Unified Cloud Orchestration Framework for Elastic High Performance Computing in the Cloud

Lukasz Miroslaw, Michael Pantic and Henrik Nordborg
Institute for Energy Technology, HSR Hochschule Rapperswil, Rapperswil, Switzerland

Keywords: Microsoft Azure, High Performance Computing, Cloud Computing, Cloud Orchestrator, Computational Fluid Dynamics, Simulations.

Abstract: The demand for computational power and storage in industry and academia is continuously increasing. One of the key drivers of this demand is the increased use of numerical simulations, such as Computational Fluid Dynamics for product development. This type of simulations generates huge amounts of data and demands massively parallel computing power. Traditionally, this computational power is provided by clusters, which require large investments in hardware and maintenance. Cloud computing offers more flexibility at significantly lower costs but the deployment of numerical applications is time-consuming, error-prone and requires a high level of expertise. The purpose of this paper is to demonstrate the SimplyHPC framework that automatizes the deployment of the cluster in the cloud, deploys and executes large scale and parallel numerical simulations, and finally downloads the results and shuts down the cluster. Using this tool, we have been able to successfully run the widely accepted solvers, namely PETSc, HPCG and ANSYS CFX, in a performant and scalable manner on Microsoft Azure. It has been shown that the cloud computing performance is comparable to on-premises clusters in terms of efficiency and scalability and should be considered as an economically viable alternative.

1 INTRODUCTION

Numerical simulations have become an essential part of modern R&D, as they allow companies to develop smarter products faster. Ideally, computer simulations can be used to test the performance of a product before the first prototype has been built, providing valuable information very early in the product development cycle. Computer simulations also allow companies to test a large number of design variation and to use numerical optimization algorithms to improve the performance of products.

Computational fluid dynamics (or CFD for short) represents one of the greatest challenges in terms of computational requirements. The advances in parallel computing facilitated the handling of the immense amount of data generated during complex simulations. Today, industrial-relevant simulations can only be efficiently executed on parallel compute clusters with hundreds of CPU cores connected through a fast high-performance network such as Infiniband. Unfortunately, such a computational infrastructure represents a significant investment and only makes sense if the computational cluster is heavily utilized to amortize the costs. This is generally not an issue for universities and research institutes, but represents a real problem for profit driven companies, who only occasionally need access to significant computing power. An underutilized compute cluster is simply too expensive to maintain.

Most modern CFD solvers support parallel processing through the Message Passing Interface (MPI). Recently, support for multi-core processors, GPUs, and the Intel Xeon Phi have begun to be implemented (Tomczak et al., 2013) (Che et al., 2015). Generally speaking, CFD simulations are very well suited for parallel processing. They can be easily parallelized using domain decomposition and the computational performance is mainly limited by memory bandwidth. Cloud computing has become ubiquitous due to its flexibility and cost-efficiency. It has clear advantages over traditional on-premises systems for institutions with limited budgets as no direct investments are needed and machines can be rented on a daily, hourly or even minute-by-minute basis. Users benefit from the cloud through the possibility of launching jobs of different sizes on dynamically sized systems and the cloud operators achieve economies of scale by building large data centers where the resources can be shared between different workloads.

Miroslaw, L., Pantic, M. and Nordborg, H.
Unified Cloud Orchestration Framework for Elastic High Performance Computing in the Cloud.
DOI: 10.5220/0005842402910298
In *Proceedings of the International Conference on Internet of Things and Big Data (IoTBD 2016)*, pages 291-298
ISBN: 978-989-758-183-0

An increasing amount of computing power is now hosted on cloud platforms such as Amazon Elastic Compute (EC2), Google Compute Engine or Microsoft Azure and more and more software and services are being hosted in the cloud. Many independent software vendors started to offer cloud services to scale scientific applications for specific industries. Companies like Rescale, Ciespace, Ubercloud, Sabalcore, Univa, Penguin Computing provide cloud services for weather research, computational fluid dynamics, structural mechanics, quantitative finance, combustion, electromagnetics and molecular dynamics (Gentzsch, 2012). They offer access to VMs with pre-installed software or a web portal where the scientific applications, such as ANSYS, Gromacs, OpenFOAM or Gerris solvers (Popinet, 2003) can be executed.
While this may be a satisfying solution for some, the ability to run custom or third-party software on the cloud infrastructure requires much more complicated procedures. A scientific application that needs to be deployed in the cloud usually consists of a command line-tool that requires complex deployment steps such as installation, configuration as well as setting up system specific services and network policies, comparable to the effort of administering an on-premises cluster. Access to the deployed applications from a local machine is also a tedious procedure and requires a significant amount of effort. This partly explains why this scenario is often too cumbersome for a domain engineer or a scientist.

A simple mechanism is needed to simplify the whole process and lower the entry barrier for cloud-based numerical simulations. Such mechanisms exist for Amazon EC2 or Eucalyptus but the second biggest public cloud provider, i.e. Microsoft Azure still lacks an easy-to-use framework to simplify cloud orchestration, and this is the reason why we targeted this cloud provider.

In this paper, we present a unified framework, named ”SimplyHPC”, that greatly simplifies the use of a distributed application on Microsoft Azure. The framework combines Azure specific HPC libraries, deployment tools, machine and MPI configuration into a common platform. Its functionality is, among others, exposed through PowerShell commandlets that ease the submission process while keeping the desired command line environment and flexibility. The framework provides tools with the ability to elastically deploy an arbitrary number of virtual machines dynamically, submit the packed application together with input and configuration files, execute it as a cloud service using MPI and, once the results are ready, download them to the local machine and stop the virtual machines. The results presented here is a follow up research of our recent studies (Miroslaw et al., 2015)

Our paper focuses on the following aspects. Section 3 describes the main components of the proposed framework as well as the platform architecture. Section 4 demonstrates the utility of the tool on two scientific applications, namely PETSc and HPCG. In addition we present the scalability study of ANSYS CFX, a commercial fluid dynamics code for realistic, industrial simulations. The paper ends with conclusions and future plans.

2 BACKGROUND

This section examines the typical impediments in deployment of HPC applications. It also briefly examines related technologies and introduces the platforms and technologies used in performance studies in the cloud and in the on-premises cluster.

Due to the fact that the cloud providers are commercial entities that compete on the market, it is very difficult to create a single orchestrator that supports the major cloud platforms. The cloud infrastructure changes very quickly, the new APIs, services and tools are released frequently and addressing them in one consistent software is very difficult. For example existing libraries such as Apache *jclouds* or *libcloud* do not support HPC functionality based on Microsoft technologies. This is the reason why we decided to target the Microsoft Azure platform with our cloud orchestrator.

Cloud orchestrators, such as the one presented in this paper, manage the interactions and interconnections among cloud-based and on-premises units as well as expose various automated processes and associated resources such as storage and network. They have become a desired alternative to standard deployment procedures because of lower level of expertise and reduced deployment time. Also their ability to perform the vertical and horizontal scaling is fundamental to the adoption of the framework in HPC scenarios.
HPC Pack is a cloud orchestrator developed by Microsoft for monitoring, executing and running jobs in both on-premises and in the cloud. It exposes functionality that is typical for cluster management software such as deployment of clusters with different configurations, a scheduler that maximizes an utilization of the cluster based on priorities, policies and usage patterns and a batch system for submission of multiple jobs with different amount of resources. The framework also allows for deployment of hy-

brid clouds, for example deploying a head node on-premises and controlling Azure PaaS nodes or deploying the head node as a IaaS VM and controling PaaS VMs from the local machine. Although it is a recommended framework to control the cluster in the Azure cloud, we will show that HPC Pack is not suitable for high performance computing by comparing its deployment time with SimpyHPC.

2.1 Related Work

Although cloud orchestrators play an important role, still there are only a few easy to use, out-of-the box platforms that simplify deployment, monitoring and accessing scientific applications in the cloud. The best situation is in Amazon EC2 where frameworks to simplify the process of orchestration of specific scientific applications exist. Linh Manh Pham et. al presented a distributed application Roboconf orchestrator for multi-cloud platforms and domain specific language to describe applications and their execution environment (Pham et al., 2015). Wong and Goscinski propose a unified framework composed of a series of scripts for running gene sequencing analysis jobs through a web portal (Wong and Goscinski, 2013) on Amazon EC2 (IaaS) infrastructure. For this purpose they designed a HPC service software library for accessing high level HPC resources from the IaaS cloud and demonstrate their framework in running mpiBLAS . Balgacem et al. measured the multiphase flow on a hybrid platform combined with Amazon *EC*2 and a private cluster (Ben Belgacem and Chopard, 2015). Marozzo et. al (Marozzo Fabrizio, 2013) present a web-based framework for running pre-defined workflows composed of web services in Microsoft Azure. The authors demonstrate the framework in data mining applications by deploying an execution workflow running a single classification algorithm and measure strong scalability in classifying different data sets that were previously partitioned. Last but not least, it is important to mention the recently released Azure Batch service aimed at running cloud-scale jobs. However, there is a significant amount of programming skills needed to deploy an application.

2.2 Microsoft Azure

Microsoft Azure is a cloud computing platform that provides, among many other services, virtual machines (VMs) in both IaaS and Paas models. PaaS machines, also called *Worker roles*, are mainly targeted at developers seeking to execute custom software in a high-availability or distributed fashion. In this model, software running on worker roles can be automatically scaled and copied to many nodes.

A set of worker roles constitutes a *Cloud Service* on Microsoft Azure. Instead of configuring all VMs manually, the user uploads a packaged application together with configuration files for a cloud service. This ”deployment package” is then installed on fresh VMs automatically and in parallel. This procedure usually takes a few minutes and is attractive in terms of usage costs, as these non-persistent VMs can be created when needed and deleted immediately after usage. Regular VMs (IaaS) are bare machines deployed with a specific OS that cannot be scaled manually or programatically within the Azure Cloud but can also be scaled and managed through external frameworks such as Ansible, Chef or Puppet.

In our framework we decided to take advantage of the cloud service (PaaS) to simplify the orchestration procedure and use Blob and Table storage entities for data management. Because the storage can be locally or geo-redundant, a high availability and persistence of the results from the simulations could be easily guaranteed.

2.3 Numerical Software

Sparse linear algebra inspired the use cases presented in this study. Large sparse matrices are generated during the discretization of partial differential equations and often appear in scientific or engineering applications. We have selected two scientific software tools, namely PETSc and High Performance Conjugate Gradient (HPCG) Benchmark to demonstrate the utility of our framework, to measure the performance on both Microsoft Azure and the local cluster.

In addition to the numerical tools, we have selected ANSYS CFX, a high performance fluid dynamic software that is well recognized in CFD community and has been successfully applied to different fluid flow problems.

3 ARCHITECTURE

In this section, the three main architectural aspects of SimplyHPC are presented. Firstly, we show the system components visible to a single node such as cluster settings, communication setup and Azure-specific elements. Secondly, we discuss the structure of the SimplyHPC framework at the cluster level and explain how the software automatizes the setup of nodes and clusters using the given system architecture. Thirdly, we briefly present the software architecture.

3.1 Node Architecture

As implied in Sec. 2, there are different type of machines available on Microsoft Azure. Here, Worker Roles are used, as they have various advantages over traditional virtual machines, especially in terms of manageability and scalability. Worker roles are set up using a *Deployment Package*. This package contains a full description of the virtual machine, including machine-specific configuration such as local storage requirements, network settings, security settings as well as software specific settings. New Worker roles are then automatically set-up according to this deployment package. This is done by the server management software, subsequently called the *Azure Cloud Controller* that exposes these APIs and is provided by Microsoft as the main interface to the Azure Cloud.

In Fig. 1, the schematic architecture of a single node is shown (one worker role equals to one node of a cloud cluster).

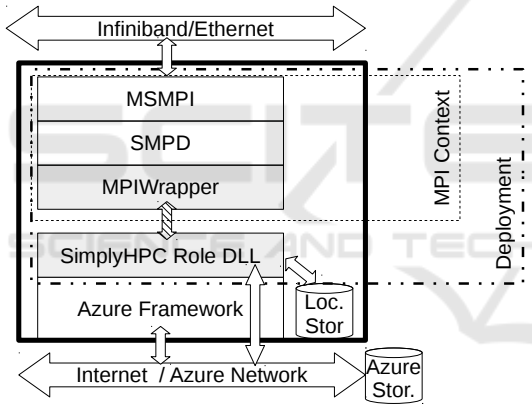

Figure 1: Schematic view of a single compute node on Azure.

It is important to note that this node architecture applies to all nodes in a cloud cluster, and is automatically configured using the aforementioned deployment package.

3.2 Cluster Architecture

Fig. 2 visualizes all relevant components of a deployed HPC cluster that has been set up using the SimplyHPC framework. A *cloud service* (dotted rectangle in figure) represents a bundle of multiple nodes using the same deployment template. Using the logical entity of a cloud service, nodes can be easily configured in terms of their number, size and the virtual network they belong to. When using A8/A9 nodes for scientific computing, the inter-node network (depicted as the upper arrow in Fig. 2) is realized with Infiniband. In this mode all inter-node storage and MPI requests are transparently redirected to this very low latency, high bandwidth interconnect. Such configuration is crucial for performance with most scientific software that take advantage of concurrency.

The availability of fast inter-node connections strongly influences the choice of storage for different tasks.

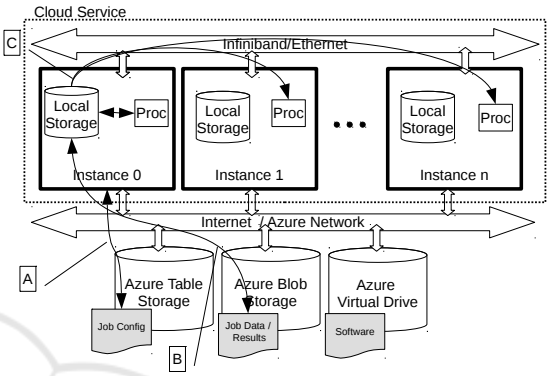

Figure 2: Schematic view of a SimplyHPC Cluster on Azure.

3.3 Software Architecture

SimplyHPC has been written in .NET and uses Azure mechanisms that automate the process of setting up the cluster described before. There are multiple levels at which the SimplyHPC interacts with Microsoft Azure. It is linked to the Microsoft Azure SDK and uses parts of it to automatize the compilation of the deployment packages. On a run-time level, it interacts directly with the management interface of Microsoft Azure to create cloud services, upload deployment packages, monitor machine status, etc.

We expose two different front ends to the user, namely PowerShell commandlets and a high-level API with a facade that hides all the complexity of orchestrating the cloud. PowerShell is a object-oriented command line environment used to script and automatize common tasks.

PowerShell commandlets are building blocks that allow administrators to prepare custom scripts. Its interface provides a number of useful commandlets needed to deploy a scientific application in the cloud, execute it, download the results back to the user and destroy the cluster. The users of the software have also the possibility to adapt the code to their needs since the software is freely available at Github (see the last Section for more details).

4 RESULTS

In the following sub-sections we compare the set-up time of the cluster created with SimplyHPC with HPC Pack, since this is a standard tool for cluster deployment that is currently supported and recommended by Microsoft. We also measure the performance of cloud clusters on Microsoft Azure and regular on-premises cluster using test cases relevant for CFD applications. Although these measurements are not representative due to different hardware configurations on both systems, they still provide insights on the performance of numerical code in the cloud. These examples serve also as an indication that running different kinds of simulation with SimplyHPC is feasible.

All tests were performed either on Microsoft Azure (cloud cluster) or on the on-premises cluster. On-premises cluster is a privately owned and maintained system at Microsoft Innovation Center at the HSR Hochschule Rapperswil (HSR), Switzerland. It is composed of 32 nodes with 48GB RAM and two Intel Xeon E5645 6 cores each, the nodes are connected with 40Gbit Infiniband Network. On each node, Windows Server 2012 R2 with HPC Pack 2012 R2 is installed. On Microsoft Azure two VM types were used: A8 nodes with 8 core Intel Xeon E5-2670 and 56 GB RAM and A4 nodes with 8 core CPU, 14 GB RAM and GigE interconnect. High-bandwidth Infiniband interconnect between the nodes was possible only for A8 and A9 nodes. For performance studies of PETSc and HPCG we used A4 and A8 nodes while for ANSYS performance tests we used A8 nodes only. For more detailed information about Microsoft Azure node types see the official specification.

4.1 Set-up time

The deployment time of a cluster in the cloud is crucial when using dynamically allocated systems and running computationally intensive tasks composed of a single job or a small series of jobs. In such scenarios the cluster is created to perform the computational tasks and is destroyed afterwards. We do not consider the set-up time on on-premises systems and focus on the cloud-based scenario only.

We compared the time needed for setting up a cloud cluster consisting of A8 nodes with SimplyHPC and Microsoft HPC Pack. This time should be of the same order regardless of the node size, because the deployment processes are similar. HPC Pack provides precise statistics on the deployment procedure that includes also creating a storage account, a virtual network, a cloud service, a domain controller as well as deployment and configuration time for the head node and compute nodes. This is also the reason why the deployment procedure takes more than an hour. Since in SimplyHPC most of these processes are omitted, as expected the time needed to deploy the head node and compute nodes was much shorter, e.g. we measured the time needed to deploy a new cloud service with a single job. Since HPC Pack neither provides automatic job submission nor it retrieves the result we have not taken these processes into account in our measurements.

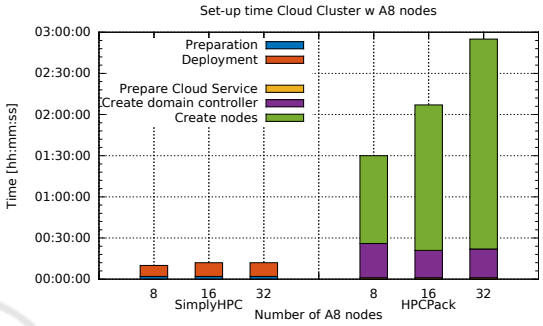

Figure 3: Deployment time of cluster composed of different number of A8 nodes with Microsoft HPC pack (right) and SimplyHPC framework (left). Time is provided in hh:mm:ss format.

4.2 Sparse Matrix Solver Performance

The objective of the second experiment was to compare the performance of a cloud cluster on Microsoft Azure with a traditional on-premises cluster using common distributed parallel sparse matrix solvers such as HPCG (High Performance Conjugate Gradients) Benchmark as a standardized performance test and PETSc solver being applied to two different real world problems. On Microsoft Azure, the HPCG results scale nearly linearly with the number of cores, with a maximum speed-up of approx. 36 with 64 cores (compared to 1 core) for A8 nodes. With A4 nodes, the scaling is not as desirable, probably also due to the lack of the low-latency Infiniband interconnect. The dynamic cloud-based cluster easily outperformed the on-premises cluster (see Fig. 4).

PETSc was run with two different matrices: *ruep* and *nord*. The *ruep* matrix is a 1162315×1162315 sparse real symmetric matrix with a total of $34.28 * 10^6$ non-zero values. The second matrix, *nord*, was generated from a thermal transient heat conduction simulation on a 3-dimensional Cartesian mesh of moderate size. It is a 398718×398718 sparse real symmetric matrix with a total of $1.1 * 10^6$ non-zero values.

In Fig. 4 the measurements for the PETSc trials are shown. From the diagram it is clear that the per-

formance on Azure nodes is close to ideal, while on the on-premises cluster is much worse, mostly because the memory bandwidth is insufficient for this core configuration and code. Because of the huge difference in the performance between A8 and A4 nodes, only cloud clusters with A8 nodes were taken into account for subsequent tests with ANSYS CFX on Microsoft Azure.

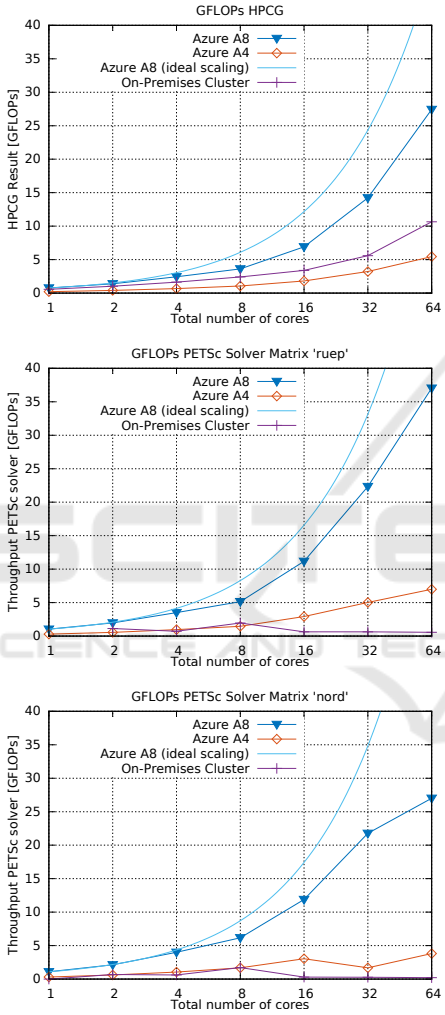

Figure 4: Performance in GFLOPs of HPCG (up) and PETSc (middle and bottom) solvers with different node sizes on Microsoft Azure and on the on-premises cluster.

4.3 ANSYS CFX Performance Study

We used ANSYS CFX as an example of a widely used tool for large-scale physical simulations. A set of independent simulations of a heat transfer simulation in a heat exchange system (subsequently called the *pipe* case).

In all the cases we modeled the steady state flows with residual targets of $1e-06$ and maximal number of iterations of 200.

4.3.1 Strong and Weak Scaling of the Heat Exchange Simulation

For the strong scaling tests, the pipe01–pipe08 problems were solved on different cluster configurations. The size of each pipe case was doubled and ranged from ca. 200 thousands elements to ca. 16 million elements. Note that the largest problem, pipe08, produced a memory overflow in a one core configuration and therefore the pipe08 strong scaling efficiency could not be computed. For the other cases, the strong scaling efficiency has been calculated according to formulas proposed by (Kaminsky, 2015). For the second largest problem, pipe07, the sequential run-time has been extrapolated, as 180 of 200 iterations could be calculated before the memory overflow and the increase of calculation time per iteration was approximately a constant fraction.

We could observe a good scaling for ANSYS CFX (see Fig. 5). The run-time is comparable, although the strong scaling efficiency on the cloud cluster is much better - this might be due to memory bandwidth limitations on the on-premises cluster (scaling from 1 to 6 cores greatly decreased efficiency). In the case of large cloud clusters (e.g. 128 cores on 32 nodes) much larger problems are needed to fully utilize their capabilities. As it can be seen from the figure the strong scaling efficiency for small problems rapidly decays. Even for larger problems the efficiency is worse with 256 cores, partly due to communication overhead.

A test with 64 with a total of 512 cores resulted in a MPI Broadcast timeout which might be due to the inefficient use of MPI collective operations or suboptimal broadcast algorithms (also reflected in the measured communication overhead). A hybrid approach (multi-threading together with MPI) and/or MPI tuning could circumvent such problems.

In order to measure weak scaling and weak scaling efficiency the problem size per processor core is kept constant while increasing the number of processor cores. The weak scaling efficiency is calculated according to formulas described in (Kaminsky, 2015).

In Fig. 6 the weak scaling measured on cloud clusters and the on-premises cluster is visualized. For ideal weak scaling a horizontal curve is observed, as the increase of problem data is proportional to the increase of cores. Here it is also evident that for large problems both clusters scale well.

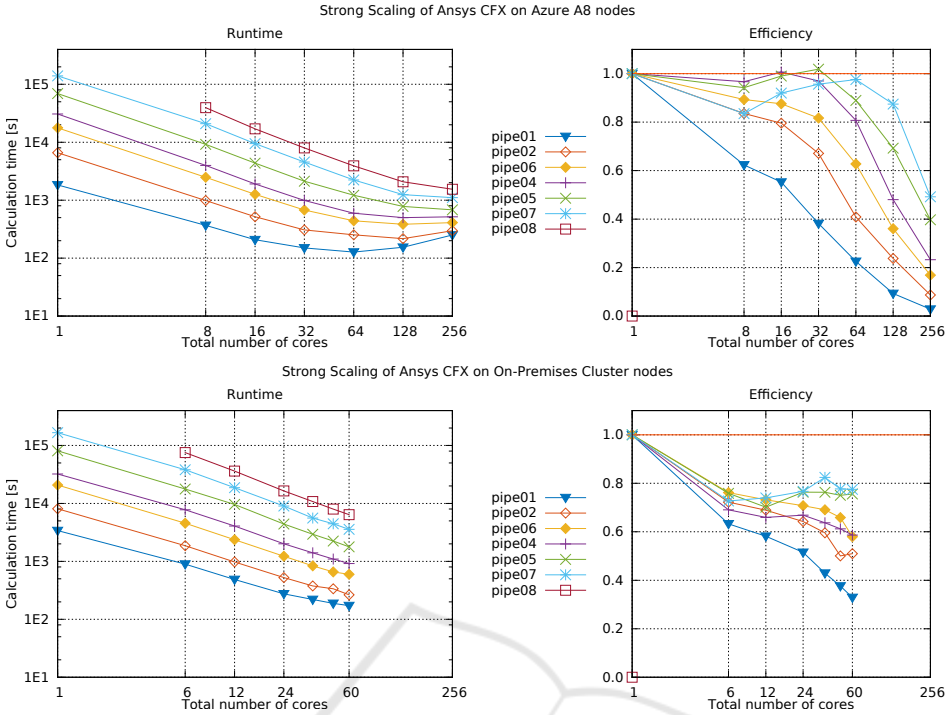

Figure 5: Run-time and strong scaling efficiency of ANSYS CFX for pipe01–pipe08 in the cloud (up) and on-premises cluster (bottom).

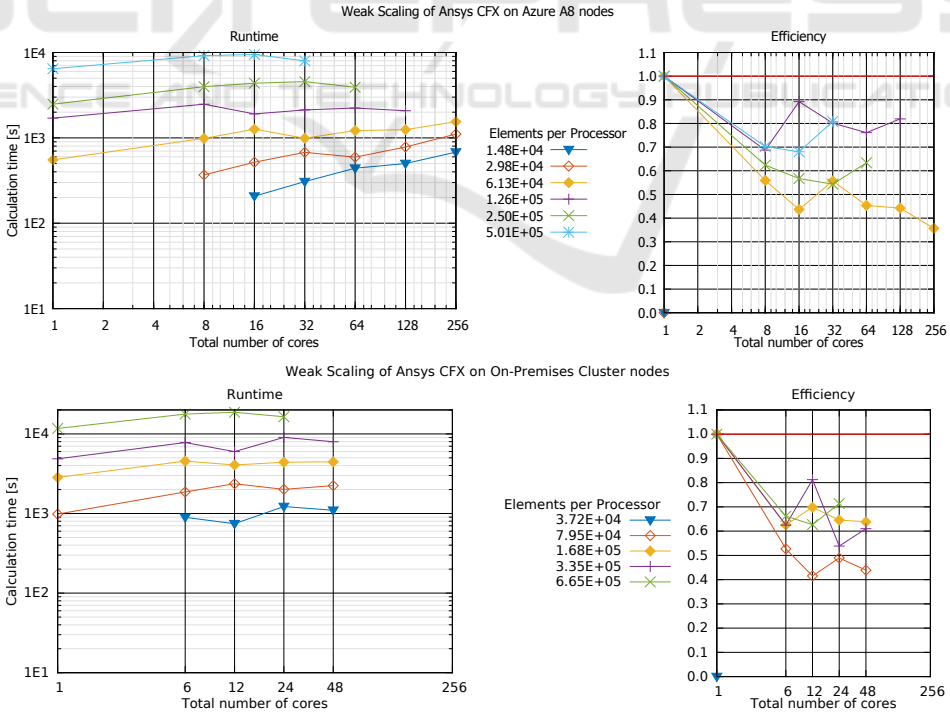

Figure 6: Run-time and weak scaling efficiency of ANSYS CFX in the cloud (up) and on-premises cluster (bottom).

5 CONCLUSION

We presented the SimplyHPC framework, a novel distributed framework for Microsoft Azure. The framework is composed of a light–weight API and a set of commandlets on top of that. The modules automate the complex deployment procedure that is necessary to run third-party applications from the local machine. Specially designed mechanism keeps track of running jobs in the cluster and a special service in the head node regularly checks the job status and uploads the results to the blob storage as soon as the job finished successfully. Since table and blob storage are distributed and mirrored, a high-availability and persistence of the job data is guaranteed. In contrary to HPC Pack from Microsoft, all unnecessary system components and services have been omitted and as a result the time needed to deploy the cluster has been reduced from more than an hour to a few minutes. We have demonstrated that the framework supports the deployment of both commercial and the open–source applications. Although we focused on fluid flow and numerical simulations the software is well suited to any research domain that uses software with a command line interface and supports MPI–based parallelization. We demonstrated that SimplyHPC can be used also in a scenario where both input and output data are usually massive. We also showed that for extreme scaling fine–tuning of the hardware configuration and MPI settings may be still necessary.

The paper clearly shows that scientists and engineers applications can benefit from the cloud if the calculations are too time–consuming for a local machine or a private cluster and that they can advantage of the competitive prices and the most recent hardware. Today, it is possible to deploy a 64–cores cluster for less than $30 and this price is likely to be lower when the reader reads this paper. Further, our scalability tests provide insights into the expected performance on Microsoft Azure up to 256 cores. SimplyHPC software is freely available at GitHub at https://github.com/vbaros/SimplyHPC.

ACKNOWLEDGEMENTS

The framework has been developed in Microsoft Innovation Center Rapperswil at HSR Hochschule Rapperswil, Switzerland. The scalability tests have been performed in Microsoft Azure as a part of Microsoft Research Grant No. $Azdem187T64934Y$. We would like also express our gratitude to Adrian Rohner, Roman Fuchs, Rita Rueppel and Vladimir Baros from HSR Hochschule Rapperswil for support and fruitful discussions.

REFERENCES

Ben Belgacem, M. and Chopard, B. (2015). A hybrid HPC/cloud distributed infrastructure: Coupling EC2 cloud resources with HPC clusters to run large tightly coupled multiscale applications. *Future Generation Computer Systems*, 42:11–21.

Che, Y., Xu, C., Fang, J., Wang, Y., and Wang, Z. (2015). Realistic performance characterization of cfd applications on intel many integrated core architecture. *The Computer Journal*.

Gentzsch, W. (2012). How cost efficient is hpc in the cloud? Technical report, Ubercloud.

Kaminsky, A. (2015). *Solving the World's Toughest Computational Problems with Parallel Computing*. Creative Commons Attribution.

Marozzo Fabrizio, Talia Dominico, T. P. (2013). *A cloud framework for big data analytics Workflows on Azure*. Advances in Parallel Computing. IOS Press.

Miroslaw, L., Baros, V., Pantic, M., and Nordborg, H. (2015). Unified cloud orchestration framework for elastic high performance computing on microsoft azure. In *Summary of Proceedings, NAFEMS World Congress*, page 216. NAFEMS Ltd.

Pham, L. M., Tchana, A., Donsez, D., de Palma, N., Zurczak, V., and Gibello, P.-Y. (2015). Roboconf: A hybrid cloud orchestrator to deploy complex applications. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 365–372.

Popinet, S. (2003). Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries. *Journal of Computational Physics*, 190(2):572–600.

Tomczak, T., Zadarnowska, K., Koza, Z., Matyka, M., and Miroslaw, L. (2013). Acceleration of iterative navier-stokes solvers on graphics processing units. *International Journal of Computational Fluid Dynamics*, 27(4-5):201–209.

Wong, A. K. and Goscinski, A. M. (2013). A unified framework for the deployment, exposure and access of HPC applications as services in clouds. *Future Generation Computer Systems*, 29(6):1333–1344.