure simulation and deployment parameters. Also, the following parameters can be configured from this dashboard: application source, application configuration, and deployment configuration.

Application Source takes the OpenFOAM-based application source link as the input. Currently, the source link can only be provided as the GitHub link.

Application Configuration is responsible for setting the following configuration parameters of the application:

- **Maximum Time Steps** is the upper bound on the number of simulation time steps.
- **Number of Cells** is the number of computational cells of the simulated geometry.
- **Test case name** is the name of the test case to run from the application source; only required if the application source consists of multiple test cases.

Deployment Configuration is responsible for setting the following deployment configuration parameters:

- **Cloud Service Provider (CSP)** is the IaaS CSP on which the application is to be deployed and tested. Currently, AWS and Google Cloud are supported.
- **VM Type** is a type of virtual machine instance to be used for running the simulations, e.g. example n1-highcpu-32 of Google Compute Engine[5] and c4.8xlarge of AWS [6].
- **Cloud Bucket Name** is the name of the bucket where the metadata of the application resides.

Ongoing and completed simulations are presented to the user in the form of a table as shown in the Fig. 5b. This table contains the test information: the number of cells, VM type used, test case name, number of time steps completed, link to download the simulation results, start and stop time of the simulation, and the link to the Grafana dashboard to monitor in the real-time the resources utilization for each test.

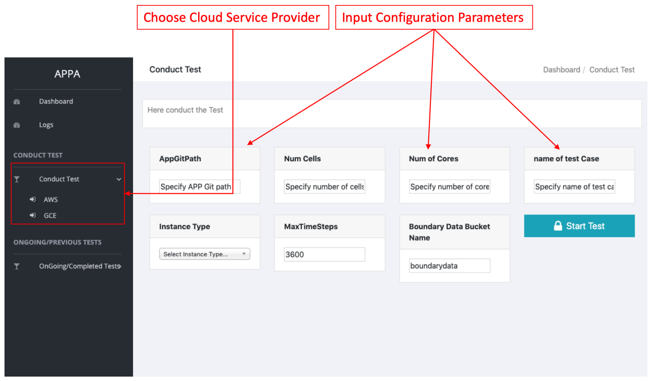
C. Application Deployer

Application Deployer is responsible for the deployment of the application on the chosen cloud service provider. Each provider offers an API to interact with its services. The application deployment starts by sending a REST API call to start a VM using the configuration parameters enclosed with the call. The application initialization steps: the creation of the application container, acquisition of the metadata from the bucket followed by the start of the simulation, are added as part of the VM instance boot script. When the simulation commences, the configuration parameters and the simulation start time are inserted into the MongoDB. Also, the current number of timesteps completed for the simulation is periodically updated in the MongoDB. Once the simulation is completed, the simulation results are stored in the cloud storage bucket and a VM termination request is sent to the cloud automatically. Lastly, the test termination time is updated in the MongoDB.

The adopted application deployment process allows starting multiple parallel simulations, thus reducing the time to get the simulation results.

A monitoring agent is deployed along with the application which continuously monitors the resources consumption (CPU, Memory, I/O, and network) of the container as well as

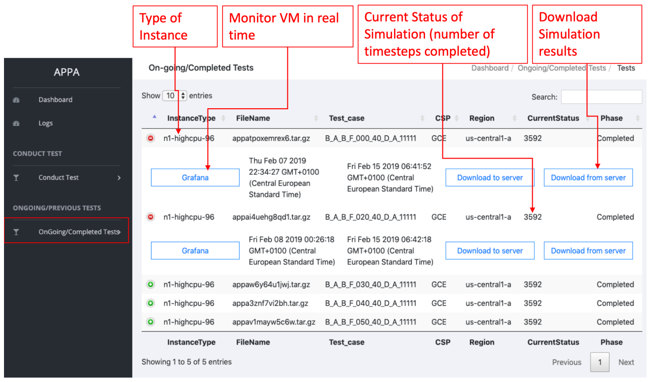
[5]GCE machine types: cloud.google.com/compute/docs/machine-types
[6]Amazon EC2 Instances: aws.amazon.com/de/ec2/pricing/on-demand/



Here conduct the Test

AppGitPath	Specify APP Git path
Num Cells	Specify number of cells
Num of Cores	Specify number of core
name of test Case	Specify name of test ca
Instance Type	Select Instance Type...
MaxTimeSteps	3600
Boundary Data Bucket Name	boundarydata

(a) Dashboard for configuring and starting the simulation.



InstanceType	FileName	Test_case	CSP	Region	CurrentStatus	Phase
n1-highcpu-96	appatpoxemrex6.tar.gz	B_A_B_F_000_40_D_A_11111	GCE	us-central1-a	3592	Completed
Grafana	Thu Feb 07 2019 22:34:27 GMT+0100 (Central European Standard Time)	Fri Feb 15 2019 06:41:52 GMT+0100 (Central European Standard Time)		Download to server		Download from server
n1-highcpu-96	appai4uehg8qd1.tar.gz	B_A_B_F_020_40_D_A_11111	GCE	us-central1-a	3592	Completed
Grafana	Fri Feb 08 2019 00:26:18 GMT+0100 (Central European Standard Time)	Fri Feb 15 2019 06:42:18 GMT+0100 (Central European Standard Time)		Download to server		Download from server
n1-highcpu-96	appaw6y64u1jwj.tar.gz	B_A_B_F_030_40_D_A_11111	GCE	us-central1-a	3592	Completed
n1-highcpu-96	appa3znf7vi2bh.tar.gz	B_A_B_F_040_40_D_A_11111	GCE	us-central1-a	3592	Completed
n1-highcpu-96	appav1mayw5c6w.tar.gz	B_A_B_F_050_40_D_A_11111	GCE	us-central1-a	3592	Completed
InstanceType	FileName	Test_case	CSP	Region	CurrentStatus	Phase

Showing 1 to 5 of 5 entries

Previous 1 Next

(b) Dashboard for viewing ongoing or completed simulations.

Fig. 5: Web user interface different views.

of the VM[7]. This data is continuously pushed to the InfluxDB for real-time visualization and further analysis.

V. Experimental Configuration

Experiments on AWS were conducted using the c4.8xlarge instance type that is optimized for compute-intensive workloads; this instance type has 36 vCPUs, 60 GiB of memory and attached storage of 150GB. Experiments on Google Cloud were conducted using n1-highcpu-32, n1-highcpu-64, and n1-highcpu-96 instance types with 32, 64, and 96 vCPUs respectively, and 28.8 GB, 57.6 GB, 86.4GB of memory respectively. Google Cloud instances were also equipped with 150GB of storage. All the machines ran OS Ubuntu-18.04-LTS-bionic with Docker version `18.03 CE` and Docker-Compose version `1.23.1` installed. OpenFOAM Docker image version `v1812` was used as the base image for creating the container of the application[8].

The simulated building is the Commonwealth Advisory Aeronautical Council (CAARC) standard tall building model [17], commonly used for benchmarking in wind engineering. The building geometry consists of a prismatic shape with a rectangular cross-section. The building and a surrounding wind tunnel form the computational domain as shown in Fig. 6.

The domain itself was decomposed into four zones with gradually refined grid sizes. Points that are far from the building do not require a fine grid as the smaller vertices there have limited influence on the building.

Further, two different grid resolutions, both with hexahedral cells, were created in order to demonstrate the influence of mesh refinement on the run time. The investigation of the physical meaningfulness of these simulations is not the subject of this study, but was examined in a preceding study, see [18].

This configuration results in the total number of **554,542** cells for Mesh A over **1,476,020** cells for Mesh B.

[7]https://github.com/ansjin/docker-node-monitoring
[8]hub.docker.com/r/openfoamplus/of_v1812_centos73

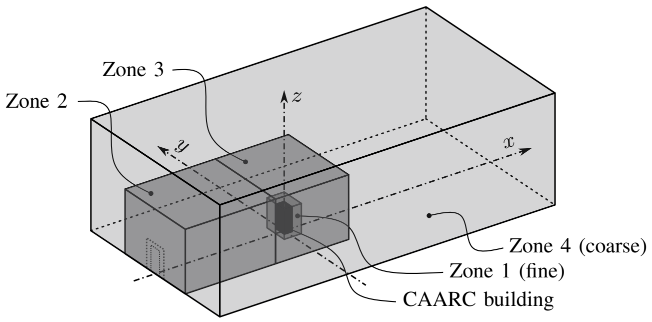


Fig. 6: Layout of the mesh refinement zones, H = Building height = 180m

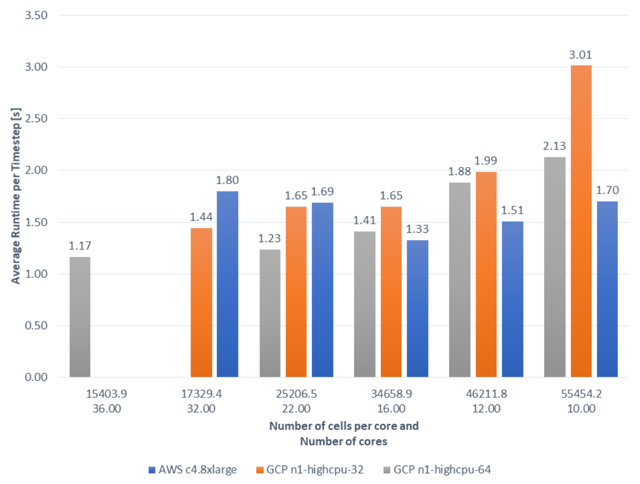
The number of time steps T_S was equal to **6,250** for each simulation, as this led to sufficient convergence to be able to draw reliable conclusions about the runtime [18].

VI. Experimental Results and Discussion

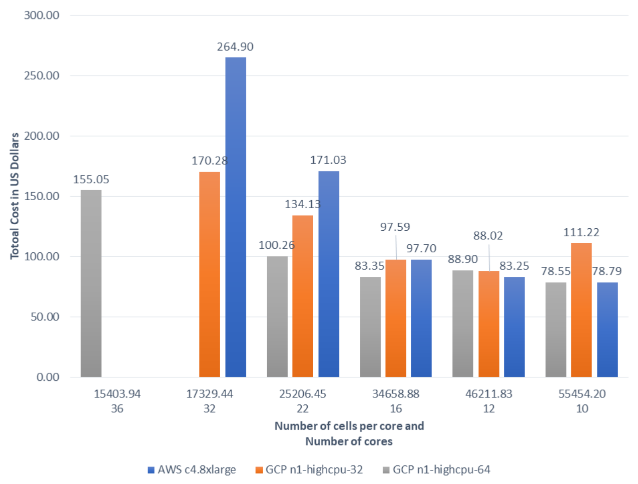
The evaluation was conducted to assess the execution time and the cost-efficiency of CFD simulations on the cloud with APPA tool.

The reduction of the simulation's execution time requires the optimal use of the computing capacity of the individual vCPUs. If the computing load exceeds the capacity of a VM, the parallelization can further reduce the execution time. Though, the increase in the number of cores used can result in higher inter-process communication overheads.

The computing load per core is proportional to the number of cells per core. The optimal number of cells per core depends on the hardware as well as on the granularity of the computational grid. Several simulations with the same configuration and a different number of cores were performed for the two meshes (Mesh A and Mesh B) on different CSPs using different VM types to determine how many cells per core yield a simulation optimal from the point of view of the execution time and from the point of view of the cost. The

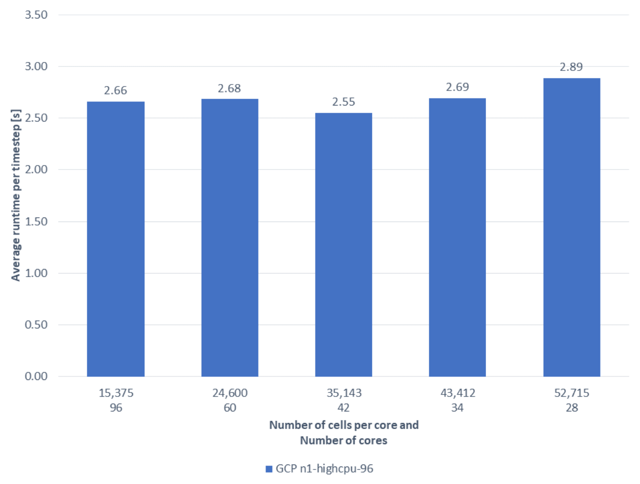


(a) Optimal number of cores based on average runtime per timestep.

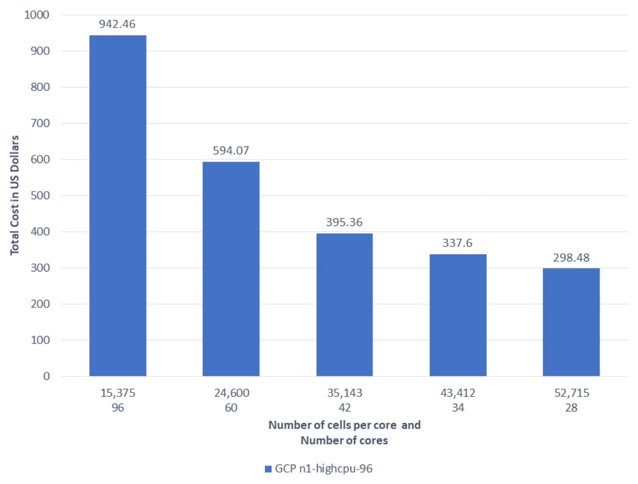


(b) Optimal number of cores based on the total cost.

Fig. 7: Performance analysis for Mesh A with different objectives.



(a) Optimal number of cores based on average runtime per timestep.



(b) Optimal number of cores based on total cost.

Fig. 8: Performance analysis for Mesh B with different objectives.

resulting graphs for both metrics are shown in Fig. 7 and 8. The missing value for 32 cores of n1-highcpu-64 instance type in the Fig. 7 is due to running out of budget before the end of the simulation.

Depending on the optimization objective and the accuracy requirements to the CFD simulation, one should select different cloud service providers and VM configurations. Based on the Fig. 7a one can conclude that the following VM configurations correspond to the goal of reducing the runtime per timestep for Mesh A simulation: **16** cores of the **c4.8xlarge** AWS EC2 instance type, **36** cores of the **n1-highcpu-64** GCE instance type, and **32** cores of the **n1-highcpu-32** GCE instance type. Comparing across AWS and GCE, one might notice that **36** cores of the **n1-highcpu-64** GCE instance type result in the least runtime of **1.17** seconds per timestep. On the other hand, if the objective is the low cost, then the optimal number of cores for both **c4.8xlarge** AWS and **n1-highcpu-64** GCE VM types is **10** cores, whereas **12** cores are required for **n1-highcpu-32** GCE instance type as seen in Fig. 7b. **10** cores of the **n1-highcpu-64** GCE VM type results in the least total cost of **78.55 USD**.

Test results for mesh B (which is finer than mesh A) exhibit the following trend in Fig. 8b: the lower the number of cores is, the lower is the total cost. **28** cores of the **n1-highcpu-96** GCE instance type has the least cost of **298.48 USD**. However, in Fig. 8a one can see that there is not much difference between the average runtime per timestep for different number of cores (with maximum difference of **0.34** seconds per timestep). **42** cores of the **n1-highcpu-96** GCE instance type result in the least average runtime of **2.55 seconds** per timestep.

In summary, **n1-highcpu-96** GCE VM met both goals: low cost and low runtime per timestep. Results summary for Mesh A and Mesh B are presented in Table I and Table II respectively. For the number of vCPUs below **16**, the **c4.8xlarge**

TABLE I: Results of the performance analysis for Mesh A tested on different CSPs and different VM types.

	Cores	**10.00**	12.00	16.00	22.00	32.00	***36.00***
	Cells / Core	55454.2	46211.8	34658.9	25206.5	17329.4	15403.9
AWS c4.8xlarge	**Total Runtime (in minutes)**	10629.30	9419.25	8291.73	10554.60	11239.00	#N/A
	Total Runtime (in days)	7.38	6.54	5.76	7.33	7.80	#N/A
	Avg Runtime per Timestep (in seconds)	1.13	1.00	0.88	1.12	1.19	#N/A
	Total Cost (in US dollars)	78.79	83.25	97.70	171.03	264.9	#N/A
GCP n1-highcpu-32	**Total Runtime (in minutes)**	18824.60	12415.60	10324.00	10319.00	9006.63	#N/A
	Total Runtime (in days)	13.07	8.62	7.17	7.17	6.25	#N/A
	Avg Runtime per Timestep (in seconds)	2.00	1.32	1.10	1.10	0.96	#N/A
	Total Cost (in US dollars)	111.22	88.02	97.59	134.13	170.28	#N/A
GCP n1-highcpu-64	**Total Runtime (in minutes)**	13296.20	11755.00	8816.69	7713.47	#N/A	7290.00
	Total Runtime (in days)	9.23	8.16	6.12	5.36	#N/A	5.06
	Avg Runtime per Timestep (in seconds)	1.41	1.25	0.94	0.82	#N/A	***0.77***
	Total Cost (in US dollars)	**78.55**	88.90	83.35	100.26	#N/A	155.05

TABLE II: Results of the performance analysis for Mesh B tested on n1-highcpu-96 (Google Compute Platform) VM.

Cores	**28**	34	***42***	60	96
Cells / Core	52715	43412	35143	24600	15375
Runtime (in min)	18049	16812	15938	16764	16622
Runtime (in days)	12.5	11.7	11.1	11.6	11.5
Avg. Runtime per Timestep (in sec.)	2.89	2.69	***2.55***	2.68	2.66
Total Cost (in US dollars)	**298.48**	337.6	395.36	594.07	942.46

AWS VM type has the least runtime per timestep in all the cases (lowest being **0.88** seconds for **16** cores). In contrast, the **c4.8xlarge** AWS VM type resulted in the highest total cost despite having the low runtime for vCPUs number equal to 16. If the budget is the key concern for CFD simulations, then GCE instances with high vCPUs are recommended to run the simulations.

VII. Conclusion

It is of great interest for CFD-simulations to have access to dynamic, high-performance and economic computing resources. Cloud computing provides a valuable response to these requirements and offers a preferable option for small and medium-sized enterprises that do not have their own computing resources.

The complexity of cloud computing led to a task of providing a user-friendly interface to conduct CFD-simulations on the cloud by engineers and of selecting the appropriate compute configuration to run experiments in accordance with execution time and total cost reduction goals.

This study addressed the outlined challenges by introducing the APPA tool to partially automate CFD-simulations in the cloud for AWS and Google Cloud. The real CFD-simulation runs had shown that the use of GCE instance types can result both in low total cost (**78.55** USD) and in low execution time per timestep (**0.77** seconds). Results of this study have also pointed out that lowering the number of cores results in the reduction of the total cost.

The promising directions of the future work include but are not limited to: 1) implementing the monitoring of the simulation time and of the resources usage to enable the optimization of resources by applying early stopping; 2) implementing dynamical scaling of the VM cluster used for the CFD simulation; 3) deriving the model that relates domain-specific parameters used for CFD-simulations to the total cost and the execution time.

VIII. Availability

The developed tool APPA is publicly available on GitHub under the link github.com/Cloud-Pie/APPA.

IX. Acknowledgements

This work was supported by the funding of German Federal Ministry of Education and Research (BMBF) in the scope of Software Campus program and also by GCP research credits. This work was also supported by the AWS Cloud Credits for Research program.

References

[1] W.-H. Hucho, *Aerodynamik der stumpfen Körper: Physikalische Grundlagen und Anwendungen in der Praxis*, ser. Grundlagen und Fortschritte der Ingenieurwissenschaften, Wiesbaden and s.l., 2002. [Online]. Available: http://dx.doi.org/10.1007/978-3-663-07758-9

[2] S. Lawson, M. Woodgate, R. Steijl, and G. Barakos, "High performance computing for challenging problems in computational fluid dynamics," *Progress in Aerospace Sciences*, vol. 52, pp. 19 – 29, 2012, applied Computational Aerodynamics and High Performance Computing in the UK. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0376042112000371

[3] I. Comprés, A. Mo-Hellenbrand, M. Gerndt, and H.-J. Bungartz, "Infrastructure and api extensions for elastic execution of mpi applications," in *Proceedings of the 23rd European MPI Users' Group Meeting*, ser. EuroMPI 2016. New York, NY, USA: ACM, 2016, pp. 82–97. [Online]. Available: http://doi.acm.org/10.1145/2966884.2966917

[4] P. Zaspel and M. Griebel, "Massively parallel fluid simulations on amazon's hpc cloud," in *2011 First International Symposium on Network Cloud Computing and Applications*, Nov 2011, pp. 73–78.

[5] C. Felippa, "Introduction to finite element methods," University of Colorado at Boulder.

[6] J. H. Ferziger and M. Perić, *Computational Methods for Fluid Dynamics*, third, rev. edition ed. Berlin, Heidelberg and s.l.: Springer Berlin Heidelberg, 2002. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-56026-2

[7] O. Inc. (2019) About OpenFOAM. [Online]. Available: https://www.openfoam.com/

[8] AWS. (2019) Buckets. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/Introduction.html#BasicsBucket

[9] G. Cloud. (2019) Machine types. [Online]. Available: https://cloud.google.com/compute/docs/machine-types

[10] G. C. Storage. (2019) Cloud Storage - Online Data Storage. [Online]. Available: https://cloud.google.com/storage/

[11] C. Education. (2019) Containerization. [Online]. Available: https://www.ibm.com/cloud/learn/containerization

[12] D. Inc. (2019) What is a Container? [Online]. Available: https://www.docker.com/resources/what-container

[13] C. Boettiger, "An introduction to docker for reproducible research, with examples from the R environment," *CoRR*, vol. abs/1410.0846, 2014. [Online]. Available: http://arxiv.org/abs/1410.0846

[14] openfoamplus. (2019) OpenFOAM(v1806) provided by OpenCFD Ltd. [Online]. Available: https://hub.docker.com/r/openfoamplus/of_v1812_centos73

[15] J. Slawinski, T. Passerini, U. Villa, A. Veneziani, and V. Sunderam, "Experiences with target-platform heterogeneity in clouds, grids, and on-premises resources," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum*, May 2012, pp. 41–52.

[16] R. Ledyayev and H. Richter, "High performance computing in a cloud using openstack," *Cloud Computing*, pp. 108–113, 2014.

[17] R. Wardlaw and G. Moss, "A standard tall building model for the comparison of simulated natural winds in wind tunnels," *CAARC, CC 662m Tech*, vol. 25, 1970.

[18] B. Strahm, "Validation of computational wind engineering techniques for tall buildings," Masterarbeit, Technical University of Munich, 2019.