

Compendium of Engineering Case Studies Using HPC Software Containers based on Docker 2015 - 2018



**Selection of Case Studies
based on UberCloud's HPC Docker Containers**

<https://www.TheUberCloud.com>



Welcome!

Applying UberCloud's HPC Containers to Engineering Applications in the Cloud

UberCloud is the online community and marketplace where engineers and scientists discover, try, and buy Computing Power as a Service, on demand. Engineers and scientists can explore and discuss how to use this computing power to solve their demanding problems, and to identify the roadblocks and solutions, with a crowd-sourcing approach, jointly with our engineering and scientific community. Learn more about the UberCloud at: <http://www.TheUberCloud.com>.

The goal of the UberCloud Experiments is to perform engineering simulation in the HPC cloud with real engineering applications in order to understand the roadblocks to success and how to overcome them. Our Compendiums are a way of sharing these results with the broader HPC and engineering community.

Our efforts are paying off. Based on the experience gained over the past six years, we have now increased the success rate of the individual experiments to 100%, as compared to 40% in 2013 and 60% in 2014.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux **Docker container** technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

For this Compendium we have selected 21 case studies from engineering cloud projects which all used UberCloud's novel HPC container technology based on Docker. The objective is to demonstrate the wide applicability of Docker based containers for really complex engineering and scientific applications, on different single- and multi-node cloud infrastructures.

We are extremely grateful for the support of our UberCloud experiments by **Hewlett Packard Enterprise and Intel**, and by our primary Media Sponsors **Digital Engineering and HPCwire**, and the invaluable case studies they supported, as well as this UberCloud Compendium series.

Wolfgang Gentsch and Burak Yenier
The UberCloud, Los Altos, CA, January 2019

*Please contact UberCloud at help@theubercloud.com before distributing this material in part or in full.
© Copyright 2019 TheUberCloud™. UberCloud is a trademark of TheUberCloud, Inc.*

The UberCloud Experiment Sponsors

We are very grateful to our Compendium sponsors **Hewlett Packard Enterprise and Intel**, our Primary Media Sponsor **Digital Engineering**, and to our sponsors ANSYS, Autodesk, Microsoft Azure, Nephoscale, NICE DCV, and Comsol Multiphysics GmbH. Their sponsorship allows for building a sustainable and reliable UberCloud community platform:



Big Thanks also to our media sponsors HPCwire, Desktop Engineering, Bio-IT World, scientific computing world, insideHPC, and Primeur Magazine for the widest distribution of this UberCloud Compendium of case studies in the cloud:



Table of Contents

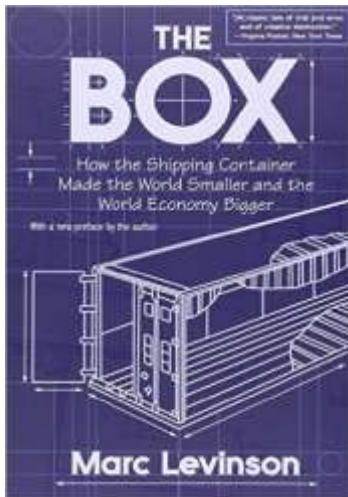
Introduction: Toward Ubiquitous HPC – With HPC Containers	06
Team 143: Clinical Cancer Genomics Pipeline on Amazon AWS Department of Human & Medical Genetics, Vilnius University Genomics pipeline open source software	10
Team 156: Pulsatile flow in a Right Coronary Artery Tree on Amazon AWS Carnegie Mellon University OpenFOAM CFD software	15
Team 159: Aerodynamic Study of an Airfoil in UberCloud’s OpenFOAM Container Praveen Bhat, Technology Consultant, INDIA ESI Group with OpenFOAM	19
Team 169: Blood Flow through Cardiovascular Medical Device Using OpenFOAM Praveen Bhat, Technology Consultant, INDIA OpenFOAM from CFD Support, Czech Republic	25
Team 177: Combustion Training in the Cloud de Jong Group, Energy and Environmental Technologies, The Netherlands Ferry Tap, Dacolt (now AVL List), The Netherlands	31
Team 181: Predicting Barehull Containership Resistance in the CPU 24/7 Cloud Glosten Inc., USA NUMECA FINE/Marine software	33
Team 182: CFD Modelling and Optimization of Transformers in the Azure Cloud ABB Research Lab, North Carolina, USA OpenFOAM CFD software	36
Team 183: Radial Fan CFD Simulation in the Azure Cloud CFD Support Ltd., Czech Republic Turbomachinery CFD software	40
Team 186: Airbag simulation with ANSYS LS-DYNA on Azure Praveen Bhat Technology Consulting, INDIA ANSYS LS-DYNA	44
Team 190: CFD Simulation of Airflow within a Nasal Cavity on Azure Charite University Hospital Berlin, Germany Siemens/CD-Adapco STAR-CCM+	49
Team 193: Implantable Planar Antenna Simulation on Nephoscale Cloud Ozen Engineering, Inc. Sunnyvale, California	55

Team 195: Simulation of Impurities Transport in a Heat Exchanger Using OpenFOAM	59
Eugeny Varseev, ROSATOM-CICE&T, Russia CFD Support, <i>OpenFOAM in Box</i> , Czech Republik	
Team 197: Studying Drug-induced Arrhythmias of a Human Heart on Advania	63
Stanford University Dassault/SIMULIA Abaqus software	
Team 199: Peptide Benchmark Using Molecular Dynamics on Amazon AWS	68
National Renewable Energy Laboratory LAMMPS open source molecular dynamics	
Team 200: Simulation of Neuromodulation in Schizophrenia on Advania	72
NIMHANS Institute for Medical Health, India Dassault/SIMULIA Abaqus software	
Team 201: Maneuverability of a KRISO Container Ship Model in the Cloud	78
Xin Gao, Technical University of Berlin NUMECA FINE/Marine	
Team 203: Aerodynamic Study of a 3D Wing Using ANSYS CFX	87
Praveen Bhat, Technology Consultant, India ANSYS CFX	
Team 204: Aerodynamic Simulations using MantiumFlow in Advania Data Centers	93
MantiumCAE OpenFOAM and customized modules	
Team 205: Vehicle Crash Simulation in the Opin Kerfi Cloud	97
Praveen Bhat Technology Consulting, India ANSYS LS-DYNA software	
Team 206: Establishing the Design Space of a Bioreactor on Microsoft Azure	103
Healthcare Group at ANSYS, Inc. ANSYS Fluent software	
Team 211: Deep Learning for Predicting Steady-State Fluid Flow in the Advania Cloud	110
Renumics, Germany OpenFOAM CFD and Renumics Deep Learning software	
Join our free and voluntary UberCloud Experiment	117

Toward Ubiquitous High Performance Computing

Passing HPC into the hands of every engineer, with novel software containers

We've never been so close to ubiquitous computing for researcher and engineer, accessible everywhere. High-performance computing (HPC) continue to progress, but the next big step toward ubiquitous HPC is coming from software container technology based on **Docker**, dramatically facilitating software packaging and porting, ease of access and use, and drastically simplify software maintenance and support.



“In April 1956, a refitted oil tanker carried 58 shipping containers from Newark to Houston. From that modest beginning, container shipping developed into a huge industry that made the boom in global trade possible.”

Marc Levinson writes in *“The Box: How the Shipping Container Made the World Smaller and the World Economy Bigger”*, how containers transformed economic geo-graphy - a great analogy to today's software containers and their growing importance for science & engineering, and for the whole application life cycle: from design, coding, testing, to software release, distribution, access and use, support and maintenance, and especially for engineers and scientists.

Towards Ubiquitous HPC

Now translating this into 'Ubiquitous HPC'. Very simplified HPC technology is split into two parts: software and hardware; both today are immensely complex in themselves; and their mutual interaction is highly sophisticated. For CAE to be ubiquitous Xerox PARC's Mark Weiser suggests to make it disappear into the background of our (business) lives; from the end user's point of view. Indeed, in the last decade, we were able to make a big step towards reaching this goal: we made access and use of engineering codes 'relatively' easy by developing user-friendly user interfaces, with its trends towards what some people call 'appification'; and we abstracted the application layer from the physical architecture underneath, through server virtualization with Virtual Machines (VMs). This great achievement came with great benefits especially for the IT folks – and for the end-users too: such as provision servers faster, enhance security, reduce hardware vendor lock-in, increase uptime, improve disaster recovery, isolate applications and extend the life of older applications, and help move things to the cloud easily. So, with server virtualization we came quite close already to ubiquitous computing . . .

Finally – Ubiquitous HPC – With Engineering Application Software Containers

Server virtualization did not really gain a foothold in HPC, especially for highly parallel HPC applications requiring low latency and high-bandwidth inter-process communication. And multi-tenant servers, with VMs competing among each other for hardware resources such as I/O, memory, and network, are often slowing down HPC application performance.

Because VM's failed to show presence in HPC (at least so far), challenges of software distribution, administration, and maintenance kept CAE systems locked up in closets, available to only a select few. In fact, the US Council of Competitiveness estimates that only about 5% of all engineers are using high-performance servers for their CAE simulations, the other 95% just use their workstations.

In 2013, **Docker Linux Containers** saw the light of day. The key practical difference between Docker and VMs is that Docker is a Linux-based system that makes use of a userspace interface for the Linux kernel containment features. Another difference is that rather than being a self-contained system in its own right, a Docker container shares the Linux kernel with the operating system running the host machine. It also shares the kernel with other containers that are running on the host machine. That makes Docker containers extremely **lightweight**, and well suited for HPC, in principle. Still it took us at UberCloud about a year to develop – based on micro-service Docker container technology – the macro-service production-ready counterpart for HPC, plus enhancing and testing it with a dozen of HPC applications and with engineering workflows, on about a dozen different single- and multi-node cloud resources. These high-performance interactive software containers, whether they be on-premise, on public or on private clouds, bring a number of core benefits to the otherwise traditional HPC environments with the goal to make HPC widely available, ubiquitous:

Packageability: Bundle applications together with libraries and configuration files:

A container image bundles the needed libraries and tools as well as the application code and the necessary configuration for these components to work together seamlessly. There is no need to install software or tools on the host compute environment, since the ready-to-run container image has all the required components. The challenges regarding library dependencies, version conflicts, configuration challenges disappear, as do the huge replication and duplication efforts in our community when it comes to deploying HPC application software.

Portability: Build container images once, deploy them rapidly in various infrastructures:

Having a single container image makes it easy for the workload to be rapidly deployed and moved from host to host, between development and production environments, and to other computing facilities easily. The container allows the end user to select the appropriate environment such as a public cloud, a private cloud, or an on-premise server. There is no need to install new components or perform setup steps when using another host.

Accessibility: Bundle tools such as SSH into the container for easy access:

The container is setup to provide easy access via tools such as VNC for remote desktop sharing. In addition, containers running on computing nodes enable both end-users and administrators to have a consistent implementation regardless of the underlying compute environment.

Usability: Provide familiar user interfaces and user tools with the application:

The container has only the required components to run the application. By eliminating other tools and middleware, the work environment is simplified and the usability is improved. The ability to provide a full featured desktop increases usability (especially for pre and post processing steps) and reduces training needs. Further, engineering software containers can be used together with a resource manager such as Slurm or Grid Engine, increasing the usability even further by eliminating many administration tasks. In addition, the lightweight nature of the HPC container suggests low performance overhead. Our own performance tests with real applications on several multi-host multi-container systems demonstrate that there is no significant overhead for running high performance workloads as an engineering application container.

Security: UberCloud understands that you need to know your data is secure:

Engineers care about the safety of their designs. Engineering applications and data are of strategic importance to an organization as a source of competitive advantage and innovation. With UberCloud you run in a **private compute environment**. We enable encryption for data transfers, remote access,

and data storage. Your cloud resources are **dedicated** to you, and they are **not shared**. When your processing is finished, your environment is deleted.

We only use professionally managed cloud infrastructures, with strong physical security controls including biometric entry authentication and armed guards. You have fine grained access control which gives you the flexibility to define your security requirements. Your IT team tells us what firewall rules and authentication methods they need. We implement them. Done.

Table: Why are HPC software containers becoming more and more popular?

#	Customer	Software Vendor (ISV)	Resource Provider
1	Full control of the simulation environment for real-time monitoring, interactive access and batch use	Familiar user interface, providing full desktop UI with HD-Quality pre-post processing	UberCloud's container technology maintains bare-metal performance
2	Software containers provide fully secure environment and add extra security by limiting connections only to the container not the node	Software containers provide fully secure environment and add extra security by limiting connections only to the container not the node	Software containers provide fully secure environment and add extra security by limiting connections only to the container not the node
3	Data segregation by keeping data in dedicated hosts that are not shared with anyone else	Data segregation by keeping data in dedicated hosts that are not shared with anyone else	Data segregation by keeping data in dedicated hosts that are not shared with anyone else
4	Fast and on-demand access to simulation environment	Fast and on-demand access to simulation environment	Full portability with unique containers easily deployable to the Resource Provider's environment
5	Familiar user interface, providing full desktop UI with HD-Quality pre-post processing	UberCloud's container technology maintains bare-metal performance	Familiar user interface, providing full desktop UI with HD-Quality pre-post processing
6	Allows complex engineering workflows by customization and combination of software from different ISVs	Building complete SaaS Offering for ISVs	Automatic cluster sizing with CycleCloud or UniCloud
7	Single point of contact for customers' software and hardware support	Full control of the simulation environment for real-time monitoring, interactive access and batch use	Single point of contact for customers' software and hardware support
8	UberCloud's container technology maintains bare-metal performance	Runs everywhere – on public, private, and hybrid cloud	ISV Certified

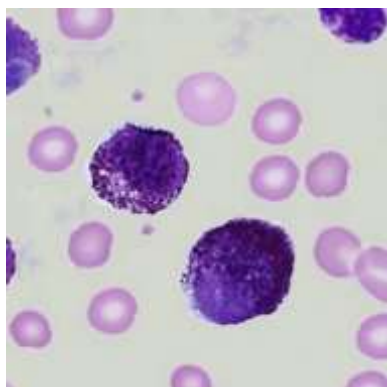
9	ISV Certified	Automatic cluster sizing with CycleCloud	Full control of the simulation environment for real-time monitoring, interactive access and batch use
10	Runs everywhere – on public, private, and hybrid cloud	Single point of contact for customers’ software and hardware support	Runs everywhere – on public, private, and hybrid cloud
11	Real-time collaboration between globally distributed team members	Full portability with unique containers easily deployable to any environment	Fast and on-demand access to simulation environment
12	Option to conserve software environments for continuity and repeatability of past simulations	Option to conserve software environments for continuity and repeatability of past simulations	Building complete SaaS Offering for ISVs
13	Automatic cluster sizing with CycleCloud	Real-time collaboration between globally distributed team members	Allows complex engineering workflows by customization and combination of software from different ISVs
14	Based on Open Source and industry standard	Allows complex engineering workflows by customization ability to combination complementary apps	Based on Open Source and industry standard
15	Full portability with unique containers easily deployable to any environment	Based on Open Source and industry standard	Conserving sw environments for continuity & repeatability of past simulations
16			Real-time collab. between globally distributed team members

During the past six years UberCloud has successfully built HPC application containers for GROMACS, ANSYS, LS-Dyna, CD-adapco, COMSOL, NICE, Numeca FINE/Marine, FINE/Turbo, OpenFOAM, PSPP, Red Cedar HEEDS, Scilab and more. These application containers are now running on cloud resources from Advania, Amazon AWS, CPU 24/7, Microsoft Azure, Nephoscale, OzenCloud, and others. Before we used our application containers, our UberCloud users’ projects took on average about 3 months. When we started to apply containerization in January 2015 (with project #143), our users’ cloud projects took only 3 days on average which speaks for itself !

Together with recent advances in application software and HPC hardware technologies, the advent of lightweight pervasive, packageable, portable, scalable, interactive, easy to access and use software containers running seamlessly on workstations, servers, and any cloud, is bringing us ever closer to what Intel – in 2012 already – called the democratization of high performance technical computing. We are arriving at the age of Ubiquitous Computing where computing “technology recedes into the background of our lives.”

Team 143:

Open Source Clinical Cancer Genomics Pipeline in the Cloud



“UberCloud container images were easily ported between environments by the Operations Engineer.”

MEET THE TEAM

End-User: Erinija Pranckeviciene, Bioinformatics Research Scientist, Department of Human & Medical Genetics, Vilnius University

Resource Provider: [Amazon AWS](#)

HPC Cloud Experts: Fethican Coskuner, Operations Engineer, [UberCloud](#).

USE CASE

Rapid progress in next generation sequencing is paving the way to individualize cancer treatment based on sequencing of the cancer genome. This process is still evolving and validated open source pipelines to process the enormous amounts of data involved are not available to everyone interested in this field.

Identifying informative genomic variants – potential disease-causing candidates – through the use of next generation sequencing (NGS) data is becoming a routine diagnostic tool. In cancer, exomes of tumor and normal tissues aligned to the reference genome serve as primary data source to start identifying these genomic variants for a better understanding of the disease.

Software programs used to align NGS data to the reference genome are governed by user-defined parameters and map the short sequenced reads to the reference genome with varying accuracy [4]. Most of the alignment tools require a long time to obtain accurate mappings of the short reads to the reference genome.

Decisions as to which of the alignment tools have to be included into the NGS data processing pipeline depend on many factors such as ease-of-use of tools, their public availability, and their ability to align NGS data from various sequencing platforms (Illumina and AB Solid). Among those factors the time required to accomplish the alignment plays a very important role. The time required for the alignment program to complete the mapping, depends on the size of the NGS data set and scales with available computational resources.

The main goal of this project was to start preparing guidelines on the optimal choices of open source alignment tools to be included into the NGS data processing pipelines by matching these tools to the available computational resources. We started by estimating how much time the alignment tools needed to complete the task of alignment of the cancer data-set to the human reference genome by running these tasks on various configurations of high-performance computing (HPC) resources.

At this stage two aligners were chosen: Mapping and Assembly with Qualities (MAQ 0.6.6 and 0.7.1) [1] and Short Read Mapping Package (SHRIMP 2.2.3) [2].

MAQ is one of the most accurate aligners and supports gaped alignments of paired end reads. However, MAQ was programmed in such way that it can't take advantage of available multiple-core processor architectures.

SHRIMP is an exhaustive mapping tool because it finds multiple candidate alignment locations (CALs) and then selects the best hits out of those CALs. SHRIMP can fully utilize and take advantage of the available memory and multiple-core processor architectures. The computational resources used in this project are listed in Table 1.

Table 1: Computing Resources.

A) AWS	B) BARE METAL #1	C) BARE METAL #2	D) WORKSTATION AT THE DEPT. OF HUMAN AND MEDICAL GENETICS, VILNIUS UNIVERSITY #3
CPU: Intel® 8 CPU cores (Model is unknown) RAM: 68GB	CPU: Intel Xeon CPU E5-2650, 32 cores RAM: 32GB	CPU: Intel Xeon(R) CPU E5620, 16 cores RAM: 144GB	Allocated resource: CPU: AMD Opteron (tm) Processor 6172, 12 cores RAM: 98GB
OS: Ubuntu 12.10 LTS Server MAQ: 0.6.6	OS: Ubuntu 12.04.2 LTS Server MAQ: 0.6.6 SHRIMP: 2.2.3	OS: Ubuntu 12.04.4 LTS Server MAQ: 0.7.1 SHRIMP: 2.2.3	OS: CentOS 5.10 MAQ: 0.7.1 SHRIMP: 2.2.3

To enable portability of the computation environment (including the aligners, scripts, configuration settings) Linux container images were used. The same image was used in the AWS, Bare Metal #1, and Bare Metal #2 test. The Linux container images were easily ported between environments by the Operations Engineer.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

PROJECT OBJECTIVES

Objective 1 – Document how much time it took for MAQ and SHRiMP to align 2 million 75 base-pair (bp) length paired end sequence reads of cancer exome data [3] to the human reference genome. The programs were tested on several HPC configurations summarized in Table 1.

Objective 2 – To compute a complete mapping of the mast-cell leukemia exome data (tumor and germline available in Illumina Genome Analyzer FASTQ format) [3] to the Human Genome Build 37 (GRCh37 hg19).

RESULTS

Total number of paired end reads in a whole sample consisted of 32,971,877 reads in the germline exome and of 37,495,826 reads in the tumor exome. To compare chosen alignment programs in terms of their execution time on different HPC configurations, 2 million paired end reads of the germline exome was used. Durations in which MAQ completed alignments of 2 million reads on tested computing resources are shown in Table 2. The shortest duration was achieved on the HPC configuration C. Next, MAQ was tested by using [UberCloud](#) Linux [container](#) images on HPC architecture C. The alignments of 2 million paired end reads were executed simultaneously on 4 nodes. Each completed in 2 hours and 5 minutes.

SHRiMP can fully utilize multiple-core processor architectures. We were interested in how much the alignment time could be reduced by using the aligner, which takes advantage of available multi-core computing resources. SHRiMP can use as many threads as set by a parameter. Durations in which SHRiMP aligned 2 million reads on different HPC architectures are shown in Table 2.

To summarize the comparative results, we note that MAQ completed the task in 75 minutes and SHRiMP completed the same task in 40 minutes by using 16 threads. Both of them completed the alignment of 2 million paired end reads within similar time range.

Table 2: Running times of the alignment tasks.

ALIGNMENT TASK	A) AWS	B) BARE METAL #1	C) BARE METAL #2	D) WORKSTATION AT THE DEPT. OF HUMAN AND MEDICAL GENETICS, VILNIUS UNIVERSITY #3
MAQ (2 mln reads) version 0.6.6	cr1.8xlarge more than 2 days experiment stopped			
MAQ (2 mln reads) version 0.7.1	m1.medium 2 h 20 min	1h 30 min	1 h 13 min	4 h 30 min
SHRiMP (2 mln reads) using 1 thread using 16 threads using 24 threads			3 h 20 min 1 h 40 min 38 min	

We estimated that the time required for MAQ to complete the mapping of 2 million reads on the architecture C within the virtualization layer takes a little over two hours. Based on this estimate, we predicted that time, required to map exomes of germline and tumor divided into 36 parts of 2 million reads each, should take approximately $36 \times 2 = 72$ hours. By using 4 computing nodes

simultaneously for this task the mapping should complete in 18 hours. The estimated time for SHRiMP using 24 threads to map 2 million reads takes about 40 minutes. Based on this estimate, we predicted that SHRiMP will align the whole sample in 25 hours. A summary of the actual time spent in the mapping of whole sample on the HPC architecture C is shown in Table 3.

We used 24 threads in mapping with SHRiMP. The mappings were performed in batch on a single node. This mapping task completed in 29 hours, which is more or less consistent with previous expectation of 25 hours that were predicted for the whole sample based on the time estimate of mapping 2 million reads (see Table 2).

Mapping by MAQ was performed using UberCloud Linux containers on four nodes. Two nodes were used to map the germline and another two nodes were used to map the tumor sample. Alignment of the whole sample by MAQ completed in 35 hours, which is almost twice as long as expected.

There are several reasons why MAQ had prolonged mapping time. The first is that, in general, reads have different sequence complexity, which influences mapping time. The average actual mapping time of a part of 2 million reads was approximately three hours. The second reason was unexpected interruptions in a batch mapping because of unexpected symbols in some parts of the data. The second reason is minor – the data was cleaned and processed again.

Table 3: Alignment of the whole exome of germline and tumor by using HPC configuration C.

Alignment program	Germline sample 0-16 parts x 2million reads	Tumor 0-18 parts x 2million reads
MAQ version 0.7.1 utilized 4 nodes in parallel	In batch: Node1 (part 0-6) From Jun 7 21:02 Till Jun 9 07:42 Node2 (part 12 - 14) From Jun 9 02:38 Till Jun 9 11:57 Realigned: Parts 7,8,9, 10,11,14,15,16	In batch : Node3 (part 0-6) From Jun 7 21:04 Till Jun 9 10:42 Node4 (part 10-16) From Jun 7 21:04 Till Jun 9 07:18 Realigned: Parts 7,8,9 17,18
SHRiMP ver 2.2.3 using 24 threads on 1 node	From Jun 7 22:02 Till Jun 8 11:59 Total 14 hours	From Jun 8 12:57 Till Jun 9 04:00 Total 15 hours

CONCLUSION AND RECOMMENDATIONS

The team gained two particularly useful insights from this project:

Some software programs are created in a such a way that they can't take advantage of modern multiple core architectures. Therefore, a virtualization approach was utilized to perform mapping with simple alignment program MAQ simultaneously on four nodes. Running simple programs in virtualization layers on HPC resources proved to be a very productive use of those resources resulting in a considerable reduction of execution times of simple processes. In NGS data analysis, numerous useful software programs exist that can't use modern HPC architectures. Using these

programs to process big data is challenging because of the time required to obtain results. One possible solution to this challenge is to use the [UberCloud](#) Linux [containers](#) on available HPC architecture.

The predicted mapping times resulting from estimates of mapping times computed on small randomly selected sample of reads are only approximate. This most likely occurs because sequence complexity impacts finding a candidate alignment location of the read in the reference genome.

Alignment of the short reads to the reference genome in NGS exome analysis pipeline is only a single intermediate step in obtaining information on which genomic variants are present in the exome. A second step after the alignment is search and identification of those genomic variants. This step is computationally demanding as well. It would be beneficial to assess a computational intensity and create resource usage guidelines for genomic variant identification tools such as Genome Analysis Toolkit (GATK), Samtools and MAQ.

REFERENCES

1. Li, H, Ruan, J, Durbin, R (2008). "Mapping short DNA sequencing reads and calling variants using mapping quality scores." *Genome Res.*, 18, 11:1851-8.
2. David, M, Dzamba, M, Lister, D, Ilie, L, Brudno, M (2011). "SHRiMP2: sensitive yet practical SHort Read Mapping." *Bioinformatics*, 27, 7:1011-2.
3. Spector, MS, Iossifov, I, Kritharis, A, He, C, Kolitz, JE, Lowe, SW, Allen, SL (2012). "Mast-cell leukemia exome sequencing reveals mutation in the IgE mast-cell receptor b chain and KIT V654A." *Leukemia*, 26(6):1422-5. PMID:22173243
4. Hatem, A, Bozda, D, Toland, AE, Catalyurek, UV (2013). "Benchmarking short sequence mapping tools." *BMC Bioinformatics*, 14:184, p.1-25.

Team 156:

Pulsatile flow in a Right Coronary Artery Tree



“The UberCloud system was found to be far more user friendly than the NSF XSEDE supercomputing resources, in terms of both simplicity and ease of access.”

MEET THE TEAM

End User – Prahlad G. Menon, PhD, Dept. of Electrical & Computer Engineering, Carnegie Mellon University

Resource Providers – [Amazon AWS](#)

Software Provider – [UberCloud](#) OpenFOAM [container](#) with OpenFOAM and ParaView.

USE CASE

Modeling of vascular hemodynamics has become increasingly popular for the assessment of the underlying patient-specific biomechanical traits of vascular disease. This modeling approach typically begins with the three-dimensional (3D) segmentation and reconstruction of a smooth surface model of the vascular region of interest from patient-specific medical image volumes obtained as a series of tomographic slices using either magnetic resonance imaging (MRI) or computed tomography (CT) imaging. The resulting 3D surface model is used to generate a discretized volume mesh for the purpose of analysis of flow using computational fluid dynamics (CFD) solution techniques.

In this study, blood flow inside a patient-specific right coronary artery tree (see Figure 1), including one inlet and a total of 7 outflow branches, was studied computationally under realistic unsteady flow conditions after segmentation from tomographic MRI slice images obtained across the whole human body of a male volunteer, at 2mm intervals.

METHODS

A finite volume mesh consisting of 334,652 tetrahedral elements / cells was prepared in the mesh building toolkit, GAMBIT (Ansys). This mesh was then passed over as input to the icoFoam solver in OpenFOAM – a free, open source CFD software package – to solve for hemodynamics.

Hemodynamics (meaning literally "blood flow, motion and equilibrium under the action of external forces") is the study of blood flow or circulation. It explains the physical laws that govern the flow of blood in the blood vessels under pulsatile inflow conditions. A realistic right coronary artery inflow

waveform was obtained from the scientific literature based on reports from catheterization studies for the purpose of this CFD simulation study (see Figure 2).

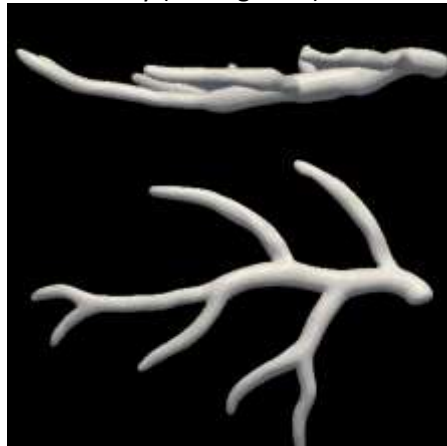


Figure 1: 3D reconstruction of the studied patient-specific human right coronary artery tree.

The solution was obtained assuming a plug-flow inlet velocity profile, zero-pressure outlet boundary conditions and rigid, impermeable, no-slip arterial wall boundaries. The choice of equal, zero-pressure outflow boundary conditions resulted in a naturally distribution of blood flow to the seven outlets of the geometry, based on their respective intrinsic geometry-based resistance pathways.

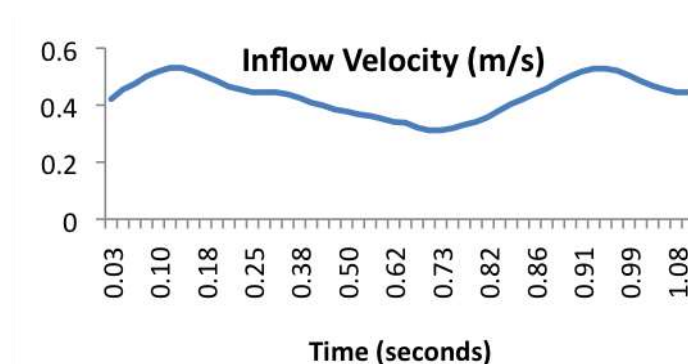


Figure 2: Realistic pulsatile right coronary artery inflow waveform considered in this study.

Blood was modeled as a Newtonian fluid with constant density (1060 kg/m³) and viscosity (0.00371 Pa.s). One full cardiac cycle was simulated with a time step of 0.00001 seconds, with first order implicit time integration and second order Gaussian integration based finite volume spatial discretization with a linear cell-to-face center interpolation scheme. The generalized geometric-algebraic multi-grid (GAMG) solver was chosen for this study for its properties of being faster than most standard solvers. This is due to its strategy of first generating a flow solution for velocity and pressure on a coarser mesh using a method of agglomeration of cells, and then mapping the coarse solution onto the full (i.e. fine) mesh to obtain a more accurate solution. The solution was obtained using the remote UberCloud multi-core computing framework – including [UberCloud's](#) new software containers – after first parallelizing the problem automatically on to 16 cores using OpenFOAM's computational domain decomposition and then submitting the job (including all necessary input files) via a secure shell (SSH) client.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on

Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

RESULTS AND DISCUSSION

Pulsatile flow results were examined using ParaView, running on a remote visualization machine. Fig. 3 shows an instantaneous snapshot of flow in the coronary artery tree under peak flow conditions.

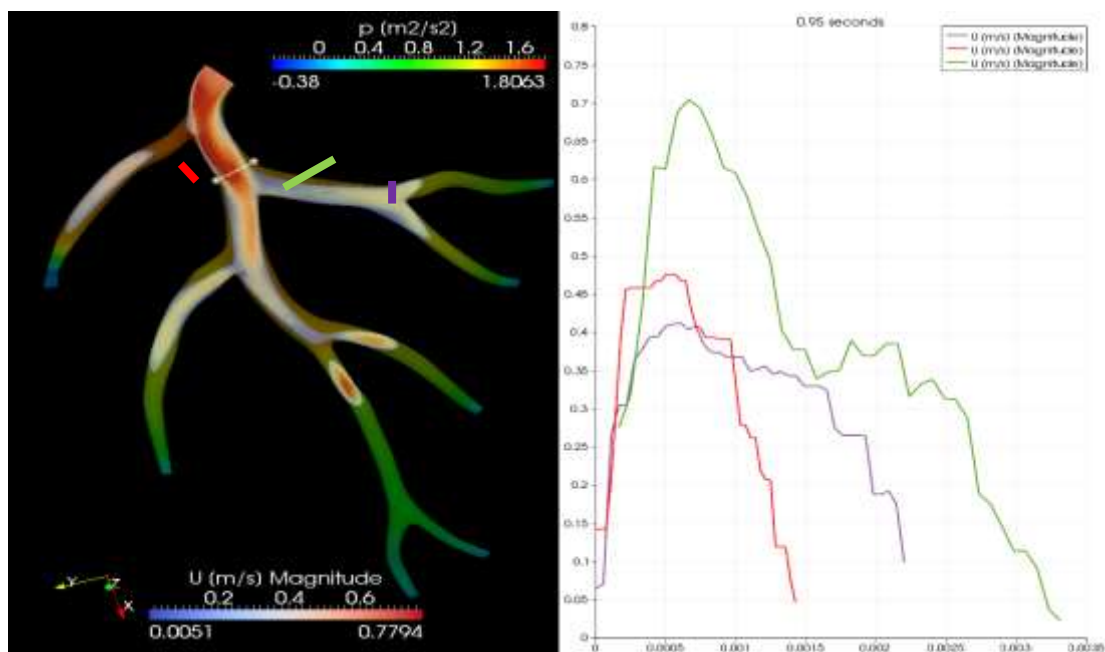


Figure 3: Visualization of hemodynamics at the instantaneous peak-flow instant, in ParaView. The full time-resolved simulation result is available for viewing at:

http://www.youtube.com/watch?v=hcS_k9xflo

Shown on the left of Figure 3, the velocity through the inlet is visualized through a longitudinal slice (blue to red color-scale) superimposed with a translucent rendering of the arterial wall colored by the arterial pressure field (green to red color-scale). The units of velocity are meters per second (m/s), whereas pressure is plotted in m^2/s^2 , which is equivalent to Pascals per unit density units (i.e. as Pascals per unit density units = $(kg \cdot m/s^2) / (kg/m^3) = m^2/s^2$; where density = 1060 kg/m^3). On the right of Figure 3, velocity profiles through three different diametric lines across the geometry (marked in the figure on the left), are shown. The x-axis of the velocity profile plot is distance measured along the diameter of vessel segment (in SI units of meters), across which the velocity profile is visualized. Realistic pulsatile flow profiles which resembled expectations of Womersley flow profiles were possible to observe over the simulated cardiac cycle.

Finally, the average simulation speed under 16-core parallelism was analyzed as a function of clock time taken per unit of simulated pulsatile flow time. This worked out to be ~34 clock hours per simulation second on the UberCloud Nephoscale system, as illustrated in Figure 4.

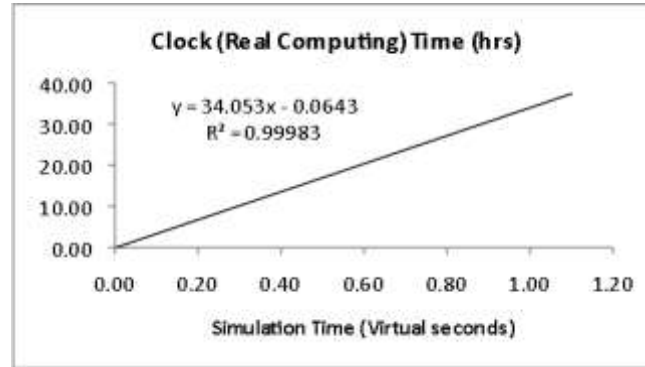


Figure 4: Illustration of the relationship between clock time (in hours) and simulated pulsatile flow time (in seconds) in the coronary artery.

BENEFITS

The UberCloud system was found to be far more user friendly – in terms of both simplicity and ease of access (via SSH) – than the National Science Foundation (NSF) XSEDE resources, which I use on a daily basis simply because the OpenFOAM cases ran without any issues with MPI libraries. On the other hand, access to XSEDE resources is less expensive than commercial cloud resources.

Parallel scalability of the simulated OpenFOAM problem was found to be quite good on the remote computing resource. A 2- to 3-fold speed improvement was noted using 16 core parallelism on the remote UberCloud machine in contrast with an equivalent local simulation run on just 4 parallel cores of a local quad-core machine.

CONCLUSIONS AND RECOMMENDATIONS

First, I like the ease of access and simplicity of the UberCloud system and would consider it as a preferred computation system if up to 128 cores were available for parallel simulation. CFD simulations generate large volumes of data and therefore it would be ideal if the UberCloud system could facilitate remote visualization of results without having to download large amounts of data to a local machine for further analysis (Now the [UberCloud Container](#) comes with ParaView and remote viz integrated).

I used SSH to download and locally process the results. This was accomplished with consistent upload/download speeds of ~700 KBps – which is pretty good. While it is possible to get 10+ MBps, inside a local network 700 MBps is a good speed for transferring data to a remote server. Availability of ParaView as a visualization software along with VNC (remote desktop) access capabilities would enhance ease of uploading / downloading simulation data and simultaneously facilitate remote processing as well as visualization of simulation results.

Second, it is very important to have sudo / root privileges for some installation and compilation procedures in the interest of customizing boundary conditions and solver settings in OpenFOAM. For example, if you were to use a 3rd party library like *Bison* for a boundary condition implementations in OpenFOAM, at the present time you would have to contact the UberCloud administrator in order to install this library and subsequently add access to this library to your Linux user environment. Most computational specialists require compiling and using their own code, and install 3rd party libraries often. Therefore, the lack of sudo-user privileges to install software (e.g., using the apt-get command) or customize environment variables may be seen as a limitation to an advanced user of the UberCloud environment, when the user is not the administrator (i.e. 'root').

Team 159: Aerodynamic Study of an Airfoil in the UberCloud OpenFOAM Container



“The combination of open source OpenFOAM and UberCloud HPC enables efficient, effective, and easy performance of complex engineering simulations.”

MEET THE TEAM

End-User/CFD Expert – Praveen Bhat, Technology Consultant, INDIA

Software Provider – [ESI Group](#) with OpenFOAM and [UberCloud Container](#)

Resource Provider – Nephoscale cloud resource provider.

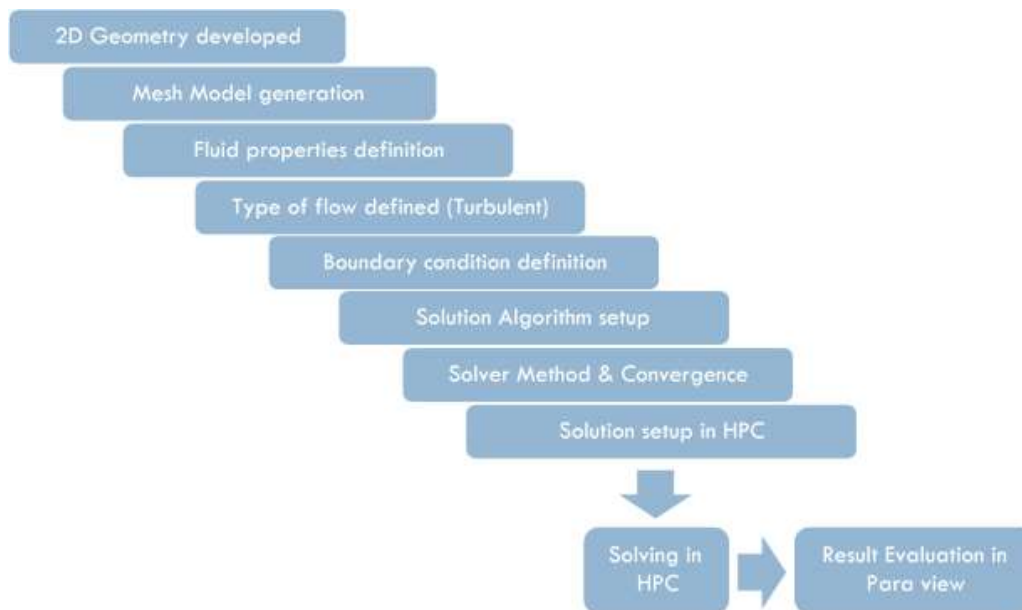


Figure 1: Model Setup flowchart.

USE CASE

The aerodynamic study on the 2D airfoil was performed with the incompressible air flow around a 2D airfoil. The model setup included the geometry preparation where a selected region was modelled to represent the surrounding air volume with the airfoil profile at the center. The airfoil profile needed to be accurately modelled to capture the variation in the airflow pattern around the airfoil. The model setup was done using the open source software OpenFOAM. The OpenFOAM software is embedded in an UberCloud Container located in the Nephoscale cloud facility.

The main objective of this project was to experience the ease-of-use of the [UberCloud OpenFOAM container](#) and to evaluate HPC performance with respect to the accuracy of result prediction as well as the solution time and resource utilization.

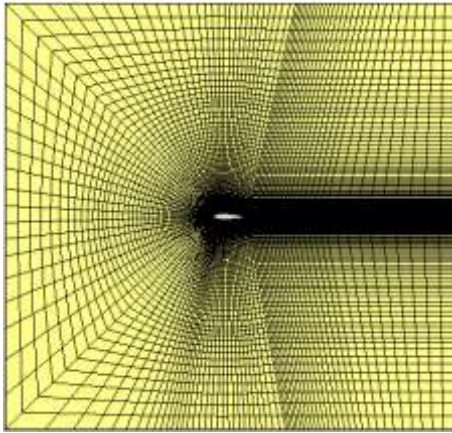
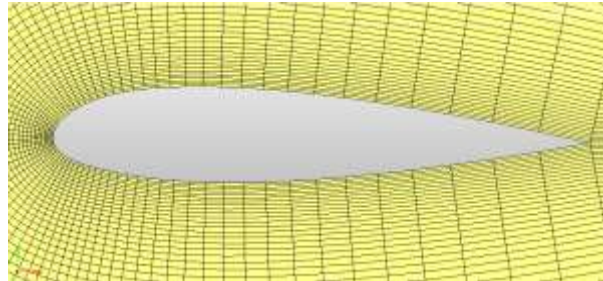


Figure 2: Mesh model for the aerofoil.



UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux [Docker container](#) technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

Process Overview

The meshing density is very fine around the airfoil and also along the path of the trailing edge. The meshes were modelled coarser as they moved away from the airfoil region. The coarseness of the mesh increased near the air volume boundary (air inlet and outlet). The following details describe the steps in the simulation model setup using OpenFoam:

1. The Finite Element mesh model was generated followed by the fluid properties definition. The entire volume surrounding the airfoil is air which is considered incompressible in nature.
2. The fluid properties are defined as Newtonian fluids with a linear relationship between the shear stress (due to internal friction forces) and the rate of strain of the fluid.
3. Atmospheric air is turbulent in nature and there is a transition phase from turbulent to laminar in the region near the airfoil. Because of this transition, the mesh model needs to be refined accurately near the airfoil region along with defining the turbulence behavior of the air which is captured through a Spalart – Allmaras turbulence model.
4. The next section in the model setup was defining the model boundary conditions and assigning the pressure and velocity initial values. The boundary conditions were assigned where the airfoil edges were considered as a wall. The three sides of the air volume were considered as an inlet, and the edge following the trailing edge of airfoil was considered as an air outlet.
5. The next step in the model development was setting up the solution algorithm. The problem was solved as steady state. The OpenFOAM solver used for solving this problem was Simple

FOAM. The following are the solution parameters for the SimpleFOAM solver: Start time: 0 sec; End time=500 sec; time step= 1sec. The SimpleFOAM solver uses the Gauss-Seidel method for solving. The pressure field is provided with a relaxation factor of 0.3 and the velocity field is assigned a relaxation factor of 0.7, along with the residual parameter which is set at 1.0×10^{-5} . The above parameters define the convergence criteria of the model.

6. The OpenFOAM model was then modified for parallel processing where the existing model was divided according to the number of HPC computing nodes.
7. The model was solved in parallel. Once the solution was converged, the solved model in the parallel processors was reconstructed to get the final simulation results. The final result is used to view the output of the airfoil simulation, and the respective result components are captured using the post-processing software tool ParaView.

The airfoil was surrounded by air volume and the variation in the flow velocity and air pressure near the airfoil section was reviewed. The different plots below show the flow of air and laminar behaviour observed in the airfoil region.

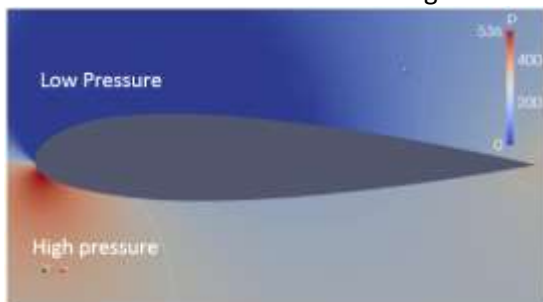


Figure 3: Pressure distribution around airfoil with high & low pressure zone.

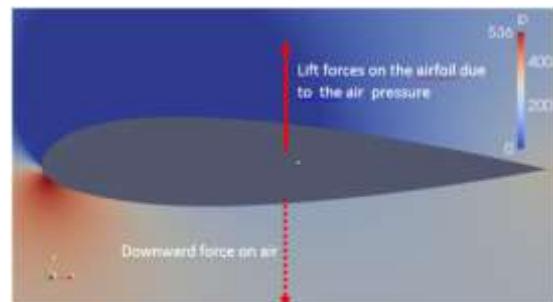


Figure 4: Lift forces represented in the airfoil.

The pressure plot shows the air pressure distribution in the airfoil sections. The first diagram represents the pressure variation around the airfoil where we observe the low-pressure region at the upper section of the leading edge of the airfoil and a higher-pressure region in the lower section of the leading edge. The low pressure and high-pressure variation section in the air volume is shown in the second diagram. The high-pressure section near the airfoil creates a lift forces on the airfoil. The lift on the airplane wing follows Newton’s third law – there will be a reaction force in the form of downward force on the air. The lift on the airplane wing should be consistent since it is governed by the conservation of the energy in the fluid.

Angle of attack is the orientation of the airfoil cord with respect to the travel direction. The state of stall can be analysed by determining the pressure co-efficient distribution over the airfoil for various angles of attack and evaluating how the pressure co-efficient value varies with the increase or decrease in the angle.

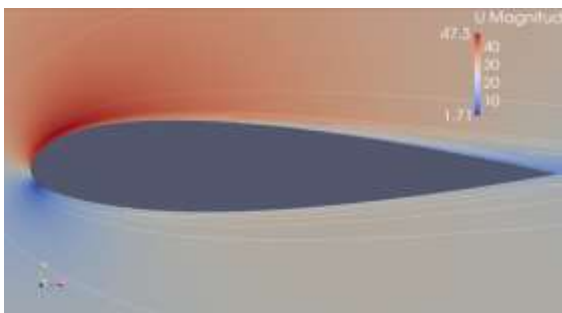


Figure 5: Velocity contour of streamline of air flow around the airfoil.



Figure 6: Velocity contour with air flow vectors around the airfoil.

The behavior of air flow was turbulent in the air volume, and the transition of the air behavior from turbulent to laminar was observed in the air volume nearing the airfoil section. The flow behavior of air was laminar around the wall of the airfoil. The airflow path near the wall boundary of the airfoil is laminar which is evident in Figures 5 and 6. The vector flow path of the air in the airfoil region was shown – the flow path represents individual air particle flow near the wall boundary of the airfoil.

HPC Performance Benchmarking

The HPC Cloud system is a 32-core system with 32 GB RAM with Ubuntu 12.04. The software installed in the container is OpenFOAM version 2.2 with OpenFoam MPI and Paraview. The model was evaluated for the accuracy of the prediction of air flow around the airfoil, with both fine and coarse mesh. The time required for solving the model with different meshes was captured to benchmark the use of HPC performance in solving high density mesh models. The boundary conditions, solution algorithm, solver setup and convergence criteria remain the same for all models.

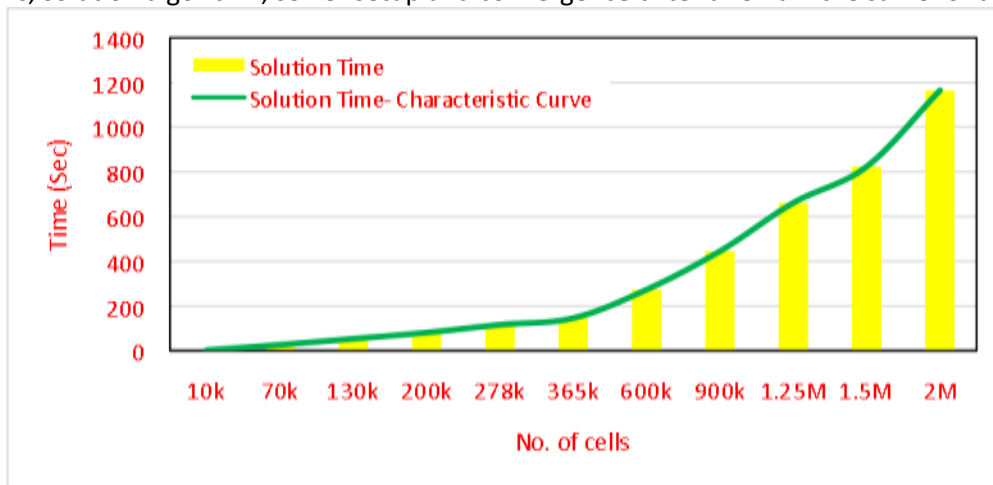


Figure 7: Solution time required in a 4-core configuration.

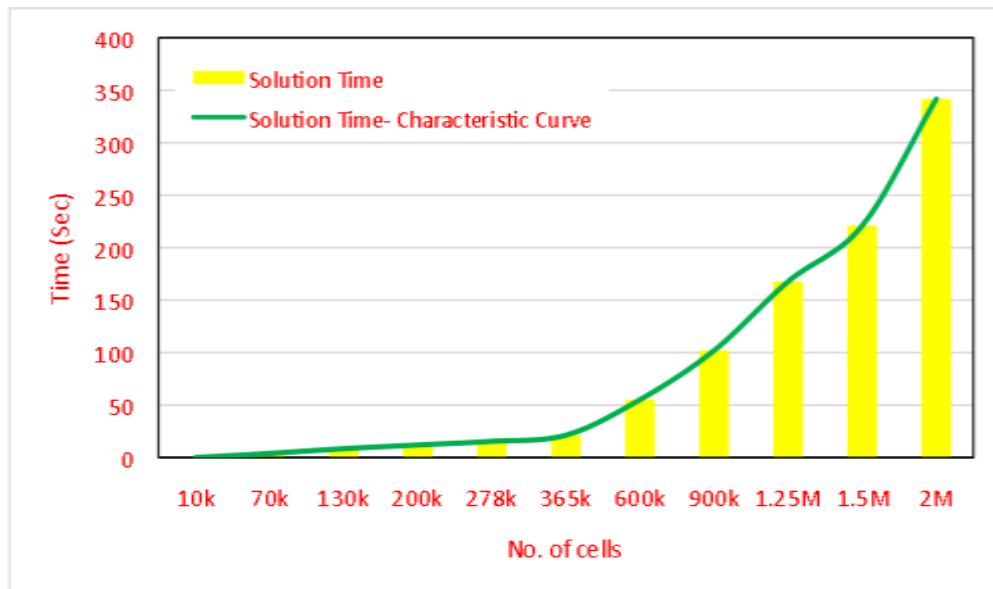


Figure 8: Solution time required in a 32-core HPC configuration.

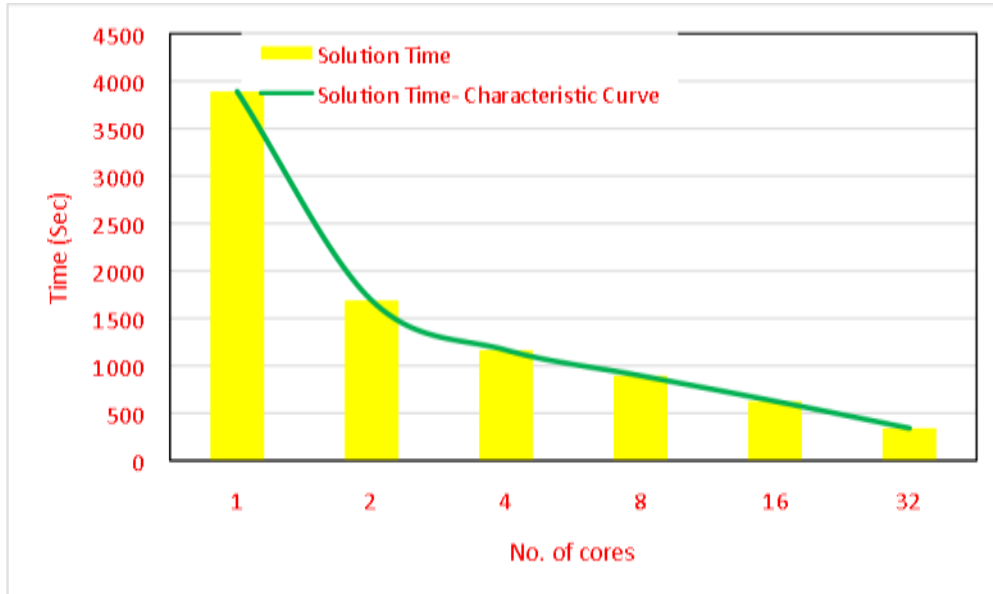


Figure 9: Solution time for a model with 2M elements using different core configurations.

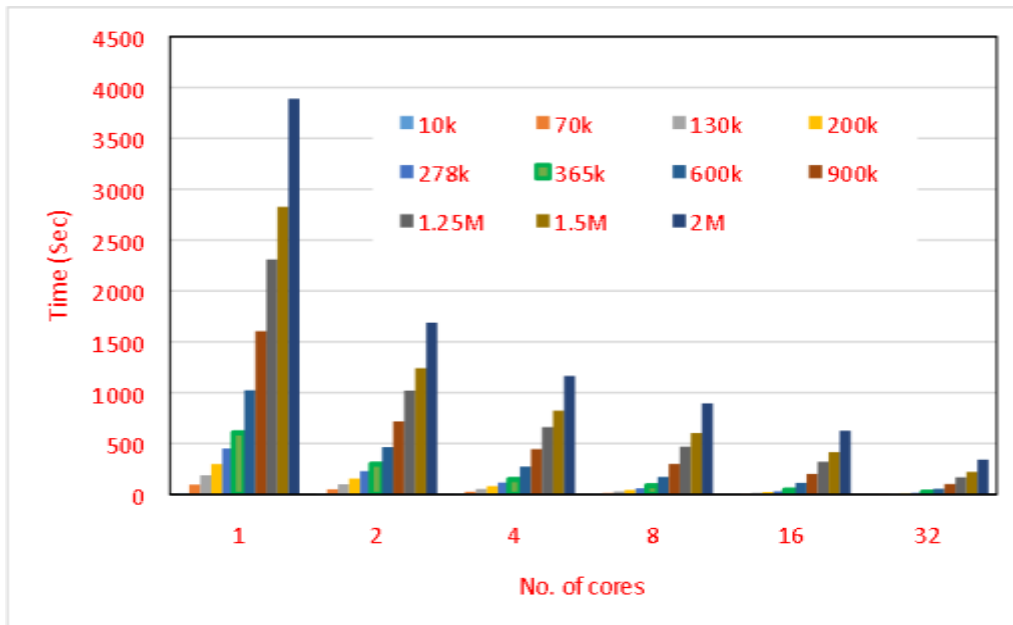


Figure 10: Comparison of solution time for different mesh densities models and different cores.

Effort Invested

End user/Team expert: 10 hours for simulation setup, technical support, reporting and overall management of the project.

UberCloud support: 3 hours for monitoring and administration of host servers and guest containers, managing container images (building and installing container images during any modifications/ bug fixes) and improvements (such as tuning memory parameters, configuring Linux libraries, usability enhancements). Most of the mentioned effort is one time and will benefit the future users.

Resources: ~200 core hours for performing various iterations in the simulation experiments.

CHALLENGES

The project started by testing the installation of OpenFOAM on the HPC server. Initial working of the application was evaluated and the challenges faced during the execution were highlighted. Once the

server performance was enhanced, the next level of challenges faced was related to technical complexity. This involved accurate prediction of flow behaviour around the airfoil which was achieved by defining the appropriate element size to the mesh model. The finer the mesh the higher the simulation runtime required: therefore, the challenge was to perform the simulation within the stipulated timeline.

BENEFITS

1. The HPC cloud computing environment with OpenFOAM and ParaView made the process of model generation easier with process time reduced drastically along with result viewing and post-processing.
2. The mesh models were generated for different cell numbers where the experiments were performed using coarse-to-fine to highly-fine mesh models. The HPC computing resource helped in achieving smoother completion of the simulation runs without re-trials or resubmission of the same simulation runs.
3. The computation requirement for a very fine mesh (2 million cells) is high, which is next to impossible to achieve on a normal workstation. The HPC cloud made it feasible to solve very fine mesh models. The simulation time was drastically reduced providing the advantage of obtaining simulation results within an acceptable run time (~30 min).
4. The UberCloud experiments in the HPC Cloud gave us extra confidence in the setup and the ability to run the simulations remotely in the cloud. The different simulation setup tools were installed in the UberCloud Container – this enabled the user to access the tool without any prior installations.
5. With the use of VNC Controls in the web browser, The UberCloud Container access was very easy with no installation of any pre-requisite software. The whole user experience was similar to accessing a website through the browser.
6. The UberCloud Containers helped with smoother execution of the project with easy access to the server resources. The regular UberCloud auto-update module via email provided huge advantage to the user that enabled continuous monitoring of the job in progress without any requirement to log-in to the server and check the status.

CONCLUSION AND RECOMMENDATIONS

1. The selected HPC Cloud environment with UberCloud containerized OpenFOAM on Nephoscale cloud resources was a very good fit for performing advanced computational experiments that involve high technical challenges and require higher hardware resources to perform the simulation experiments.
2. Cloud resources were an excellent solution for performing advanced computational simulation experiments that involved high technical challenges and required higher hardware resources.
3. There are different high-end commercial software applications which can be used to perform virtual simulation. Open source OpenFOAM with HPC UberCloud Containers helped us solve this problem with minimal effort in setting up the model and performing the simulation trials.
4. The combination of HPC Cloud, UberCloud Containers, and OpenFOAM helped in speeding up the simulation trials and allowed us to complete the project within the stipulated time frame.

Team 169

Complex Blood Flow through Cardiovascular Medical Device Using OpenFOAM on Advania



“The Advania Cloud and UberCloud containers are an ideal resource for our European engineering and scientific customers who want to burst into the cloud for complex resource-hungry HPC workloads.”

MEET THE TEAM

End-User/CFD Expert: Praveen Bhat, Technology Consultant, INDIA

Software Provider: Lubos Pirkli, CFD Support, Prague, Czech Republic

Resource Provider: Jean-Luc Assor, HPE, Paris, France, and Aegir Magnusson, Per-Ola Svensson, Hans Rickardt, Advania, Iceland.

Cloud Expert: Hilal Zitouni, Fetican Coskuner, Burak Yenier, The UberCloud, California, USA.

USE CASE

Many small and medium size manufacturers cannot afford to buy a powerful and expensive compute server to be able to run more complex and larger numbers of simulations which is necessary to manufacture higher quality products in shorter time. Buying a high-performance computer for the company means long procurement cycles, HPC expert knowledge to administer and operate the computer, additional expensive engineering software licenses, and high total cost of ownership.

This case study proves that using Advania’s cloud resources together with UberCloud’s application software containers provide an excellent alternative to owning on-premise computing resources, coming with ease of software portability to the cloud, instant access to and seamless use of Advania cloud resources, and performance scalability from few to many cores, with an on-demand and pay-per-use business model.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don’t have to learn anything about

HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

During this one-week Proof of Concept (PoC) we used UberCloud containers to set up a technical computing environment on the Advania Platinum instances. OpenFOAM, the popular open source computational fluid dynamics toolkit was used to simulate complex blood flow through a cardiovascular medical device. Pre-processing, simulation runs, and post-processing steps were performed successfully with the OpenFOAM container coming with a fully equipped powerful virtual desktop in the cloud and containing all the necessary software, data and tools.

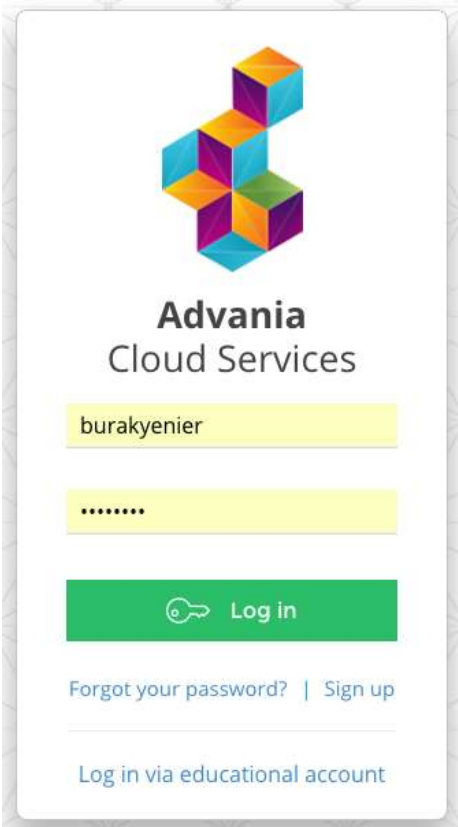
<p>An Advania Qstack Cloud Service demo account was created and the Advania control panel was used as the primary method of infrastructure administration.</p> <p>The Advania Cloud Services user interface was easy to use and responsive.</p> <p>The Platinum 3X Large instance type was selected for the PoC due to its large size.</p> <p>The Platinum 3x Large specifications are:</p> <ul style="list-style-type: none">16 virtual CPU cores61 GB RAM20 GB disk was selected <p>The instance start times were around 2-4 minutes.</p> <p>Instances were easily cloned and the clone instances performed as expected.</p>	
--	---

Figure 1: Advania Environment Setup.

An Advania Qstack Cloud Service demo account was created and the Advania control panel was used as the primary method of infrastructure administration. The Advania Cloud Services user interface was easy to use and responsive. The Platinum 3X Large instance type was selected for the PoC due to its large size. The Platinum 3x Large specifications are: 16 virtual CPU cores, 61 GB RAM, and 20 GB disk. The instance start times were around 2-4 minutes. Instances were easily cloned and the clone instances performed as expected.

DOCKER RUNTIME AND UBERCLOUD CONTAINER SETUP

The Advania instances were accessed via SSH, and the Docker run time environment was set up. This set up process took around 5 minutes and was automated down to a single command. The OpenFOAM container was pulled from the UberCloud private registry. This process took around 10 minutes. The OpenFOAM container was then launched on the Docker run time environment with no

further configuration or set up. To allow access to the OpenFOAM container via remote desktop VNC service, the related ports were opened through the Advania control panel as seen in Figure 2.



Figure 2: Firewall configuration through Advania control panel.

THE ENGINEERING USE CASE: MEDICAL DEVICE MODEL

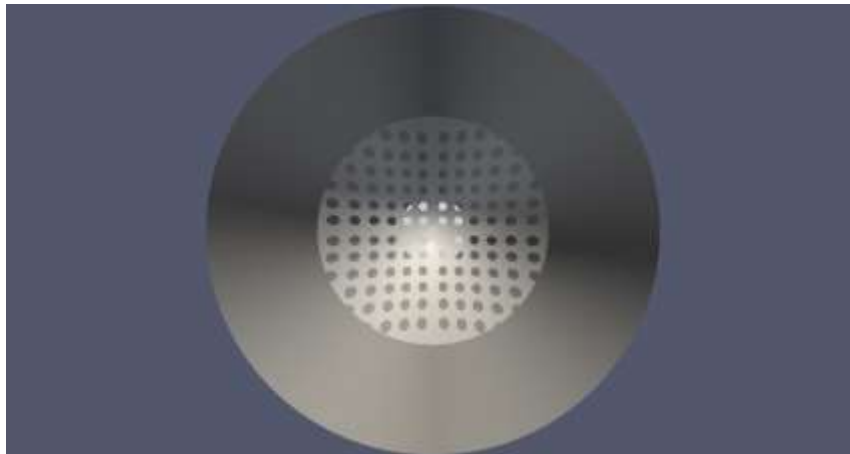


Figure 3: CAD model of the blood clot filter placed inside an artery. Captured using ParaView running inside an UberCloud Container on the Advania Cloud. To increase complexity of the problem blot clots were also inserted into the model (not shown above).

The model which was set up for testing is a simulation of blood flow through a cardiovascular medical device, a blot clot filter. Figure 3 shows a screenshot of the CAD model captured by accessing ParaView running inside an OpenFOAM container.

The CAD model of the medical device was used to generate a mesh of 5 million cells. The blood flow through the medical device was computed on 16 cores in parallel over 1,000 time steps using the simpleFOAM solver. The results were then post-processed using ParaView running inside the OpenFOAM container.

Figure 4 shows the resulting streamlines, representing the path the blood flows in the artery and at the inlet of the medical device. Figure 5 displays the velocity plot, showing the blood flow at the inlet and the outlet of the medical device.

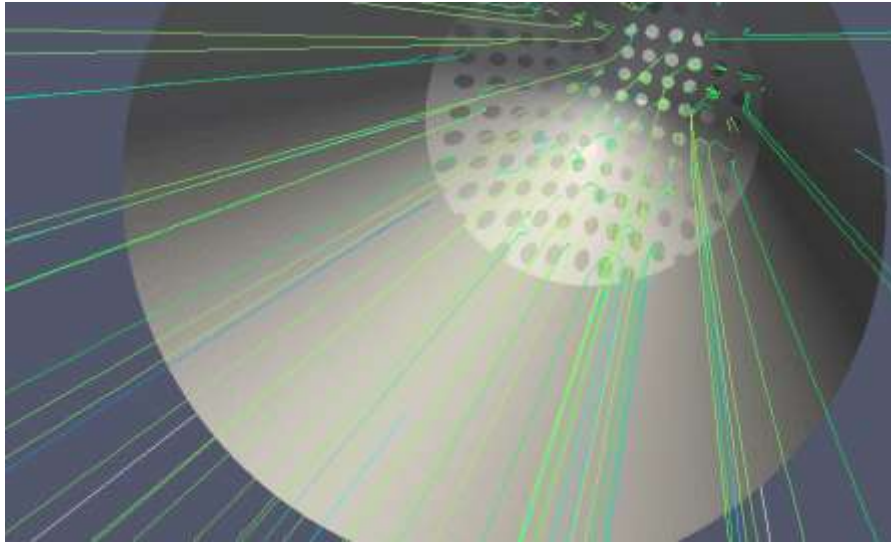


Figure 4: Streamlines, representing the path the blood flows in the artery and at the inlet of the medical device, using ParaView running inside an UberCloud Container on the Advania Cloud.

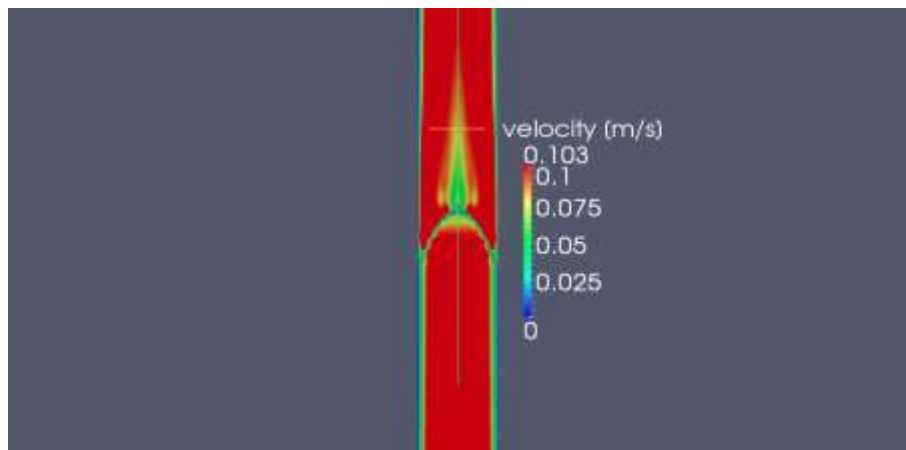


Figure 5: Velocity plot of the blood flow at the inlet and the outlet of the medical device, using ParaView running inside an OpenFOAM Container on the Advania Cloud.

USER EXPERIENCE

ParaView running inside an OpenFOAM Container on the Advania Cloud and accessed remotely via VNC demonstrated good performance. The end user was able to post-process the results, view 3D representations and manipulate these graphics (pan, zoom, rotate, etc.) in real time. The full featured desktop provided the entire regular feature set of ParaView (see Figure 6); there was no training required for the user to access and use this application container on the Advania Cloud. Screen captures were generated using ParaView and the resulting image files were transferred from the UberCloud container to a local desktop using SCP to generate this report.

MONITORING AND PERFORMANCE

During the testing phase, system utilization information was collected through two methods: the Advania dashboard and the fully automated UberCloud Container Monitoring. Advania dashboard offers basic, easy to use monitoring of the CPU and network utilization. The report can be run for

daily, weekly and monthly intervals. The reports update frequently and reflect the utilization of the resources at summary level.

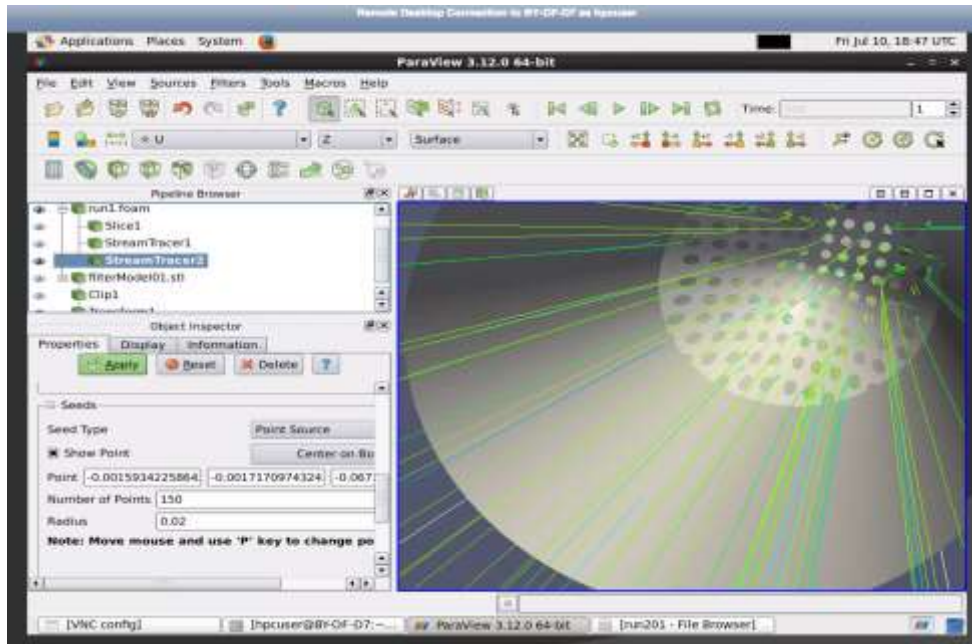


Figure 6: Full featured desktop view for remote visualization using ParaView in an UberCloud Container on the Advania Cloud. The desktop view provides usability and eliminates user training needs.

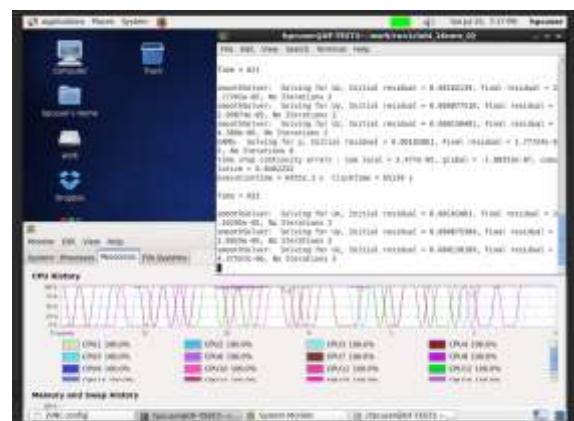
UberCloud containers are equipped with an automated monitoring feature, which sends the user an up to date snapshot of the system utilization levels, and the progress of the running simulation. During testing the automated monitoring feature of the UberCloud containers running on Advania resources worked as expected and the testing team was able to monitor system utilization and record when test runs are complete. This test was not intended to achieve the best performance possible; no effort was put into tuning the compute environment and gathering statistically relevant performance data. To provide a sense of the intensity of the calculation the following rough estimates are provided.



During the testing phase, system utilization information was collected through two methods: the Advania dashboard and the automated UberCloud Container Monitoring. Advania dashboard offers basic, easy to use monitoring of the CPU and network utilization. The report can be run for daily, weekly and monthly intervals.

The reports update frequently and reflect the utilization of the resources at summary level.

UberCloud containers are equipped with an automated monitoring feature, which sends the user an up to date snapshot of the system utilization levels, and the progress of the running simulation.



During testing the automated monitoring feature of the UberCloud containers running on Advania resources worked as expected and the testing team was able to monitor system utilization and record when test runs are complete. This test was not intended to achieve the best performance possible; no effort was put into tuning the compute environment and gathering statistically relevant performance data. To provide a sense of the intensity of the calculation the following rough estimates are provided.

On a Platinum 3X Large instance, the SimpleFOAM solver ran on 16 cores in parallel for 1,000 time-steps of the cardiovascular device simulation in 30,000 seconds (roughly 8 hours).

TOTAL EFFORT

The total effort (without the 8 hours simulation run time) described above to access Advania's OpenCloud, familiarize with the environment, setting up the OpenFOAM container, testing, developing the medical application geometry, boundary conditions, starting the jobs, and doing the post-processing with ParaView, and contacting and talking to Advania Support, was as follows:

- 2 hours in setting up the test account, getting familiar with GUI, requesting increase in quotas
- 1 hour in setting up the Docker environment, getting our base container, doing a quick test
- 3 hours in setting up the medical device simulation, doing steps like meshing, running the simulations (by the way, we ran it 5 times), monitoring, opening tickets with support, etc.

In total this resulted in a person effort of 6 hours for all the activities described above.

BUSINESS BENEFITS AND NEXT STEPS

The tests on Advania resources proved the compatibility of UberCloud's container technology and Advania's compute resources. Using the two together, a desktop like environment, with familiar user experience and with very low overhead can be set up, effortlessly. The major business benefits which are demonstrated by this case study are:

Benefits for the end-user:

- Portability: any cloud looks like the user's workstation
- User-friendly: nothing new to learn, ease of access and use
- Control: container monitoring allows the user to control his assets in the cloud.

Benefits for the resource provider:

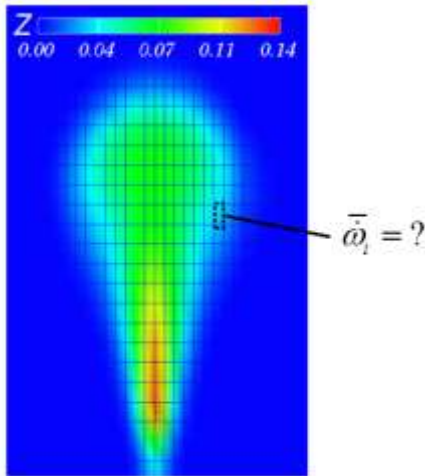
- Getting variability into their environment. Customers want different products which is easily implemented with container packaging and stacking
- Low overhead resulting in high performance
- High utilization by better use of resources.

Benefits for the ISV:

- Portability: software can run on a variety of different resource providers; built once, run anywhere
- Control of software usage via container-based license and usage monitoring
- Control of user experience
- The faster the software runs the better the user experience; containers enable porting of the software to workstations, servers, and to any cloud.

Team 177

Combustion Training in the Cloud



“UberCloud container technology harnessing Ansys Fluent CFD software on CPU24/7 HPC resources, accessed from a browser on a laptop computer, provided a light and seamless user experience.”

MEET THE TEAM

End-user: A. de Jong Group, Energy and Environmental Technologies, The Netherlands

Combustion Expert: Ferry Tap, [Dacolt](#), The Netherlands

Software Provider: Wim Slagter, [ANSYS Inc.](#) and [UberCloud Containers](#)

Ansys Container Provider: Fethican Coskuner, [UberCloud](#)

Resource Provider and HPC Experts: Thomas Gropp, Alexander Heine, Christian Unger, [CPU 24/7](#), Potsdam, Germany.

USE CASE

Dacolt provides highly appreciated Combustion CFD trainings for ANSYS Fluent since 2012. When delivering such trainings on-site, a number of challenges are faced:

- **Does the end-user have sufficient CFD licenses available?**
- **Does the end-user have sufficient HPC resources available?**
- **How are the HPC resources accessed?**

Not so long ago, some training sessions involved running on laptop computers which had to be physically moved around as well as being up-to-date from the Operating System and the CFD software perspective, involving substantial logistics and potential IT headaches.

In this UberCloud Experiment, the ANSYS software is provided in a Linux (Docker-based) [UberCloud container](#), which runs on CPU 24/7 HPC Cloud resources. The trainer accesses the HPC system via a web browser, using the end-user company’s guest WIFI network. The four end-user trainees each access the HPC system from their local workstations, also directly in the web browser.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on

Linux [Docker container](#) technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

USER EXPERIENCE

The end-users and trainer used Fluent on their own workstations. The login process is simple, getting files in and out of the HPC Cloud system works without any problem using a web-based file exchange system, in this case Dacolt's Basecamp account. The whole experience was so natural that it seemed as if this way of working was daily routine. For the trainer, the [UberCloud container](#) technology provides a very simple and scalable solution to provide training in the field of HPC, having to bring only a laptop computer.

BENEFITS

1. The UC Ansys container is very intuitive to use, it is a remote desktop running within the web browser. Also non-Linux users did not have any trouble to run their tutorials.
2. For the end-user, the company did not have to prepare any logistics to host the training.
3. For the trainer, the logistics only consisted in being on time, knowing the required resources were up and running in the Cloud.

CHALLENGES

The only real challenge encountered was on the back-end, to let the UC Ansys containers with Fluent check-out a license from the CPU24/7 license server. Through very effective team work and excellent support from Ansys, UberCloud and CPU24/7 resolved this issue swiftly.

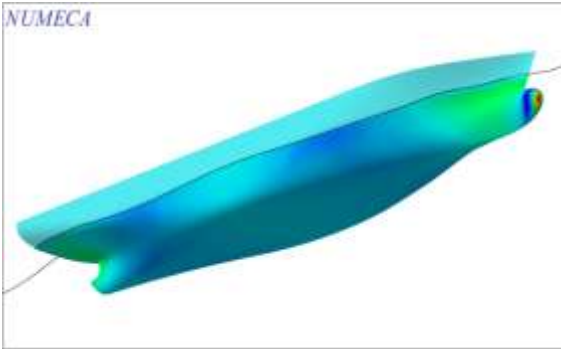
CONCLUSION & RECOMMENDATIONS

1. The selected HPC Cloud environment with the [UberCloud Ansys container](#) was a very good combination to provide the training to multiple trainees on a customer site.
2. The HPC resources from CPU24/7 were more than sufficient to allow the users to run their tutorials.
3. The light-weight web-access to the training environment is very comfortable for both trainer and trainees.

Case Study Author – Dr. Ferry Tap, Dacolt/AVL List

Team 181

Prediction of Barehull KRISO Containership Resistance in the Cloud



“A responsive graphical user interface, novel container technology, and an outstanding hardware performance make CPU 24/7 a viable alternative to local server acquisition and management.”

MEET THE TEAM

End User and Team Expert – Justin M. Morgan, PE Principal, Ocean Engineering & Analysis, Glosten Inc.

Software Provider – Aji Purwanto, Business Development Director, NUMECA International S.A.

Resource Provider – Richard Metzler, Software Engineer, CPU 24/7 GmbH

Technology Experts – Hilal Zitouni Korkut and Fethican Coskuner, UberCloud Inc.

USE CASE

In this project, we calculated the barehull resistance of the KRISO containership (KCS) in the cloud. The KRISO containership is a standard hull form frequently used as a benchmark case for computational fluid dynamics in the marine industry. Both basic hull form parameters and experimental results are available in published literature. Detailed information for the KCS test case is also available on the internet here:

https://www.nmri.go.jp/institutes/fluid_performance_evaluation/cfd_rd/cfdws05/

The purposes of this project were to become familiar with the mechanics of running a FINE/Marine simulation in an UberCloud container and to assess the performance of the available hardware compared to resources currently used by the end-user. The benchmark case was analyzed on local hardware, on virtual instances in the cloud, and on the bare-metal cloud solution offered by CPU 24/7 and UberCloud. All simulations were run using version 4 of Numeca’s FINE/Marine software.

The cloud resource provider CPU 24/7 GmbH is a leading provider of CAE as a Service solutions for all application areas of industrial and academic /university research and development. Headquartered in Potsdam/Germany, CPU 24/7 develops and operates unique on demand services for High Performance Computing that are based on the latest globally accepted industry standards for hardware, software, and applications.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the

access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

CHALLENGES

No challenges were experienced in downloading project files into the FINE/Marine container, running the simulation, or retrieving data. The remote desktop user interface was responsive without any significant delays. Logging into the system is simple and the Numeca software pre-installed in an UberCloud container runs without any user setup. The only user setup required is to adjust the display resolution.

PROCESS AND BENCHMARK RESULTS

The simulation was setup as a steady state solution, fixed in trim and heave to duplicate the conditions of the experimental data. The half model mesh contains 1.6 million cells. Simulation control variables in FINE/Marine were as follows:

- 300 time steps
- Uniform time step = 5 sub-cycles, 8 non-linear iterations

The solution converges to a steady state resistance force within about 150 time steps; however, the simulation was allowed to run to completion on all platforms to provide a performance comparison.

The calculated total resistance coefficient for this model is 0.003574 compared to the experimental result of 0.00356, a 0.4% difference. Figure 1 illustrates the calculated wave field (top) compared to measured data (bottom).

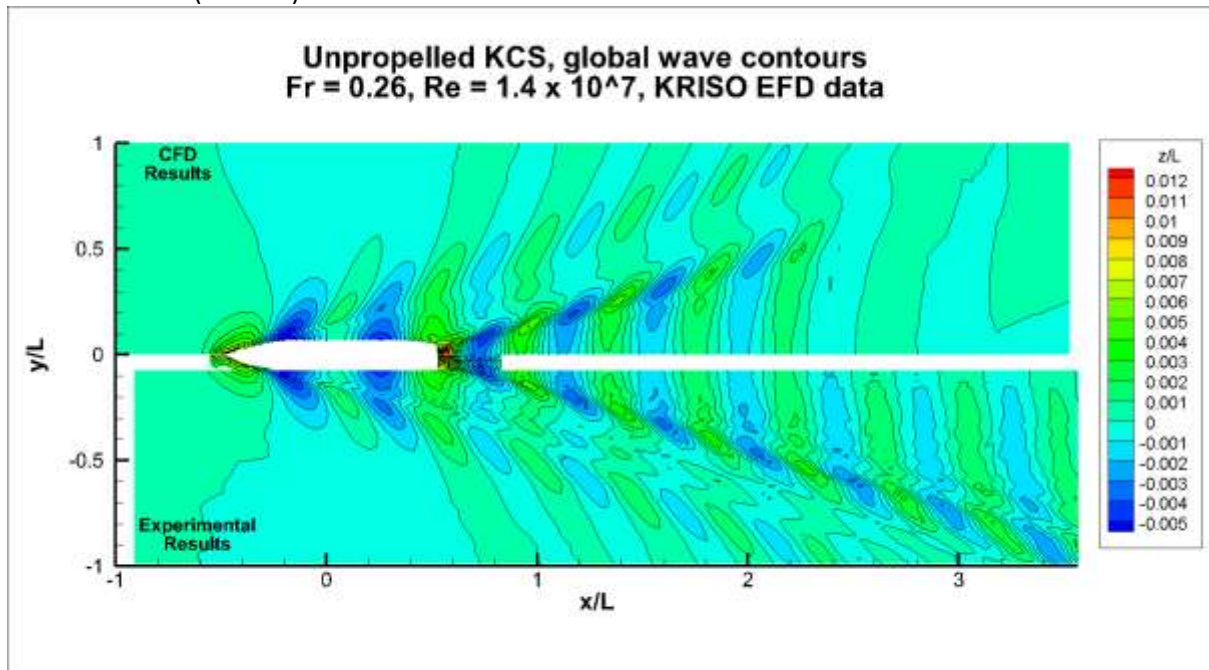


Figure 2: Comparison of experimental and calculated results.

The processors available through the UberCloud container provide a significant improvement in performance over local Glosten hardware and over virtual instances available through Amazon Web Services (AWS). The AWS compute instance used here is the third generation, c3.8xlarge. A fourth-

generation compute instance is available on AWS with Intel Xeon CPU E5-2666 v3; however, setting up a new virtual instance was considered too costly for this project.

Platform	Processor	FINE/Marine	#cores	Time [hrs]
local	Intel Xeon CPU E5645 @ 2.4 GHz x 2	v4.2	12	6.9
UberCloud	Intel Xeon CPU E5-2690 v3 @ 2.6 GHz x 16	v4.1	12	3.0
UberCloud	Intel Xeon CPU E5-2690 v3 @ 2.6 GHz x 16	v4.1	16	2.5
UberCloud	Intel Xeon CPU E5-2690 v3 @ 2.6 GHz x 16	v4.1	24	1.7
AWS	Intel Xeon CPU E5-2680 v2 @ 2.8 GHz x 23	v4.2	16	3.5
AWS	Intel Xeon CPU E5-2680 v2 @ 2.8 GHz x 23	v4.2	24	2.9

Figure 3: Performance comparison (Benchmark numbers are from 2015). The difference between v4.2 and v4.1 is only in patches not affecting performance.

BENEFITS

This use case helped us understand the performance benefits offered by UberCloud and its partners. Glosten considers the UberCloud service to be a viable alternative to a local server upgrade. Additional benefits include the on-demand access and use of the software and hardware resources, a reduction in overhead required to manage virtual instances and to maintain software updates.

CONCLUSIONS

- We showed that the CPU 24/7 HPC bare-metal cloud solution provides performance advantages for Numeca FINE/Marine users who want to obtain higher throughput or analyze larger, more complex models.
- CPU 24/7 and UberCloud effectively eliminate the need to maintain in-house HPC expertise.
- The container approach provides immediate access to high performance clusters and application software without software or hardware setup delays.
- The browser-based user interface is simple, robust, and responsive.

APPENDIX: UberCloud Application Containers for Numeca FINE™/Marine

UberCloud Containers are ready-to-execute packages of software. These packages are designed to deliver the tools that an engineer needs to complete his task in hand. In this experiment, the FINE™/Marine software has been pre-installed, configured, and tested, and were running on bare metal, without loss of performance. The software was ready to execute literally in an instant with no need to install software, deal with complex OS commands, or configure.

The UberCloud Container technology allows wide variety and selection for the engineers because the containers are portable from server to server, Cloud to Cloud. The Cloud operators or IT departments no longer need to limit the variety, since they no longer have to install, tune and maintain the underlying software. They can rely on the UberCloud Containers to cut through this complexity.

This technology also provides hardware abstraction, where the container is not tightly coupled with the server (the container and the software inside isn't installed on the server in the traditional sense). Abstraction between the hardware and software stacks provides the ease of use and agility that bare metal environments lack.

Case Study Author – Justin M. Morgan, Glosten Inc.

Team 182

OpenFOAM CFD Modelling and Product Optimization of Dry-type Transformers



“Cloud based computational techniques extend our potential to design/develop better products. By utilizing the potential, the products can be optimized with a much faster pace.”

MEET THE TEAM

End user and Team Expert – Wei Wu, Senior R&D Design/Development Engineer, ABB

Software Provider – [ESI – OpenCFD](#) providing OpenFOAM

Resource Provider – [Microsoft Azure](#) with [UberCloud OpenFOAM Container](#)

Technology Experts – Fethican Coskuner, Hilal Zitouni, and Baris Inaloz, [UberCloud Inc.](#)

USE CASE

Dry-type transformer has growing applications in transformer market because the technology is non-flammable, safer and environmentally friendly. On the other hand, dry-type transformers typically have bigger dimensions compared to the liquid-immersed units to have sufficient dielectric insulation and cooling capacity. Therefore, how to manage to design dry-type transformers with smaller sizes in order for lower material cost, while still satisfying dielectric and thermal performance, is one of the high priority tasks of a transformer manufacturer.

In this project, we aim at using OpenFOAM open-source CFD package to simulate the heat transfer of a same dry-type transformer unit with a group of different dimensions. In this way, the temperature rises can be evaluated and compared, directing a way to optimize the transformer design in terms of thermal performance. As the CFD model is built as 3D, even though only one quarter of the geometry is taken into account considering the geometry symmetry, still a number of millions of computation cells will be generated. This is why a cloud based computational platform has been considered as an option to speed up the entire evaluation cycle.

TECHNOLOGY: UBERCLOUD CONTAINERS

[UberCloud Containers](#) are ready-to-execute packages of software. These packages are designed to deliver the tools that an engineer needs to complete the task in hand. The ISV (Independent Software Vendor) or Open Source tools are pre-installed, configured, and tested, and are running on bare metal, without loss of performance. They are ready to execute, literally in an instant with no need to install software, deal with complex OS commands, or configure. The [UberCloud Container](#) technology allows a wide variety and selection for the engineers because they are portable from

server to server, Cloud to Cloud. The Cloud operators or IT departments no longer need to limit the variety, since they no longer have to install, tune and maintain the underlying software. They can rely on the UberCloud Containers to cut through this complexity. This technology also provides hardware abstraction, where the container is not tightly coupled with the server (the container and the software inside isn't installed on the server in the traditional sense). Abstraction between the hardware and software stacks provides the ease of use and agility that bare metal environments lack.

CHALLENGES

Overall, the computation speed is the bottleneck of the simulation project. On a modern desktop workstation computer with 4 to 6 cores (8 to 12 threads), the iterative steady-state computation of a single case can take 1.5 to 3 hours, which is actually bearable. However as in design optimization one single case is certainly not enough. Geometry dimensions and physic properties can all be changeable and the combination of the changed parameters can be a large number of simulation cases. Furthermore, for special cases, time-transient models may be necessary which takes even more computational effort to accomplish.

PROCESS AND BENCHMARK RESULTS

The computations were performed on a 10-node class "medium" cluster, where eight compute nodes were equipped with dual socket Intel(R) Xeon(R) CPU E5-2670 @ 2.60GHz and 112 GB of RAM, giving a total count of 128 cores and 1TB of RAM. The nodes were connected with 40Gbit/s InfiniBand network with remote direct memory access (RDMA) technology. This hardware setup was chosen since it is suitable for a first test of the capabilities of HPC computing in the cloud. The hardware was supplied by Microsoft Azure, and OpenFOAM CFD package was running on UberCloud Containers.

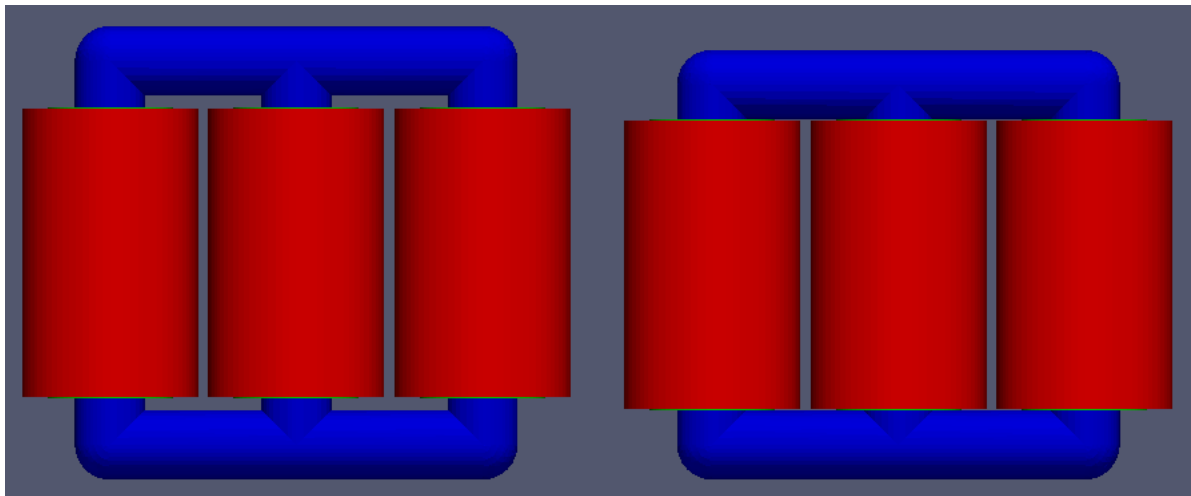


Figure 1: The two cases simulated. The right-hand size case has lower dimension than the left-hand side one. The blue color part is iron core and the red color part is high-voltage coils.

The benchmarks were performed with two cases with different height dimensions, as Fig. 1 shows. The simulation took 7 and 8 minutes respectively for these two cases to accomplish 500 iteration steps; the results are illustrated in Fig. 2. However, the same simulation cases will need 77 and 82 minutes respectively on a local workstation with 11 core threads, which demonstrates the speed-up linearly proportional to the number of processor core threads. The computational time comparison is summarized in Fig. 3.

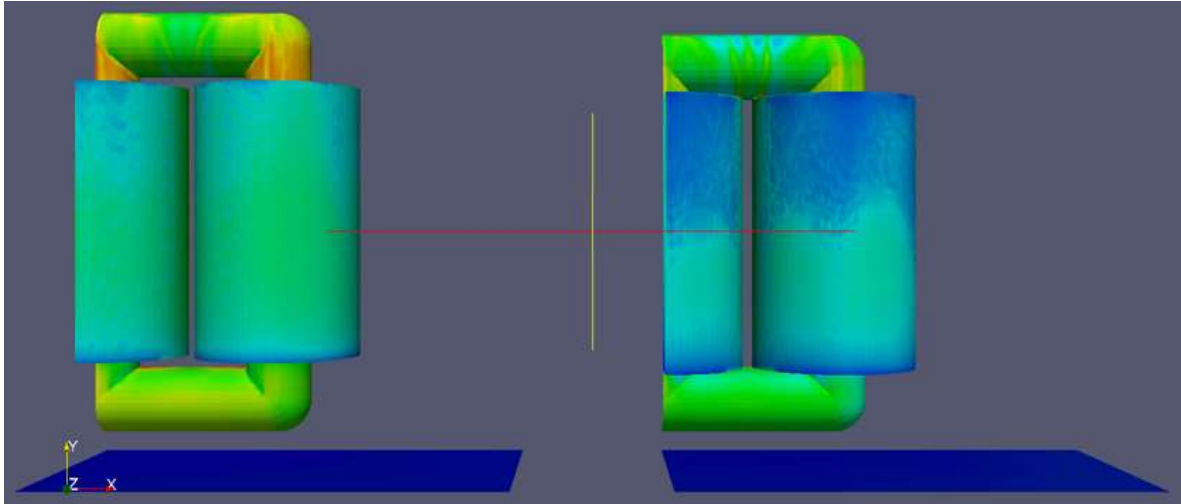


Figure 2: Temperature distribution example of both cases in Fig. 1. Due to geometry symmetry only a quarter of the full domain was simulated.



Figure 3: Computational time comparison between the two cases in Fig. 1. With the cloud cluster, computational time is reduced to only 10%.

BENEFITS

The cloud-based technology has significantly higher computational speed, which made parametric study or optimization of transformer designs much faster. On the other hand, the technology costs relatively lower compared to owning in-house HPC equipment with equivalent computational power.

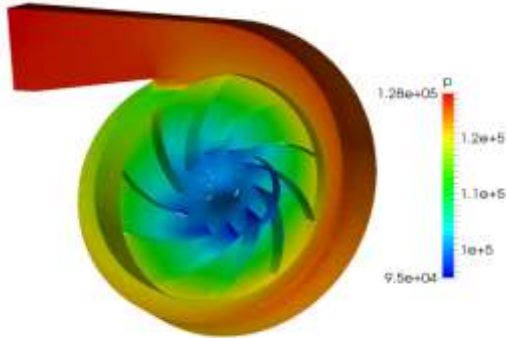
CONCLUSIONS

- We showed that the Microsoft Azure based UberCloud container solution is a beneficial solution for OpenFOAM users who have the need to deliver their simulation results in a much faster time manner.
- To use the cloud-based cluster computing, there is no investment in in-house HPC equipment or expertise needed, since UberCloud offers customized and handy cloud cluster solutions with all requisite software packages pre-installed.

Case Study Author – Wei Wu, ABB

Team 183

Radial Fan CFD Simulation in the Azure Cloud



“The Azure cloud together with UberCloud Containers provides excellent performance for Turbomachinery CFD users who want to obtain higher throughput or analyze more complex models.”

MEET THE TEAM

End User & Team Expert – Luboš Pirkli, Co-founder & Technical Director, CFD Support Ltd.

Software Provider – Turbomachinery CFD, CFD Support Ltd.

Resource Provider – Microsoft Azure Cloud

Technology Experts – Hilal Zitouni Korkut and Fethican Coskuner, UberCloud Inc.

USE CASE

CFD Support supports manufacturers around the world with numerical simulations. One of the main CFD Support's businesses is providing full support for virtual prototyping of rotating machines: compressors, turbines, fans and many other turbomachinery. All the rotating machines need to be simulated to test, confirm or improve its aerodynamic efficiency, which has major effect on its energy consumption. Each machine design is tested many times and is optimized to find the best efficiency point. In practice these CFD simulations are very demanding, because of complexity and number of simulations to run.

CFD Support is aiming at demonstrating the use of UberCloud's OpenFOAM offer on Microsoft Azure to be able to scale the number of parallel simulations to minimize the "time to market" of its complex consultancy projects. On the business side this will result in performing simulations in greater detail (resulting in higher quality turbomachineries and increased competitiveness), and much faster because of using many more and faster cloud resources (resulting in faster time to market and increased competitiveness).

In this project, we calculated the radial mechanical fan characteristics in the Azure Cloud, on different number of cores. The purpose of this project is to test the current fan design and to get the best fan efficiency possible to save energy costs. The project presents a smooth workflow of performing complex CFD analysis of radial fan using Turbomachinery CFD based on the OpenFOAM® software. Detailed information for this test case is also available here:

<http://www.cfdsupport.com/radial-fan-cfd-study.html>

The fan model is designed in [CFturbo](#). CFturbo is a modern powerful software for interactive design of turbomachinery. It's easy to use and enables the designer to either start a model from scratch or redesign existing geometries.

The designed model data are then exported from CFturbo. The surface model data together with the initial physical flow data are loaded into Turbomachinery CFD. This CFD methodology employs a multi-region approach, which means the model is split into a certain number of regions. Each region is being computed separately and communicates with other (neighboring) regions via interfaces.

The CFD workflow is fully automated, or can be automated for every other type of machine. Finally, in effective work, the user just runs one single script; all is done automatically. The mesh is created, the set-up is done, the CFD simulation is run and finally the results are evaluated. The mesh is created automatically within snappyHexMesh. Every region has its own mesh. In this fan case there are two independent regions (meshes): rotor and stator.

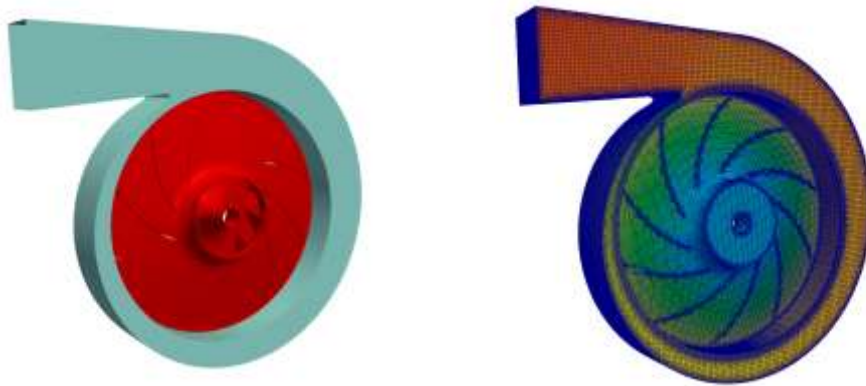


Figure 1: Turbomachinery CFD fan nq28 compressible noHousing: physical model and full computational mesh.

PROCESS AND BENCHMARK RESULTS

The initial CFD simulation set-up consists of the following steps and parameters:

- Compressible steady-state flow model
- Medium: air
- BEP Pressure ratio: $\Delta p_{Tot} = 1.265$ [-]
- Temperature at inlet: $T = 40$ [°C]
- Viscosity: $\mu = 1.831e-5$ [Pa.s]
- Rotation speed: 2980 [RPM]
- BEP Flow Rate: 38000 [m3/h]
- Interface: mixingInterface (radial averaging)
- Turbulence Model: k- ω SST
- Mesh: snappyHexMesh, hexadominant
- Mesh Cells: 996112
- Mesh Average y^+ (full/segment): 90 [-]

For more details about this CFD Simulation Set-up please see the [Turbomachinery CFD Manual](#).

Platform	Processor	TCFD	#cores	Time [hrs]
local	Intel Core i7-3970X CPU @ 3.50GHz	v. 15.09	6	8.80
Azure Cloud	Intel Xeon E5-2698B v3 @ 2.00GHz	v. 15.09	2	16.30
Azure Cloud	Intel Xeon E5-2698B v3 @ 2.00GHz	v. 15.09	4	10.05
Azure Cloud	Intel Xeon E5-2698B v3 @ 2.00GHz	v. 15.09	8	7.45
Azure Cloud	Intel Xeon E5-2698B v3 @ 2.00GHz	v. 15.09	16	4.97
Azure Cloud	Intel Xeon E5-2698B v3 @ 2.00GHz	v. 15.09	32	2.96

Figure 2: Performance comparison for local workstation versus Azure Cloud G5 instances and different number of cores.

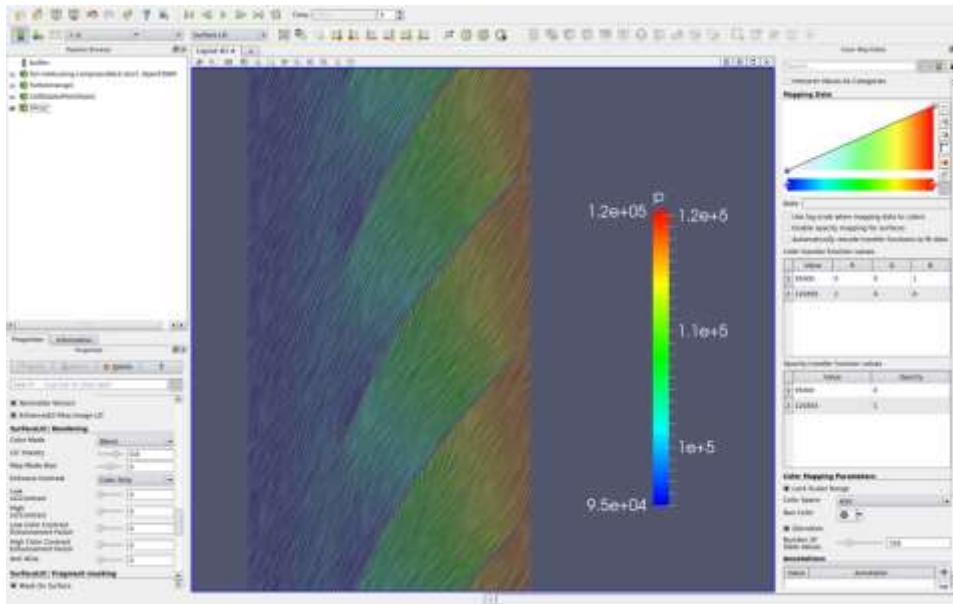


Figure 3: Turbomachinery CFD fan nq28, compressible turbo blade post stream traces.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

BENEFITS

This use case helped us understand the performance benefits offered by the Azure Cloud together with UberCloud Containers. CFD Support considers the UberCloud service to be a viable alternative to a powerful local workstation. Additional benefits include the on-demand access and use of the software and hardware resources, a reduction in overhead required when managing virtual instances and to maintain software updates.

No challenges were experienced in downloading project files into the Turbomachinery CFD container, running the simulation, or retrieving data. The remote desktop user interface was responsive without any significant delays. Logging into the system is simple and the Turbomachinery CFD software pre-installed in an UberCloud container runs without any user setup.

CONCLUSIONS

- We showed that the Azure cloud solution together with the UberCloud Containers provides excellent performance advantages for Turbomachinery CFD users who want to obtain higher throughput or analyze larger, more complex models.

- Azure Cloud and UberCloud effectively eliminate the need to maintain appropriate in-house HPC expertise.
- The container approach provides immediate access to suitable high performance computing resources and application software without software or hardware setup delays.
- The browser-based user interface is simple, robust, and responsive.

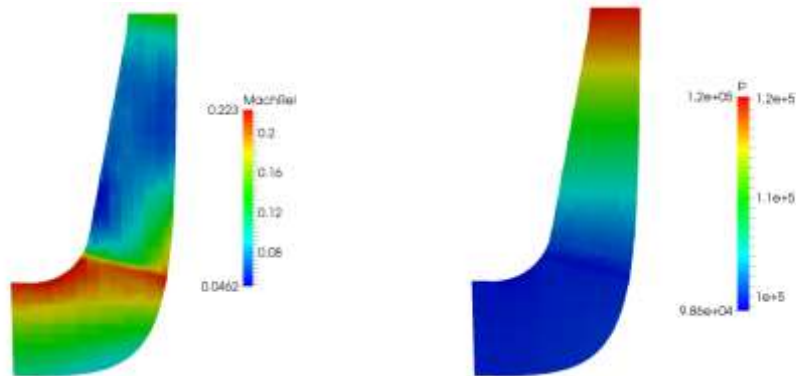


Figure 4: Turbomachinery CFD fan nq28 compressible meridional average relative Mach versus average Pressure.

APPENDIX: UberCloud Application Containers for Turbomachinery CFD

UberCloud Containers are ready-to-execute packages of fully portable software. These packages are designed to deliver the tools that an engineer needs to complete his task in hand. In this experiment, the Turbomachinery CFD software has been pre-installed in a container, configured, and tested. The software was ready to execute literally in an instant with no need to install software, deal with complex OS commands, or configure.

The UberCloud Container technology allows wide variety and selection for the engineers because the containers are portable from server to server, Cloud to Cloud. The Cloud operators or IT departments no longer need to limit the variety, since they no longer have to install, tune and maintain the underlying software. They can rely on the UberCloud Containers to cut through this complexity.

The container software technology also provides hardware abstraction, where the container is not tightly coupled with the server (the container and the software inside isn't installed on the server in the traditional sense). Abstraction between the hardware and software stacks provides the ease of use and agility that bare metal environments usually lack.

Case Study Author – Luboš PirkI, CFD Support Ltd.

Team 186

Airbag simulation with ANSYS LS-DYNA in the Microsoft Azure Cloud



“Microsoft Azure resources with UberCloud Containers and ANSYS LS-DYNA provide an excellent platform to develop and run accurate simulation models that involve complex impact physics.”

MEET THE TEAM

End-User/FEA Expert: Praveen Bhat, Technology Consultant, INDIA

Software Provider: ANSYS INC. and UberCloud LS-DYNA Container

Resource Provider: Microsoft Azure with UberCloud Containers

HPC Expert: Burak Yenier, Co-Founder, CEO, UberCloud

USE CASE

Automobile airbags are the results of some incredible engineering. In a high speed crash the driver of the car can be hurled into the steering wheel, but in an airbag equipped car, a small electronics device will inflate the airbag providing enough cushion to protect the driver from impact. Fatality and serious injury rates have been reduced since the widespread installation of airbags.

The main objective of this project is to understand the air bag inflation behaviour under dynamic conditions. The simulation framework is developed and executed with ANSYS LS-DYNA in an UberCloud container on Microsoft Azure computing resources to achieve good accuracy in result prediction and also with respect to the solution time and resource utilization.

PROCESS OVERVIEW



Figure 1: Geometry & Mesh model for a steering with closed airbag model.

1. The steering wheel with folded air bag is meshed using the 2D quad mesh elements. The contacts and interactions between different components in the steering wheel assembly and air bag is defined.
2. The material properties for the steering wheel assembly with air bag are defined. The section properties are defined which involved thickness definition for different components in the assembly.
3. The next step of the model setup is defining the model boundary conditions and assigning load curves. The steering wheel geometry is fixed and the load curve provides the air bag opening forces which are defined on the air bag component.
4. Solution algorithm and convergence criteria are defined along with output parameters and results to be used for post processing.
5. The Model is solved in ANSYS LS-DYNA with parallel computing on 1 to 16 cores. The final result is used to view the output of the simulation result, and the respective result components are captured using the post-processing software tool in ANSYS.

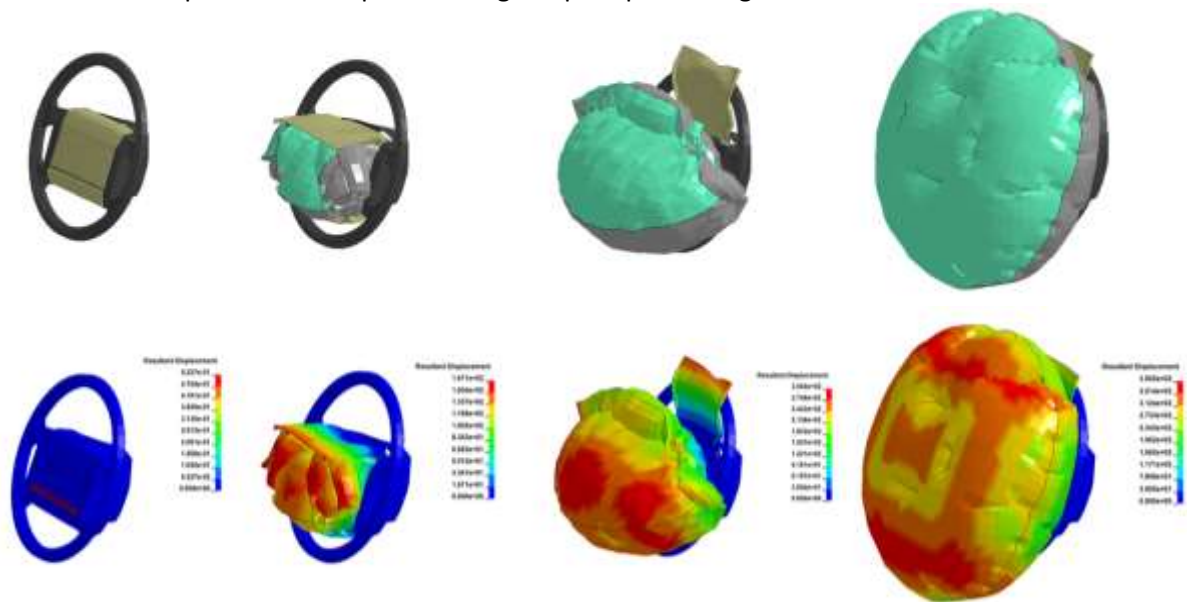


Figure 2: Deformation plot of air bag (a) Opening sequence of air bag, and (b) Contour plot on the steering and air bag assembly.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

HPC PERFORMANCE BENCHMARKING

The HPC system is a Microsoft Azure GS5 Instance: 32 cores, 448 GB RAM, Max Disk size OS = 1023 GB and local SSD = 896 GB, Cache size 4224, and Linux operating system. The air bag model is simulated using ANSYS LS-DYNA in an UberCloud Container on the Microsoft Azure cloud platform.

The model is evaluated for the air bag behaviour and it also determines the rate of air bag opening and the stresses developed on the air bag material.

Different finite element models are developed for fine and coarse mesh. The model data are submitted to the ANSYS LS-DYNA container. The time for solving the model with different mesh intensity is captured to benchmark the performance in solving high density mesh models. Boundary conditions, solution algorithm, solver setup and convergence criteria remain same for all models.

Figures 3 & 4 provide a comparison of solution times required for different mesh density model without (1 core) and with (8 cores) parallel processing. The comparison of the solution time with single core processor and 32 core processors shows that the time required to solve using parallel computing is significantly less when compared with running the same simulations with single core.

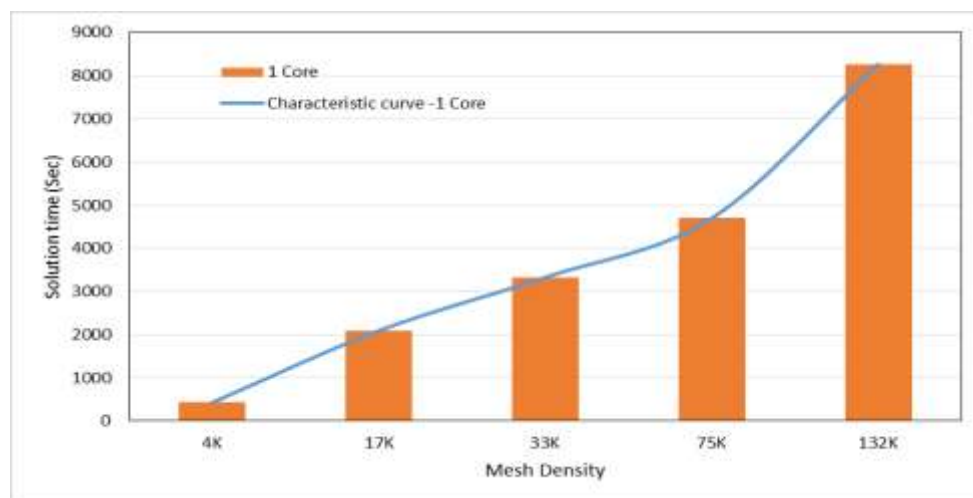


Figure 3: Solution time required for different mesh density with single CPU Core.

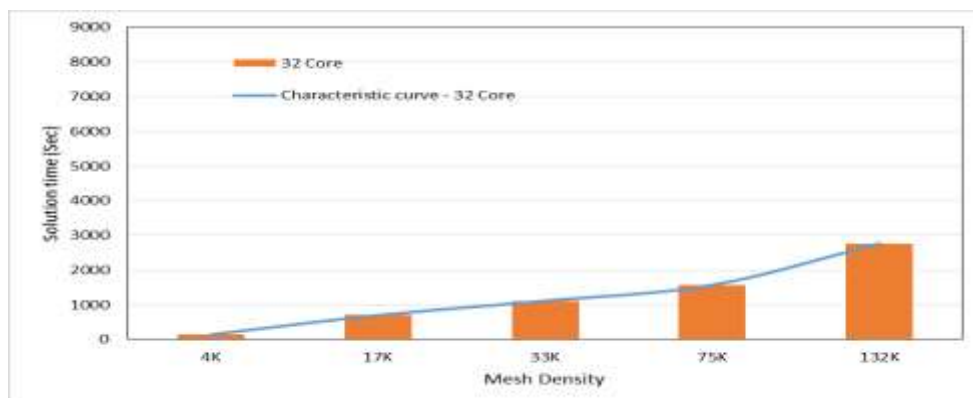


Figure 4: Solution time required for different mesh density using 8 CPU Core.

Figure 5 shows the comparison of the solution time required for a model with 132K elements which is submitted with different CPU cores. Figure 6 provides a comparison of the solution for different mesh models using different CPU cores. The comparison of the solution time with single core processor and 32 core processor again demonstrates that the time with parallel computing is significantly less when compared with running the same simulations with single core.

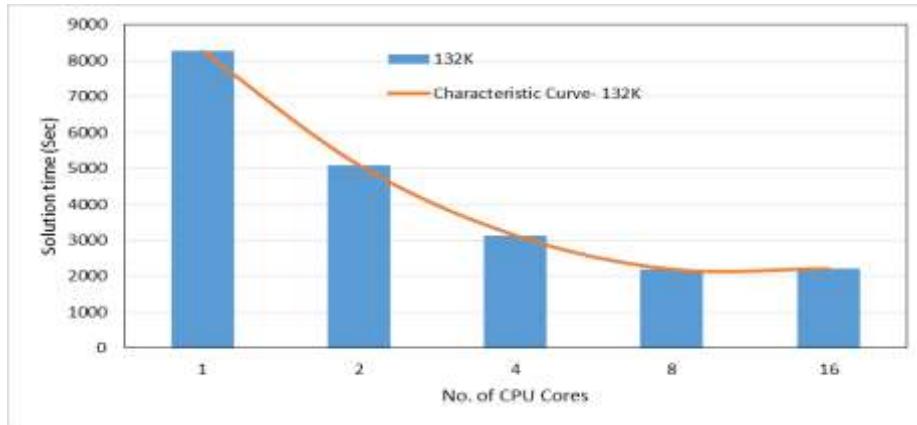


Figure 5: Solution time for a model with 132K elements solved using different HPC Core configuration.

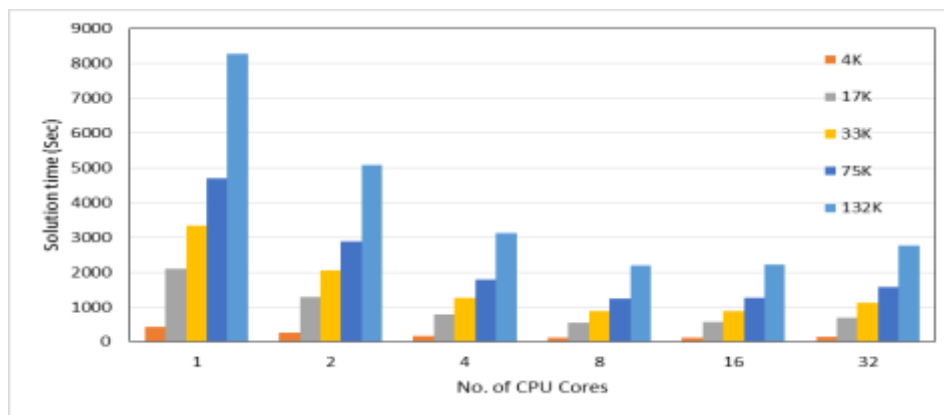


Figure 6: Solution time for models with different mesh densities using different HPC core configurations.

EFFORT INVESTED

End User/Team Expert: 10 hours for the simulation setup, technical support, reporting and overall management of the project.

UberCloud support: 1 hour for monitoring & administration of the performance in the host server.

Resources: ~1000 core hours used for performing various iterations in the simulation experiments.

CHALLENGES

The project challenges faced were related to technical complexity of the application and the ability to run the dynamic simulation within a very short period of execution time. Hence it was necessary to perform trials with different mesh density models to accurately capture the air bag behaviour. The finer the mesh the better is the simulation result accuracy, but the higher is the simulation runtime, and hence the challenge was to perform the simulation within the stipulated timeline. Getting exposure to the Azure Cloud platform and using its features consumed some time at first as this required going through learning and following written instructions provided by Azure.

BENEFITS

1. The HPC cloud computing environment with ANSYS workbench & LS-DYNA made the process of model generation easier with process time reduced drastically along with result viewing & post-processing because of the ANSYS / Azure / UberCloud HPC set-up.
2. The mesh models were generated for different cell numbers where the experiments were performed using coarse - to - fine to very fine mesh models. The HPC computing resource helped in achieving smoother completion of the simulation runs without re-

trials or resubmission of the same simulation runs thereby helping the user to achieve highly accurate simulation results.

3. The computation time requirement for a fairly fine mesh (~132K cells) is quite high, which is nearly impossible to achieve on a normal workstation. The HPC Cloud provided this feasibility to solve highly fine mesh models and the simulation time drastically reduced providing an advantage of getting the simulation results within acceptable time (~30 min).
4. The experiments in the HPC Cloud showed the possibility and gave extra confidence to setup and run the simulations remotely in the cloud. The different simulation setup tools were pre-installed in the HPC container and this enabled the user to access the tools without any prior installations.
5. With the use of VNC Controls in the Web browser, the HPC Cloud access was very easy with no installation of any pre-requisite software. The whole user experience was similar to accessing a website through the browser.
6. The UberCloud containers helped with smooth execution of the project with easy access to the server resources. The UberCloud ANSYS container integrated with the Microsoft Azure platform proved to be powerful as it facilitates running parallel UberCloud containers. A dashboard in the Azure helped in viewing the system performance and usage.

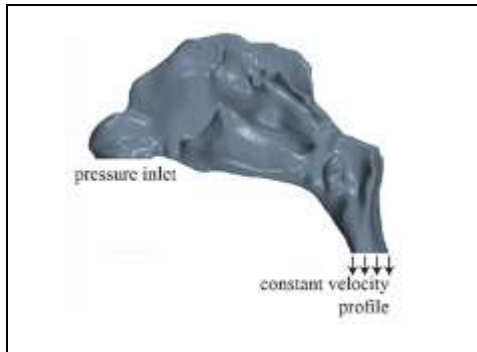
CONCLUSION & RECOMMENDATIONS

1. The selected HPC Cloud environment with UberCloud containerized ANSYS Workbench with LS-DYNA on Microsoft Azure was an excellent fit for performing complex simulation that involved huge hardware resource utilization with a high number of simulation experiments.
2. Microsoft Azure and UberCloud Containers enabled performing advanced computing that involve high technical challenges with complex geometries and which cannot be solved on a normal workstation.

Case Study Author – Praveen Bhat, Technology Consultant, INDIA

Team 190

CFD Simulation of Airflow within a Nasal Cavity



“Relatively small wall clock time necessary for simulation of one nasal cavity is very promising since it allows short rent times for cloud-based machines as well as software licenses.”

MEET THE TEAM

End Users, Team Experts – Jan Bruening, Dr. Leonid Goubergrits, Dr. Thomas Hildebrandt, Charité University Hospital Berlin, Germany

Software Provider – CD-Adapco providing STAR-CCM+

Resource Provider – Microsoft Azure with UberCloud STAR-CCM+ software container

Technology Experts – Fethican Coskuner, Hilal Zitouni, and Baris Inaloz, UberCloud Inc.

USE CASE

In-vivo assessment of nasal breathing and function is limited due to the geometry of the human nasal cavity, which features several tortuous and narrow gaps. While spatially well resolved, investigation of that complex geometry is possible due to sophisticated methods like x-ray computed tomography (CT) and acoustic rhinometry, there is no sufficient method for assessment of the nasal airflow as of yet. The current gold-standard for objective assessment of nasal function is the rhinomanometry, which allows measurement of the nasal resistance by measuring the pressure drop as well as the volume flow rate for each side of the nose. Thus, a complete characteristics curve for each side of the nose can be obtained.

While high total nasal resistance measured using rhinomanometry correlates well with perceived impairment of nasal breathing, indications may be faulty in some cases. This is caused by several reasons. Firstly, there is no lower limit of “healthy” nasal resistance. In patients featuring a very low nasal resistance, rhinomanometry would always indicate no impaired nasal breathing. However, conditions that feature low nasal resistances as well as heavy impairment of perceived nasal breathing exist (e.g. Empty Nose Syndrome). Furthermore, rhinomanometric measurements allow no spatially-resolved insight on nasal airflow and resistances. It is impossible to determine which region of the nasal cavity poses the highest resistance. This may be the main reason that the role of Computational Fluid Dynamics (CFD) for assessment and understanding of nasal breathing was rapidly increasing within the last years.

In this study the airflow within a nasal cavity of a patient without impaired nasal breathing was simulated. Since information about necessary mesh resolutions found in the literature vary broadly (1 to 8 million cells) a mesh independence study was performed. Additionally, two different inflow models were tested. However, the main focus of this study was the usability of cloud based high performance computing for numerical assessment of nasal breathing.

Methods

The geometry of the nasal cavity was segmented from CT slice images with a nearly isotropic resolution of $0.263 \times 0.263 \times 0.263 \text{ mm}^3$. The segmentation was performed mostly manually using radio density thresholds. The rough geometry of the segmented nasal cavity was then smoothed and cut at the nostrils as well as the pharynx at height of the larynx.

The truncation at the nostrils and thus neglecting the ambient surrounding of the face is common practice in numerical assessment of nasal breathing. No severe differences in numerically calculated pressure drops and wall shear stress distributions were found when including the ambient compared to geometries truncated at the nostrils. However, no numerical study has been performed yet investigating the change in intranasal airflow while wearing a mask, as it is necessary during the rhinomanometric measurements. Therefore an additional geometry was created, where an oval shaped mask with an outflow nozzle with a diameter of 22 mm was created as well. Therefore, flow differences caused by those two inflow conditions could be evaluated. The mesh independency study was only performed for the truncated models.

Finite Volume meshes were created using Star-CCM+ (v. 11.02). Surfaces were meshed using the *Surface Remesher* option. Different *Base Sizes* (1.6 mm, 1.2 mm, 0.8 mm, 0.6 mm, 0.4 mm, 0.3 mm, 0.2 mm) were used to generate numerical meshes of varying resolution for the mesh independency study. For every *Base Size* one mesh featuring a *Prism Layer* and one without such a *Prism Layer* was created. The *Prism Layer* consisted of 3 layers with the first layer's height being 0.08 m. Each consecutive layer's height was then 1.2 times the height of the previous layer, resulting in a total *Prism Layer* thickness of 0.29 mm. Thus 14 meshes were created for the mesh independency study.

Steady state simulations of restful inspiration were performed. A negative, constant velocity equaling a volume flow rate of 12 l/min (200 ml/s) at the pharynx was specified. Both nostrils were specified as pressure outlets. Using these boundary conditions, different resistances of the left and right nasal cavity could be taken into consideration. The volume flow rate passing through each side of the nasal cavity would be defined by those resistances. It was not to be expected, that velocities within the nasal cavity would exceed a magnitude of 10 m/s. Thus, Mach numbers were below 0.1 and the inspired air could be modelled as incompressible medium. No developed turbulence can be observed within the nose during restful breathing. However, transitional turbulent regions can be found. To take those transitional effects into account a k-omega-SST turbulence model with a low turbulence intensity of two percent was used. Simulations were considered converged if residuals of continuity and all momentums were below $1.0e-5$.

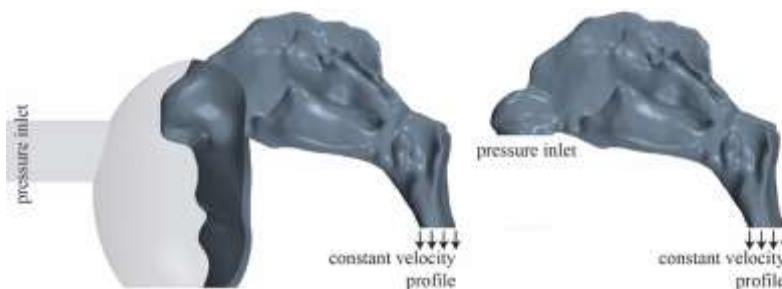


Figure 1: Comparison of the extended computational model, where part of the face and a simplified mask was attached to the nasal cavity (left) and the standard computational model, where the nasal cavity is truncated at the nostrils.

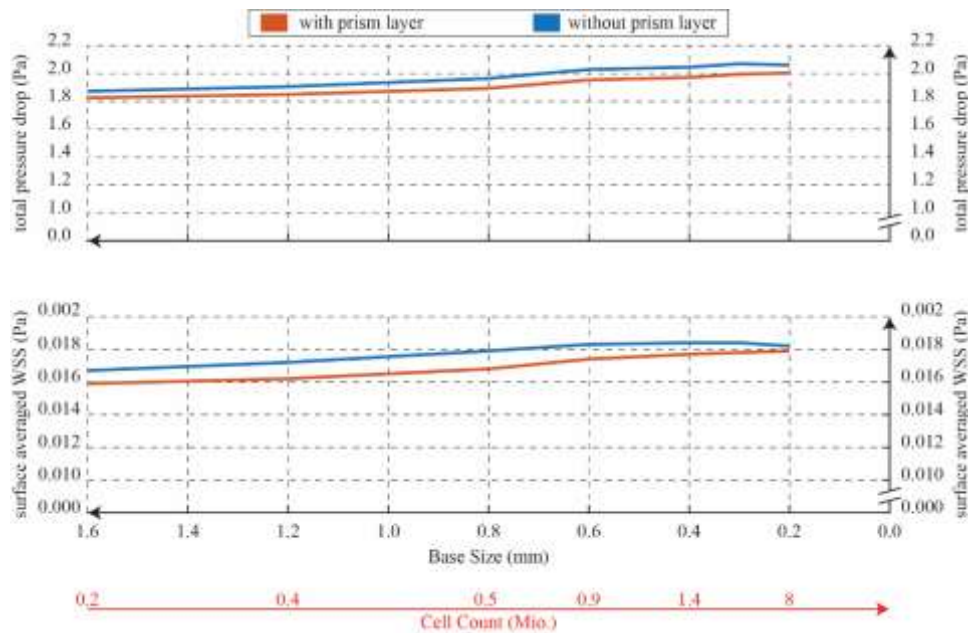


Figure 2: Total pressure drop (upper panel) and surface averaged wall shear stress (lower panel) calculated on all meshes created for the mesh independency study. The meshes' base size decreases from left to right, while the meshes' cell count increases simultaneously. The total pressure drop as well as the surface averaged wall shear stress increases with an increasing cell count, while meshes featuring a prism layer always resulted in lower values than meshes without a prism layer adjacent to the wall.

Results – Mesh Independency

To determine the resolution sufficient for obtaining mesh independent solutions the total static pressure drop was calculated as well as the surface averaged wall shear stress at the nasal cavity's wall. The total pressure drop across the nasal cavity as function of the numerical meshes' Base Size is shown in the upper panel of Figure 2. The lower the Base Size the higher the calculated pressure drop across the nasal cavity. Calculated pressure drops are higher for meshes not featuring a Prism Layer. However, all calculated values lie within close proximity to each other. The difference between the highest and the lowest calculated pressure drop is 13 percent, while the difference between pressure drops calculated on both meshes with a Base Size of 0.2 mm is only 3 percent. Therefore, the calculated total pressure drop is insensitive to different mesh resolutions. Similar trends can be observed upon evaluation of surface averaged wall shear stress (WSS) as function of the numerical meshes' Base Size. Again, no severe differences in averaged values of wall shear stresses could be discerned.

Therefore, meshes generated using a Base Size of 0.4 mm seem suited to correctly calculate integral measures as the total nasal pressure drop and thus the total nasal resistance as well as the surface averaged WSS. To ensure that not only averaged WSS values are mesh independent at a Base Size of 0.4 mm, qualitative and quantitative comparison of WSS distributions were performed. WSS values calculated on meshes with a Base Size of 0.2 mm and 0.4 mm and featuring a Prism Layer were sampled onto the original geometry obtained after segmentation. Thus, a point-wise comparison of WSS values was possible. The correlation between WSS distributions calculated using Base Size of 0.4 mm, and those using a Base Size of 0.2mm were 0.991. Even when no Prism Layer was used correlations were good (0.95).

Results – Incorporation of Ambient

Adding a simplified mask to the nasal cavity yielded in no severe differences in pressure drop. The pressure drop from mask to pharynx was 2.1 Pascal (Pa), while the pressure drop across the

truncated model was 2.2 Pa. The division of airflow to both nasal cavities wasn't influenced by incorporation of the mask either. Within the simulation using the simplified mask 56 percent of the air went through the left nasal cavity. Within the truncated model 55 percent of the air went through that side of the nose.

However, WSS distributions as well as streamlines exhibit clear differences as shown in Figure 3. While positions, where high WSS (>0.1 Pa) occur, correlate well, the shape, pattern and size of these regions differ. Especially in the vicinity of the nasal isthmus, downstream of the nostrils, truncating the nasal cavity at the nostrils led to higher wall shear stresses. Incorporation of a simplified mask led to a more chaotic flow within the nasal cavity as well. While streamlines within the truncated model are smooth and perpendicular, those streamlines in the model using a simplified mask show more variations. However, both models show the classical distribution of airflow within the nasal cavity. The highest amount of air passes through the middle meatus. This can be seen within the WSS distributions as well.

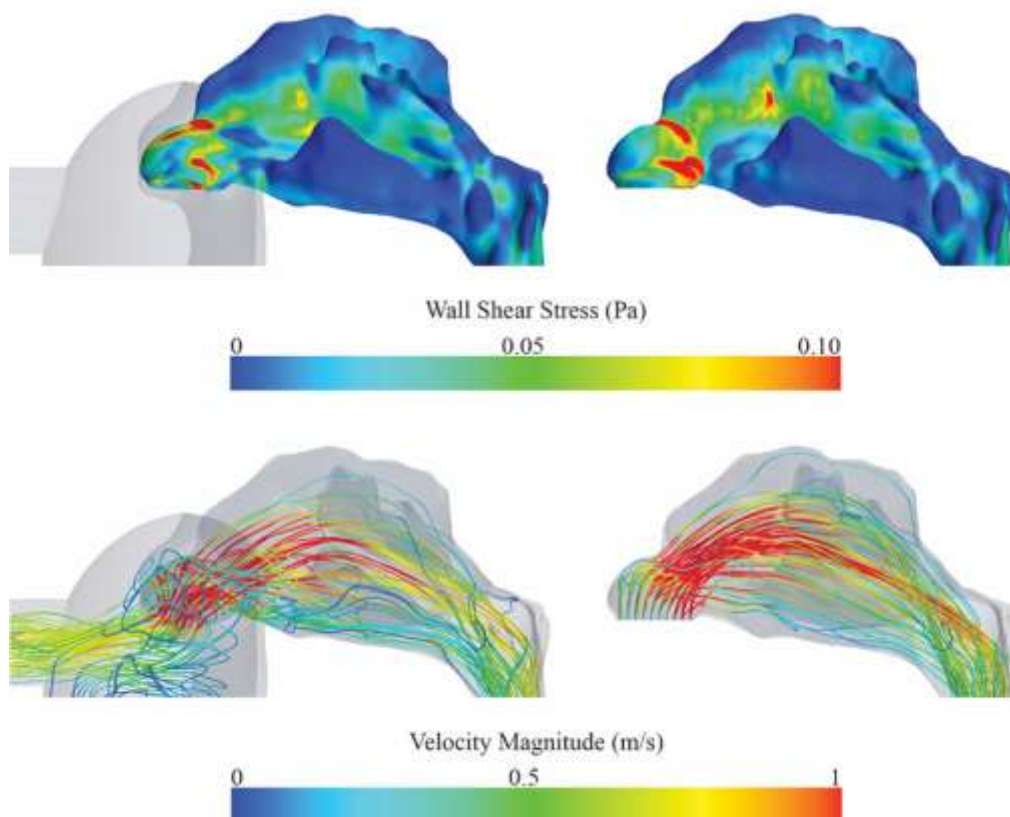


Figure 3: Wall Shear Stress distributions at the nasal cavity's wall (upper panel) and velocity information visualized using streamlines (lower panel). Those distributions are shown for simulations using a simplified mask (left) as well as for simulations, where the nasal cavity was truncated at the nostrils.

Results – Wall Clock Times and Usability of Cloud Based HPC

A dedicated machine within Microsoft's Azure Cloud was used for performing above simulations. This machine featured dual-socket Intel® Xeon® E5 processors with QDR Infiniband and RDMA technology and MPI support, allowing usage of 32 (virtual cores). Thanks to the VD-adapco STAR-CCM+ POD license provided by UberCloud, simulation of the truncated nasal geometry with highest resolution (Base Size of 0.2 mm, ca. 8 million cells) took approximately 8 hours of wall clock time. Simulation of the same geometry with the resolution shown to be sufficient within the mesh independency study (Base Size of 0.4 mm, ca. 0.9 million cells) took a little bit less than one hour of wall clock time. Therefore, within a 24 hour session, 20 or more geometries could be calculated. The

simulation of the nasal cavity being attached to a simplified mask (Base Size 0.4 mm, ca. 2 million cells) couldn't be finished within the 12 hour POD time frame. However, estimated by simulation on a local machine, convergence would have been reached after approximately 6 hours of wall clock time. This relatively long duration when compared to both other simulations is due to the fact that no convergence was reached using a steady state solver, demonstrating the necessity to switch to an implicit unsteady solver using steady boundary conditions.

DISCUSSION AND BENEFITS

The overall experience using UberCloud's integrated STAR-CCM+ container environment on the Azure cloud was very convincing.

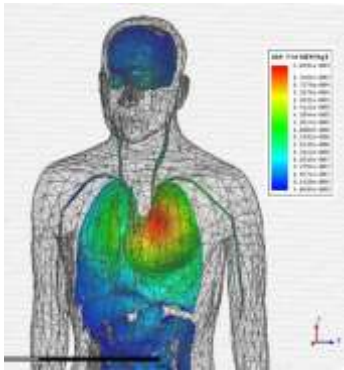
- Handling of the overall cloud environment was straight-forward and due to the whole setup being browser-based no complications regarding application software requirements occurred. Simulation files were uploaded using the author's institution's OwnCloud Service. However, an upload using Dropbox would have been possible as well, since a Dropbox client was already installed on the machine.
- Simulation speeds were overwhelmingly fast compared to the workstations the authors usually work with. Handling was pretty much the same as on a local computer. An hourly screenshot of the cloud machine's state and emailed log files allowed monitoring of the simulation state without any need to log into the cloud.
- The relatively small wall clock time necessary for simulation of one nasal cavity is very promising since it allows short rent times for cloud-based machines as well as software licenses. Thus, simulation of a patient's nasal cavity as a diagnostic tool might be performed relatively cheap. However, at this time, it is totally unknown what a good nasal airflow is and which airflow patterns and phenomena are related to impairment of nasal breathing. Within the near future, patient-specific computation of nasal airflow might become a relevant diagnostic tool within otolaryngology.

The difference between WSS and velocity distributions within the nasal cavity might indicate that additional research is necessary to better understand how wearing a mask alters the airflow. As of yet, several numerical studies including the ambient were conducted. However, all of these studies used an unobstructed ambient. This preliminary investigation about including the mask resulted in no severe change in the pressure drop across the nasal cavity. This has to be further investigated to ensure that rhinomanometric measurements, where a similar mask is worn by the patient, do not alter the airflow resistance of the nasal cavity by altering the inflow conditions.

Case Study Author – Jan Bruening, Charité Berlin, Germany

Team 193

Implantable Planar Antenna Simulation with ANSYS HFSS in the Nephoscale Cloud



“ANSYS HFSS in UberCloud’s application software container provided an extremely user-friendly on-demand computing environment very similar to my own desktop workstation.”

MEET THE TEAM

End user – Mehrnoosh Khabiri, Ozen Engineering, Inc. Sunnyvale, California

Team Expert – Metin Ozen, Ozen Engineering, Inc. and Burak Yenier, UberCloud, Inc.

Software Provider – Ozen Engineering, Inc. and UberCloud, Inc.

Resource Provider – Nephoscale Cloud, California.

Use Case

In recent years, with rapid development of wireless communication technology, Wireless Body Area Networks (WBANs) have drawn a great attention. WBAN technology links electronic devices on and in the human body with exterior monitoring or controlling equipment. The common applications for WBAN technology are biomedical devices, sport and fitness monitoring, body sensors, mobile devices, and so on. All of these applications have been categorized in two main areas, namely medical and non-medical, by IEEE 802.15.6 standard. For medical applications, the wireless telemetric links are needed to transmit the diagnostic, therapy, and vital information to the outside of human body. The wide and fast-growing application of wireless devices yields to a lot of concerns about their safety standards related to electromagnetic radiation effects on human body. Interaction between human body tissues and Radio Frequency (RF) fields are important. Many researches have been done to investigate the effects of electromagnetic radiation on human body. The Specific Absorption Rate (SAR), which measures the electromagnetic power density absorbed by the human body tissue, is considered as an index by standards to regulate the amount of exposure of the human body to electromagnetic radiation.

In this case study implantable antennas are used for communication purposes in medical devices. Designing antennas for implanted devices is an extremely challenging task. The antennas require to be small, low profile, and multiband. Additionally, antennas need to operate in complex environments. Factors such as small size, low power requirement, and impedance matching play significant role in the design procedure. Although several antennas have been proposed for implantable medical devices, the accurate full human body model has been rarely included in the simulations. An implantable Planar Inverted F Antenna (PIFA) is proposed for communication between implanted medical devices in human body and outside medical equipment. The main aim

of this work is to optimize the proposed implanted antenna inside the skin tissue of human body model and characterize the electromagnetic radiation effects on human body tissues as well as the SAR distribution. Simulations have been performed using ANSYS HFSS (High-Frequency Structural Simulator) which is based on the Finite Element Method (FEM), along with ANSYS Optimetrics and High-Performance Computing (HPC) features.

ANSYS HUMAN BODY MODEL AND ANTENNA DESIGN

ANSYS offers the adult-male and adult-female body models in several geometrical accuracy in millimeter scale [17]. Fig. 1 shows a general view of the models. ANSYS human body model contains over 300 muscles, organs, tissues, and bones. The objects of the model have geometrical accuracy of 1-2 mm. The model can be modified by users for the specific applications and parts, and model objects can simply be removed if not needed. For high frequencies, the body model can be electrically large, resulting in huge number of meshes which makes the simulation very time-consuming and computationally complex. The ANSYS HPC technology enables parallel processing, such that one has the ability to model and simulate very large size and detailed geometries with complex physics.

The implantable antenna is placed inside the skin tissue of the left upper chest where most pacemakers and implanted cardiac defibrillators are located, see Figure 1. Incorporating ANSYS Optimetrics and HPC features, optimization iterations can be performed in an efficient manner to simulate the implantable antenna inside the human body model.

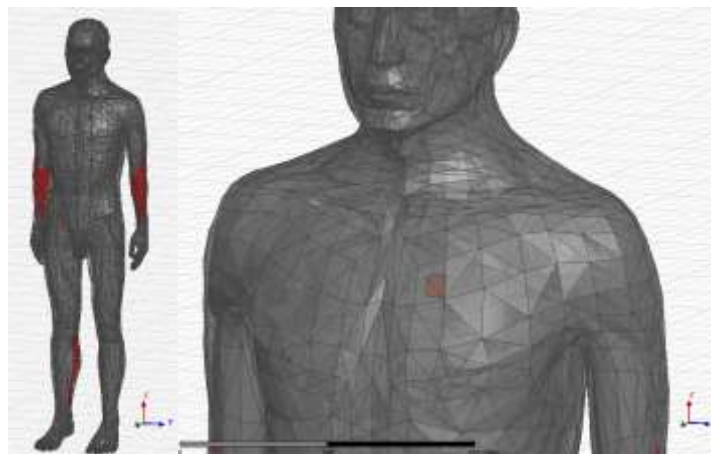


Figure 1: Implanted antenna in ANSYS male human body model.

The antenna is simulated in ANSYS HFSS which is a FEM electromagnetic solver. Top and side view of proposed PIFA is illustrated in Figure 2 (left), the 3D view of the implantable PIFA is demonstrated in Figure 2 (right). The thickness of dielectric layer of both substrate and superstrate is 1.28 mm. The length and width of the substrate and superstrate are $L_{sub}=20\text{mm}$ and $W_{sub}=24\text{mm}$, respectively. The width of each radiating strip is $W_{strip}=3.8\text{mm}$. The other parameters of antenna are considered to be changed within the solution space in order to improve the PIFA performance. HFSS Optimetrics, an integrated tool in HFSS for parametric sweeps and optimizations, is used for tuning and improving the antenna characteristics inside the ANSYS human body model.

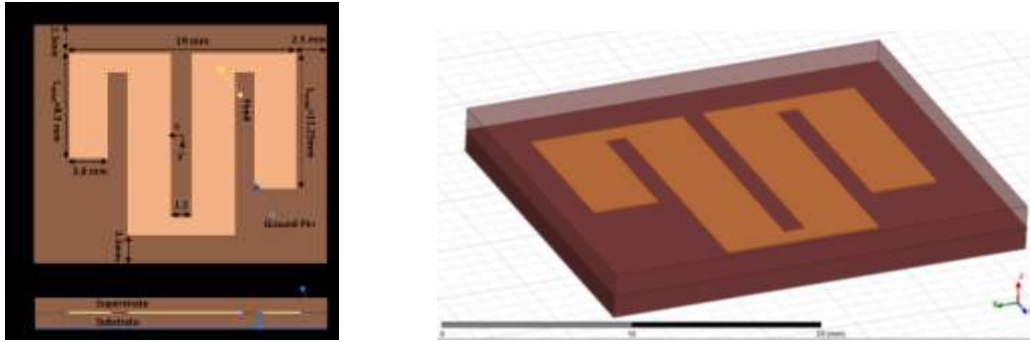


Figure 2: Top and side view of PIFA (left) and 3D view of PIFA geometry in HFSS (right).

RESULTS AND ANALYSIS

Figure 3 illustrates the far-field radiation pattern of the proposed PIFA at 402 MHz. Since the antenna is electrically small and the human body provides a lossy environment, the antenna gain is very small (~ -44 dBi) and the EM fields are reactively stored in the human body parts in vicinity.

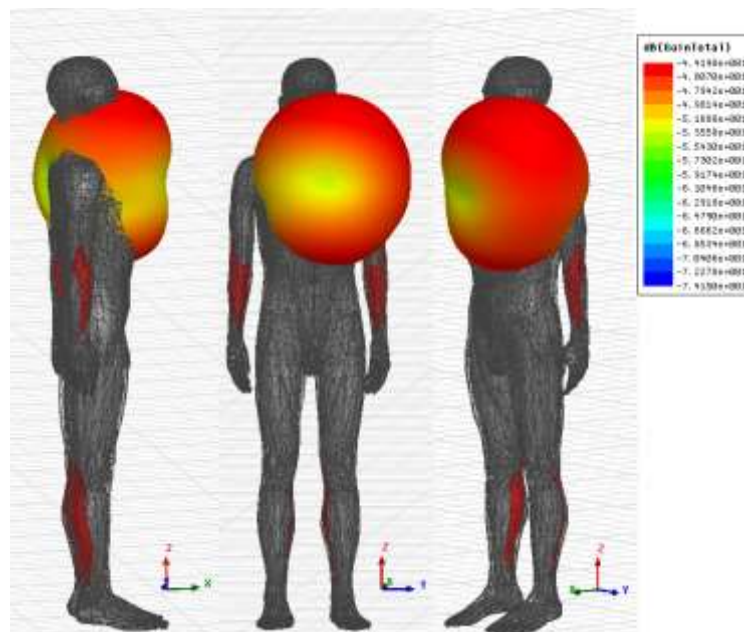


Figure 3: 3D Radiation pattern of implanted PIFA inside the human body model.

Figure 4 shows the simulated electric field distributions around the male human body model at 402 MHz center frequency. The electric field magnitude is large at upper side of the body, and it becomes weaker as going far away from the male body chest.

The electromagnetic power absorbed by tissues surrounding the antenna inside the human body model is a critical parameter. Hence, SAR analysis is required to evaluate the antenna performance. SAR measures the electromagnetic power density absorbed by the human body tissue. SAR measurement makes it possible to evaluate if a wireless medical device satisfies the safety limits.

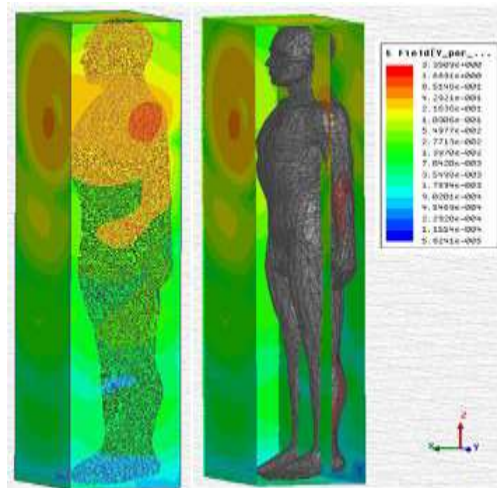


Figure 4: Electric Field distribution around male body model at 402 MHz.

SAR is averaged either over the whole body or a small volume (typically 1 g or 10 g of tissue). ANSYS HFSS offers SAR calculations according to standards. The 3D plots of the local SAR distribution are shown in Figure 5 and Figure 6. In Figure 5, the detailed male body model with heart, lungs, liver, stomach, intestines, and brain are included. It can be observed that the left upper chest region where SAR is significant is relatively small. The peak SAR of the PIFA is smaller than the regulated SAR limitation. Figure 5 shows the SAR distribution on the skin tissue of the full human body model.

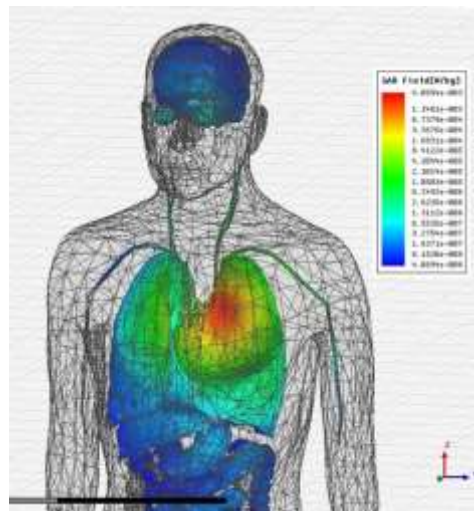


Figure 5: Local SAR distribution on upper side of male body model at 402 MHz.

A more detailed discussion of this use case by Mehrnoosh Khabiri can be found in the Ozen Engineering white paper [“Design and Simulation of Implantable PIFA in Presence of ANSYS Human Body Model for Biomedical Telemetry Using ANSYS HFSS”](#).

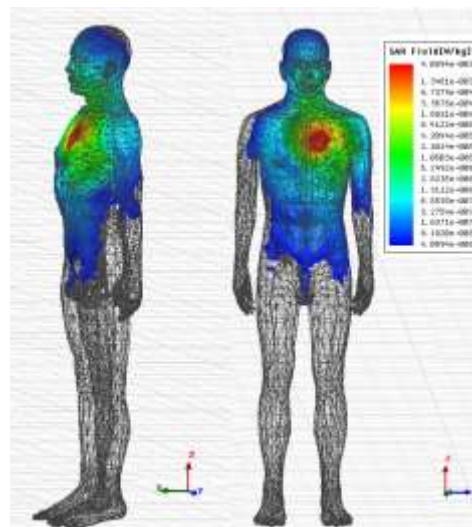


Figure 6: Local SAR distribution on the skin tissue of male body model at 402 MHz.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

CONCLUSIONS

Design modification and tuning of antenna performance were studied with the implantable antenna placed inside the skin tissue of ANSYS human body model. The resonance, radiation, and Specific Absorption Rate (SAR) of implantable PIFA were evaluated. Simulations were performed with ANSYS HFSS (High-Frequency Structural Simulator) which is based on Finite Element Method (FEM). All simulations have been performed on a 40-core Nephoscale cloud server with 256 GB RAM. These simulations were about 4 times faster than on the local 16-core desktop workstation.

ANSYS HFSS has been packaged in an UberCloud HPC software container which is a ready-to-execute package of software designed to deliver the tools that an engineer needs to complete his task in hand. In this experiment, ANSYS HFSS has been pre-installed, configured, and tested, and running on bare metal, without loss of performance. The software was ready to execute literally in an instant with no need to install software, deal with complex OS commands, or configure.

This technology also provides hardware abstraction, where the container is not tightly coupled with the server (the container and the software inside isn't installed on the server in the traditional sense). Abstraction between the hardware and software stacks provides the ease of use and agility that bare metal environments lack.

Case Study Author: Mehrnoosh Khabiri, Ozen Engineering, and Wolfgang Gentzsch, The UberCloud

Team 195

Simulation of Impurities Transport in a Heat Exchanger Using OpenFOAM



Figure 1: 3D model picture of the initial version of heat exchanger, with coil inside.

"I've been using cloud computing for several years now, tried at least four different cloud providers and found the UberCloud service by far the best. I didn't expect it would be SO easy to use."

MEET THE TEAM

End-User/CFD Expert: Eugeny Varseev, Central Institute for Continuing Education & Training (ROSATOM-CICE&T.), Obninsk, Russia

Software Provider: Lubos Pirkli, CFD Support, with *OpenFOAM in Box* hosted in an UberCloud software container, Prague, Czech Republik

Resource Provider: Aegir Magnusson, Per-Ola Svensson, Hans Rickardt, Advania, Iceland

Cloud Expert: Hilal Zitouni, Fetican Coskuner, The UberCloud, Izmir, Turkey.

USE CASE

In this case study the numerical simulation of the of the impurities transport in a heat exchanger designed for coolant purification was performed using CFD Support's *OpenFOAM in Box v16.10* packaged in an UberCloud software container and hosted on the Advania Cloud. The transient process of the purification trap operation was simulated in to find the process stabilization time.

Almost any power equipment requires to maintain some level of coolant purity to provide the most reliable, effective way of operation. Studying the characteristics of the purification trap considered within this simulation is driven by the need to sustain the number of the impurities at a reasonably low level to keep the equipment of the circuit from fouling and heat transfer deterioration.

The study was performed at two general stages: first, the steady-state thermal hydraulic simulation of coolant flow pattern inside the heat exchanger was done using standard OpenFOAM capabilities on the local desktop. Second, the transient simulation of both dissolved impurities and crystalized particulates transport was performed using a custom OpenFOAM transport solver hosted in an UberCloud OpenFOAM software container.

METHOD

The simulation case was locally prepared on the engineer's desktop and based on a CAD model created using the Salome software. Meshing was done by means of the snappyHexMesh utility. The model is a cylinder with an inlet tube inserted inside and an asymmetrically located outlet pipe at

the top (see Figure 2). During the first stage of the study, which is computationally less demanding, a number of thermal hydraulic simulation runs were performed to determine the optimal mesh size of the model, which is between 0.1 M, 0.9M and 1.5M of hexahedral cells.

For the next stage, a custom OpenFOAM solver has been designed to consider the crystallization of dissolved impurity occurring due to coolant temperature decrease.

The formula of the impurity transport equation can be represented in the following mathematical way:

$$\frac{dC_i}{d\tau} + \text{div}(vC_i) - \text{div}\left(\left(\frac{\nu}{Sc} + \frac{\nu_t}{Sc_t}\right)\text{grad}C_i\right) = Q_i,$$

where C = impurity concentration, ppm; and index «i» means dissolved and crystallized phases; u = coolant velocity, m/s; ν and ν_t = viscosity and turbulent viscosity, m^2/s ; Sc and Sc_t = Schmidt number and turbulent Schmidt number; Q = source of concentration in the cell (dissolution of crystallization), ppm.

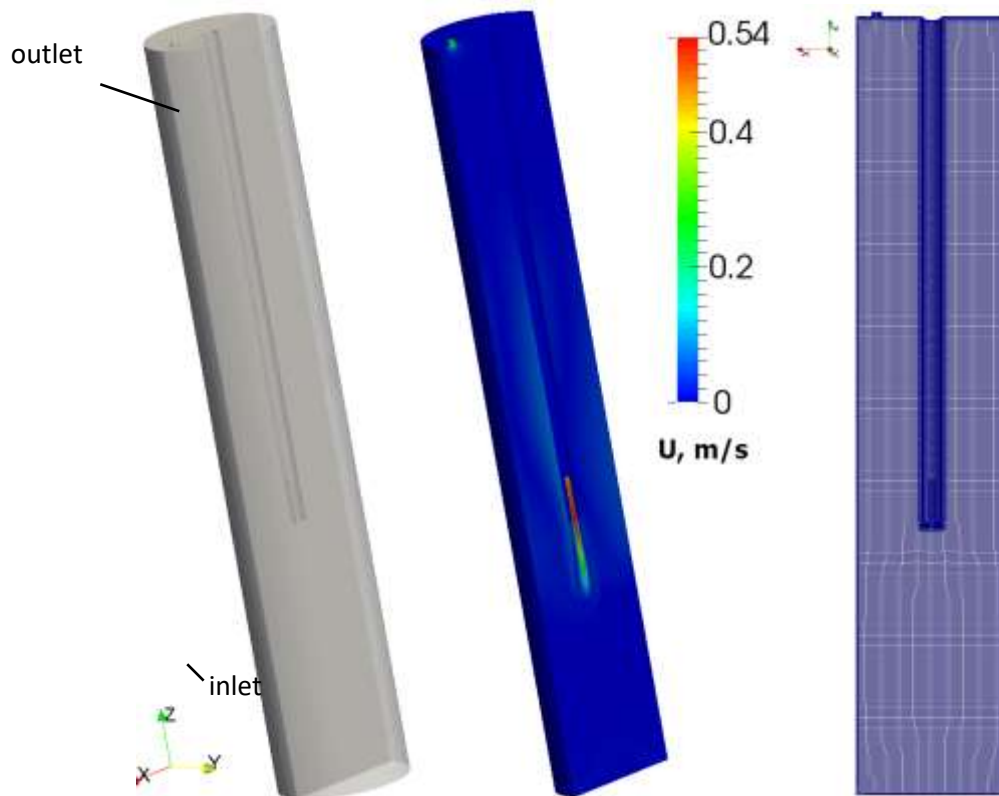


Figure 2. Symmetrical half of the CAD model, steady-state velocity field, and mesh of the model.

The custom computational model considers additional phenomena, such as:

- If the value of concentration in the given cell is less than that of the impurity concentration of the saturation ($C < C_s$), the value of saturated impurity concentration is set equal to the saturation concentration and surplus concentration transforms to particulate phase with concentration C_p .
- The reverse process of impurity dissolution.

The validation and verification of the custom solver based on experimental data of mass transfer in pipes preceded the simulation runs.

After the custom solver was ready to use, it was uploaded into the UberCloud container, precompiled for OpenFOAM v3.0, and moved into a folder for user solvers, and then it was ready to run right away.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

SIGNIFICANT CHALLENGES

The stabilization time of the purification process is in the order of dozens of hours of real life time, so for transient simulation with time step value in the order of 0.001 sec using several millions of cells the purification simulation is definitely very time consuming. With the power of HPC, however, reducing simulation time dramatically, allows for studying models with less simplifications.

RESULTS AND DISCUSSION

The simulations were running on Advania cloud resources, on one dual-socket compute node with 2x Intel E5-2680 v3 processors, 24 cores, and 16GB of memory. The UberCloud software container on these resources allowed for getting at least a 15 times performance increase compared with the desktop system used for preparing this use case.

The spacial distribution of the dissolved impurity and particulates inside the purification trap was obtained as a result of the simulation. The analysis of the dissolved and precipitated concentration fields allowed obtaining mass transfer characteristics of the device.

The time it takes to stabilize the process was obtained from computation results by means of the ParaView post-processing and visualization right in the cloud and presented in Figure 3.

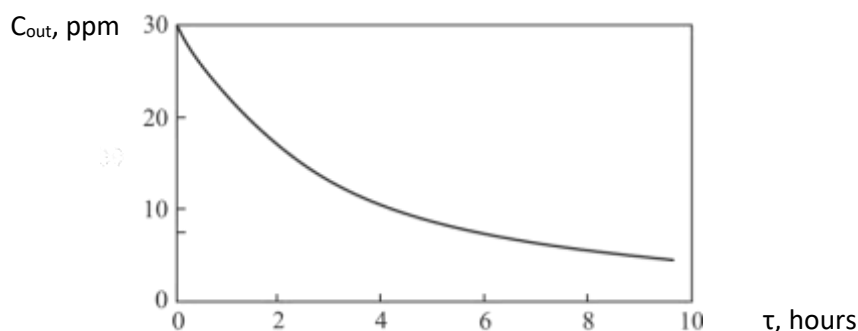


Figure 3. Concentration at the outlet of the model via time.

BENEFITS

The calculation in the cloud using an UberCloud OpenFOAM container allowed us to get the necessary result in rapid turn-around time – in less than 8 hours. This means a user can get his simulation results instead in days on his desktop now in just one night instead.

The ease-of-use experience was due to the fact that it takes virtually no time to adjust to the remote workspace offered by the UberCloud OpenFOAM container, because it looks and feels as if the user were doing the simulation on his personal Linux-based desktop. There are no special commands and configuration scripts to run.

The post-processing of files doesn't require downloading big chunks of data back to the user's computer – they simply were post-processed and analyzed right in the cloud by means of the tools the user is used to without any limitations. And for really big simulation cases this is especially important because they require huge computation power not only for the calculation but for the post-processing as well.

For file managing it's possible to use conventional cloud resources, which are much more comfortable to use than FTP file managers, for example.

CONCLUSION

Transient simulation of purification trap device operation was performed in the cloud using CFD Support's *OpenFOAM in Box v16.10* hosted in an UberCloud software container on Advania cloud resources. The time of the purification process stabilization was calculated with a 15 times computational performance advantage in comparison with the user's personal desktop system used for the preparation of the use case.

The whole simulation process (mesh preparation, the simulation itself, and post-processing) has been done within the software container in the cloud, using automation and common post-processing scripts. It allows performing CFD studies and parametrical analysis of the models very quickly and as if a user just were using another workspace remotely.

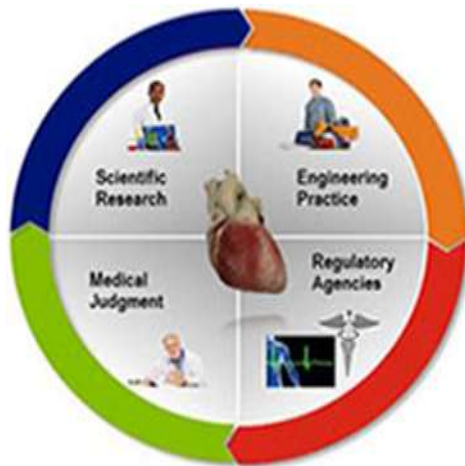
ACKNOWLEDGMENT

Authors are very thankful to the IPPE Sodium laboratory team. Special thanks to F.A. Kozlov, Yu.I. Zagorulko, V.V. Alexeev, and C. Latge for helpful discussions of the topic.

Case Study Author – Eugeny Varseev

Team 197

Studying Drug-induced Arrhythmias of a Human Heart with Abaqus 2017 in the Cloud



"We were able to easily access sufficient HPC resources to study drug-induced arrhythmias in a reasonable amount of time. With our local machines, with just 32 CPU cores, these simulations would have been impossible."

MEET THE TEAM

End User – Francisco Sahli Costabal, PhD Candidate, and Prof. Ellen Kuhl, Living Matter Laboratory at Stanford University.

Software Provider – Dassault/SIMULIA (Tom Battisti, Matt Dunbar) providing Abaqus 2017 software and support.

Resource Provider – Advania Cloud in Iceland (represented by Aegir Magnusson and Jon Tor Kristinsson), with access and support for the HPC server from HPE.

HPC Cloud Experts – Fethican Coskuner and Wolfgang Gentsch, UberCloud, with providing novel HPC container technology for ease of Abaqus cloud access and use.

Sponsor – Hewlett Packard Enterprise, represented by Stephen Wheat, Bill Mannel, Jean-Luc Assor.

"Our successful partnership with UberCloud has allowed us to perform virtual drug testing using realistic human heart models. For us, UberCloud's high-performance cloud computing environment and the close collaboration with HPE, Dassault, and Advania, were critical to speed-up our simulations, which help us to identify the arrhythmic risk of existing and new drugs in the benefit of human health."

Prof. Ellen Kuhl, Head of Living Matter Laboratory at Stanford University

USE CASE

This cloud experiment for the Living Heart Project (LHP) is a follow-on work of Team 196 first dealing with the implementation, testing, and Proof of Concept in the Cloud. It has been collaboratively performed by Stanford University, SIMULIA, Advania, UberCloud, and sponsored by Hewlett Packard Enterprise. It is based on the development of a Living Heart Model that encompasses advanced electro-physiological modelling. The goal is to create a biventricular finite element model to study drug-induced arrhythmias of a human heart.

The **Living Heart Project** is uniting leading cardiovascular researchers, educators, medical device developers, regulatory agencies, and practicing cardiologists around the world on a shared mission to develop and validate highly accurate personalized digital human heart models. These models will establish a unified foundation for cardiovascular in silico medicine and serve as a

common technology base for education and training, medical device design, testing, clinical diagnosis and regulatory science —creating an effective path for rapidly translating current and future cutting-edge innovations directly into improved patient care.

Cardiac arrhythmias can be an undesirable and potentially lethal side effect of drugs. During this condition, the electrical activity of the heart turns chaotic, decimating its pumping function, thus diminishing the circulation of blood through the body. Some kind of arrhythmias, if not treated with a defibrillator, will cause death within minutes.

Before a new drug reaches the market, pharmaceutical companies need to check for the risk of inducing arrhythmias. Currently, this process takes years and involves costly animal and human studies. With this new software tool, drug developers would be able to quickly assess the viability of a new compound. This means better and safer drugs reaching the market to improve patients' lives.

The Stanford team in conjunction with SIMULIA have developed a multi-scale 3-dimensional model of the heart that can predict the risk of this lethal arrhythmias caused by drugs. The project team added several capabilities to the Living Heart Model such as highly detailed cellular models, the ability to differentiate cell types within the tissue and to compute electrocardiograms (ECGs). A key addition to the model is the so-called [Purkinje](#) network. It presents a tree-like structure and is responsible of distributing the electrical signal quickly through the ventricular wall. It plays a major role in the development of arrhythmias, as it is composed of pacemaker cells that can self-excite. The inclusion of the Purkinje network was fundamental to simulate arrhythmias. This model is now able to bridge the gap between the effect of drugs at the cellular level to the chaotic electrical propagation that a patient would experience at the organ level.

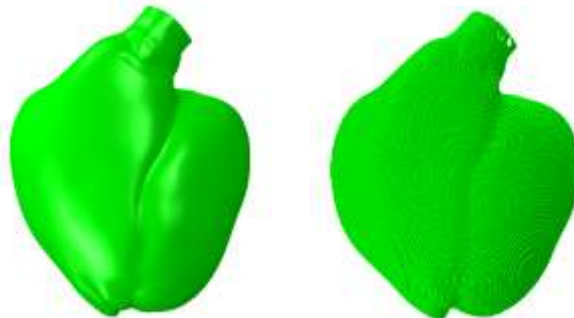


Figure 1: Tetrahedral mesh (left) and cube mesh (right).

A computational model that is able to assess the response of new drug compounds rapidly and inexpensively is of great interest for pharmaceutical companies, doctors, and patients. Such a tool will increase the number of successful drugs that reach the market, while decreasing cost and time to develop them, and thus help hundreds of thousands of patients in the future. However, the creation of a suitable model requires taking a multiscale approach that is computationally expensive: the electrical activity of cells is modelled in high detail and resolved simultaneously in the entire heart. Due to the fast dynamics that occur in this problem, the spatial and temporal resolutions are highly demanding.

During the preparation and Proof of Concept phase (UberCloud Experiment 196) of this LHP project, we set out to build and calibrate the healthy baseline case, which we then used to perturb with different drugs. After creating the [UberCloud container](#) for SIMULIA's Abaqus 2017 and deploying it on HPE's server in the Advantia cloud, we started refining the computational mesh which consisted of roughly 5 million tetrahedral elements and 1 million nodes. Due to the intricate geometry of the heart, the mesh quality limited the time step, which in this case was 0.0012 ms for a total simulation time of 5000 ms. After realizing that it would be very difficult to calibrate our model with such a big

runtime, we decided to work on our mesh, which was the current bottleneck to speed up our model. We created a mesh that was made out of cube elements (Figure 1). With this approach, we lost the smoothness of the outer surface, but reduced the number of elements by a factor of ten and increased the time step by a factor of four, for the same element size (0.7 mm).



Figure 2: The final production model with an element size of 0.3 mm. The Purkinje network is shown in white. Endocardial, mid layer and epicardial cells are shown in red, white and blue respectively.

After adapting all features of the model to this new mesh with now 7.5 million nodes and **250,000,000 internal variables that are updated and stored within each step of the simulation** (Figure 2), we were able to calibrate the healthy, baseline case, which was assessed by electrocardiogram (ECG) tracing (Figure 3) that recapitulates the essential features.

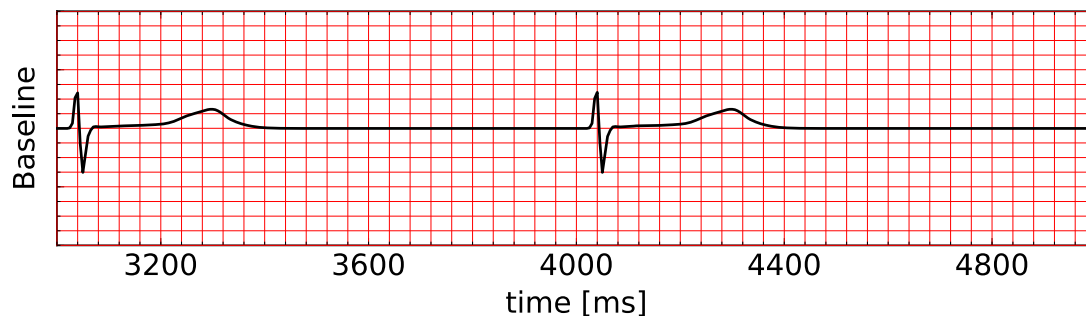


Figure 3: ECG tracing for the healthy, baseline case.

During the final production phase, we have run 42 simulations to study whether a drug causes arrhythmias or not. With all these changes we were able to **speed up one simulation by a factor of 27** which then (still) took 40 hours using 160 CPU cores on Advania's HPC as a Service (HPCaaS) hardware configuration built upon HPE ProLiant servers XL230 Gen9 with 2x Intel Broadwell E5-2683 v4 with Intel OmniPath interconnect. We observed that the model scaled without a significant loss of performance up to 240 compute cores, making the 5-node sub-cluster of the Advania system an ideal candidate to run these compute jobs. In these simulations, we applied the drugs by blocking different ionic currents in our cellular model, exactly replicating what has been observed before in cellular experiments. For each case, we let the heart beat naturally and see if the arrhythmia is developing.

Figure 4 shows the application of the drug Quinidine, which is an anti-arrhythmic agent, but it has a high risk of producing [Torsades de Pointes](#), which is a particular type of arrhythmia. It shows the electrical transmembrane potentials of a healthy versus a pathological heart that has been widely used in studies of normal and pathological heart rhythms and defibrillation. The propagation of the electrical potential turns chaotic (Figure 4, bottom) when compared to the baseline case (Figure 4, top), showing that our model is able to correctly and reliably predict the anti-arrhythmic risk of

commonly used drugs. We envision that our model will help researchers, regulatory agencies, and pharmaceutical companies rationalize safe drug development and reduce the time-to-market of new drugs.

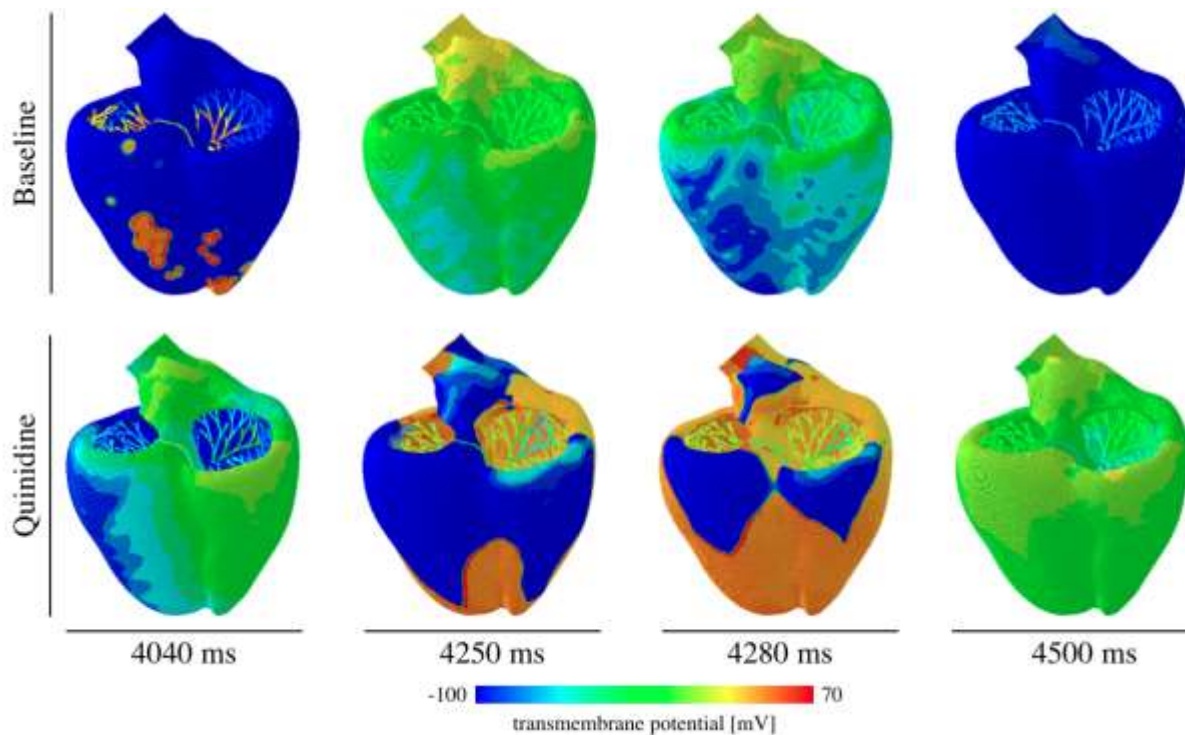


Figure 4: Evolution of the electrical activity for the baseline case (no drug) and after the application of the drug Quinidine. The electrical propagation turns chaotic after the drug is applied, showing the high risk of Quinidine to produce arrhythmias.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux [Docker container](#) technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

Some of the challenges that we faced during the project were:

- Although the remote desktop setup enabled us to visualize the results of our model, it was not possible to do more advanced operations. The bandwidth between the end user and the servers was acceptable for file transfer, but not enough to have a fluid remote desktop. We suggested to speed-up remote visualization which has now been implemented including NICE Software's DCV into the UberCloud software container, making use of GPU accelerated data transfers.
- Running the final complex simulations first on the previous-generation HPC system at Advania took far too long and we would have not been able to finish the project in time. Therefore, we moved our Abaqus 2017 container seamlessly to the new HPC system (which

was set up in July 2017) and got an immediate speedup of 2.5 between the two HPE systems.

Some of the benefits that we experienced:

- Gaining easy and intuitive access to sufficient HPC resources enabled us to study drug-induced arrhythmias of a human heart in a reasonable amount of time. With our local machines, with just 32 CPU cores, these simulations would have been impossible.
- As we had a dedicated 5-node HPC cluster in the cloud, it was easy to run post-processing scripts, without the need of submitting a second job in the queue, which would be the typical procedure of a shared HPC resource.
- Since all project partners had access to the same Abaqus 2017 container on the HPC server, it was easy to jointly debug and solve problems as a team. Also, sharing models and results between among the end user and the software provider was straight-forward.
- The partnership with UberCloud has allowed us to perform virtual drug testing using realistic human heart models. For us, UberCloud's high-performance cloud computing environment and the close collaboration with HPE, Dassault, and Advania, were critical to speed-up our simulations, which help us to identify the arrhythmic risk of existing and new drugs in the benefit of human health.

Case Study Author – Francisco Sahli Costabal together with Team 197.

Appendix

This research has been presented at the Cardiac Physiome Society Conference in Toronto November 6 – 9, 2017, <https://www.physiome.org/cardiac2017/index.html>:

Predicting drug-induced arrhythmias by multiscale modeling

Francisco Sahli Costabal, JIang Yao, Ellen Kuhl

Abstract: Drugs often have undesired side effects. In the heart, they can induce lethal arrhythmias such as Torsades de Points. The risk evaluation of a new compound is costly and can take a long time, which often hinders the development of new drugs. Here we establish an ultra high resolution, multiscale computational model to quickly and reliably assess the cardiac toxicity of new and existing drugs. The input of the model is the drug-specific current block from single cell electrophysiology; the output is the spatio-temporal activation profile and the associated electrocardiogram. We demonstrate the potential of our model for a low risk drug, Ranolazine, and a high risk drug, Quinidine: For Ranolazine, our model predicts a prolonged QT interval of 19.4% compared to baseline and a regular sinus rhythm at 60.15 beats per minute. For Quinidine, our model predicts a prolonged QT interval of 78.4% and a spontaneous development of Torsades de Points both in the activation profile and in the electrocardiogram. We also study the dose-response relation of a class III antiarrhythmic drug, Dofetilide: At low concentrations, our model predicts a prolonged QT interval and a regular sinus rhythm; at high concentrations, our model predicts the spontaneous development of arrhythmias. Our multiscale computational model reveals the mechanisms by which electrophysiological abnormalities propagate across the spatio-temporal scales, from specific channel blockage, via altered single cell action potentials and prolonged QT intervals, to the spontaneous emergence of ventricular tachycardia in the form of Torsades de Points. We envision that our model will help researchers, regulatory agencies, and pharmaceutical companies to rationalize safe drug development and reduce the time-to-market of new drugs.

Team 199

HPC Cloud Performance of Peptide Benchmark Using LAMMPS Molecular Dynamics Package

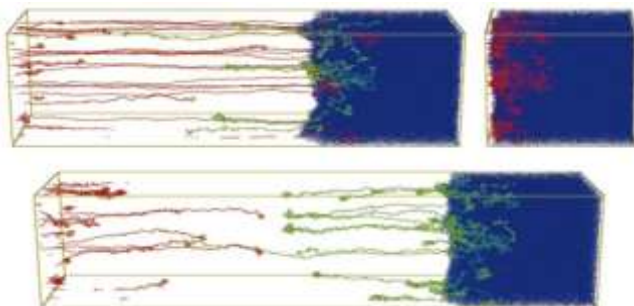


Figure 1: Simulation snapshots using LAMMPS, studying adhesion dynamics for surface-tethered chains entangled in polymer melt.

“HPC software container-based cloud computing is an easy process compared to building and maintaining your own cluster in the cloud.”

MEET THE TEAM

End User – National Renewable Energy Lab (NREL), Tech-X Research

Software Provider – LAMMPS open source software and Steven J. Plimpton (Sandia National Lab)

Resource Provider – Amazon Web Services (AWS)

HPC Experts – Dr. Scott W. Sides, Senior Scientist, Tech-X Research Boulder, CO, Fethican Coskuner and Ender Guler, The UberCloud.

USE CASE

To address realistic problems in nanomaterials and pharmaceutical industries, large-scale molecular dynamics (MD) simulations must be able to fully utilize high-performance computing (HPC) resources. Many small- and medium-sized industries that could make use of MD simulations do not use HPC resources due to the complexity and expense of maintaining in-house computing clusters.

Cloud computing is an excellent a way of providing HPC resources to an underserved sector of the simulation market. In addition, providing HPC software containers with advanced application software can make the use of these codes more straightforward and further reduce the barriers for entry to small- and medium-sized businesses.

The molecular dynamics package LAMMPS is widely used in academia and some industries. LAMMPS has potentials for solid-state materials (metals, semiconductors) and soft matter (biomolecules, polymers) and coarse-grained or mesoscopic systems. It can be used to model atoms or, more generically, as a parallel particle simulator at the atomic, meso, or continuum scale.

The cloud service provider, Amazon Web Services, provided a number of virtual machines each with up to 16 cores for this experiment with different levels of network communication performance.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically,

from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

Technical Details of the Simulation

Figure 2 shows the parallel scaling performance of the UberCloud LAMMPS containers running on an AWS multi-node cluster with each of the nodes having 16 cores available. A simple peptide chain model that is included in the tests for LAMMPS was used for performance scaling. The initial peptide input file only contains 2004 particles, but using the 'replicate' keyword available in LAMMPS the initial simulation cell may be copied in the x,y,z directions an arbitrary number of times.

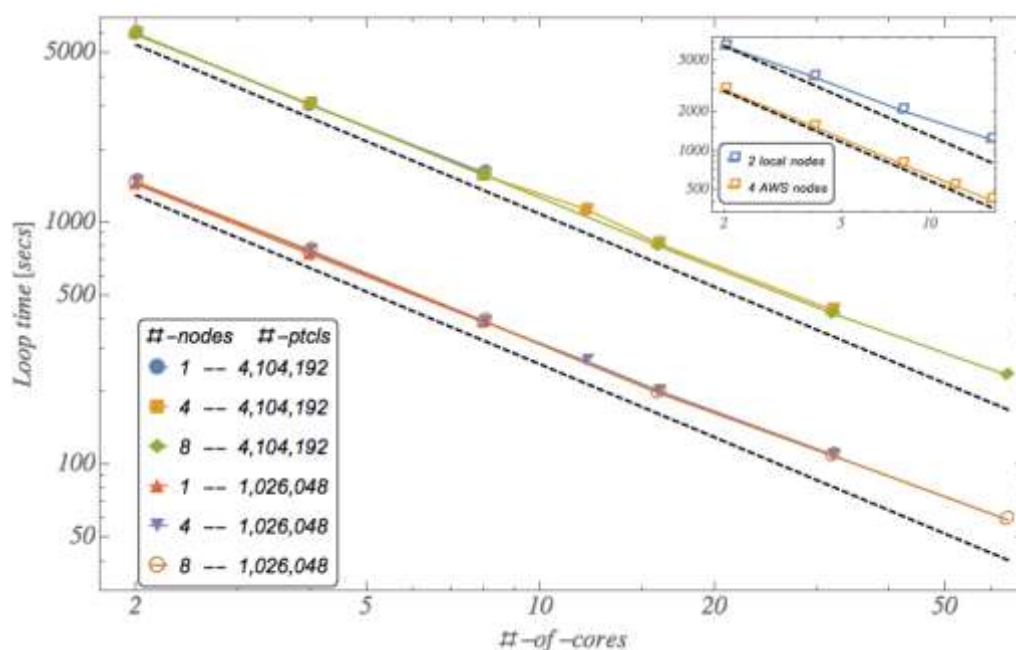


Figure 2: LAMMPS parallel scaling performance on an AWS multi-node cluster with each of the nodes having 16 cores available. Inset upper right: Comparison of the parallel scaling performance between LAMMPS running on the bare-metal 2-node test cluster at Tech-X and LAMMPS containers running on a 4-node remote AWS cluster. The dotted lines indicate the optimal scaling behavior, showing that the performance of the LAMMPS containers running in the cloud is excellent.

The simulations in Figure 2 show two system sizes using $\approx 10^6$ and $\approx 4.1 \cdot 10^6$ particles run for 300 update steps for reasonable timing statistics. The inset in the upper right shows a comparison of the parallel scaling performance for a system with $\approx 2.0 \cdot 10^6$ particles between LAMMPS running on the bare-metal 2-node test cluster at Tech-X and UberCloud LAMMPS containers running on a 4-node remote AWS cluster. The dotted line in the main figure and inset is the optimal scaling trend. The main figure shows that the LAMMPS multi-node container performance persists as the number of nodes in the cloud cluster increases. There was degraded performance when the number of processors/node reaches the maximum number of cores available as listed by AWS and is due to hyper-threading. But, there appears to be no degradation of performance as the size of the cluster increased, suggesting that an arbitrary number of processors can be used for HPC molecular dynamics simulations using LAMMPS in the cloud.

Summary of the SBIR project

This cloud experiment was initially funded as part of a Small Business Innovation Research (SBIR) grant. The solicitation called for enabling modern materials simulations in a larger sector of the industrial research community. High performance computing (HPC) is a technology that plays a key role in materials science, climate research, astrophysics, and many other endeavors. Numerical simulations can provide unique insight to physical phenomena that cannot be easily obtained by other means. Numerical simulations complement experimental observations, help in validating models, and advance our understanding of the world. Advances in HPC software development and algorithms are becoming increasingly important in materials science and for industries developing novel materials. According to a recent survey by the US Council on Competitiveness, faster time to market, return on investment, and enabling work that could not be performed by any other means are cited as the most common justifications for using HPC in industry. For instance, Goodyear was able to significantly reduce the time to bring new tires to market through a collaboration with Sandia National Laboratory by leveraging high performance clusters. The oil, aeronautic, and automobile industries are examples of big industries where HPC technologies have been leveraged for decades. The growing penetration of HPC into engineering fields has been fueled by the continued performance improvements of computer chips as well as the emergence of hardware accelerators such as general-purpose graphics processing units (GPUs) and the Intel Xeon Phi co-processor (also known as many integrated core architecture, or MIC).

However, one of most striking features of the US Council on Competitiveness survey, is how underrepresented are the companies that would be most likely to take advantage of soft materials simulations. The biosciences sector accounted for only 5.9% and the chemical engineering sector accounted for only 4.0% of respondents on their use of HPC resources. The Phase I SBIR proposal granted to Tech-X addresses this call and the two issues outlined above, by using an extensible object-oriented toolkit (STREAMM) for linking quantum chemistry (DFT) and classical molecular dynamics (MD) simulations and making this code suite available to take advantage of HPC cloud computing.

Process Overview

1. Kickoff team meeting of the experiment using WebEx.
2. Organization of project tasks, communication and planning through RedMine.
3. The end user, Scott Sides, obtained an AWS account and provided ssh-keys to UberCloud in order to setup a project specific security group that is used to configure the multi-node multi-container environment.
4. A specialized installer was created for LAMMPS and made available to the team.
5. The end user performed an MD scaling study on a 1-node, 4-node, and 8-node cluster.
6. The end user analyzed performance data and communicated the results to the rest of the team.

CHALLENGES

End user perspective - The cloud computing service at Amazon Web Services (AWS) provided high-quality compute nodes with efficient communication networks that enabled the good scaling seen in Figure 2. There is quite a bit of manual setup that needs to be performed by the end-user for AWS. For any cloud computing project, the first step is to create the remote compute instances. One must apply for an account at AWS and use the AWS web interface to navigate to the services for the Elastic Compute Generation 2 (EC2). The 'elastic' refers to the ability to expand or shrink the hardware usage for a particular task at a given time. Then the desired number, type and security settings for the EC2 instances must be selected. For a first-time setup, an ssh-key pair is generated and stored within the user's account information. The web interface instructs the user how to setup

their local ssh configuration so that access to any remote AWS instance can be obtained. This procedure is straightforward but again, must currently be done manually. The security group must also be specified manually and is one that is configured by UberCloud in order for the networking modules to function. Now the separate instances must be assembled and configured into a multi-node cluster.

The next steps are to copy setup applications, scripts and configuration files needed to install Docker, pull all needed Docker images, and start the computational images with all of the appropriate network configuration settings. The remote copy requires the DNS addresses generated by the AWS instance startup outlined above and must currently be performed manually. Then one of the compute instances must be designated as the 'Master' node which has two main purposes: (i) to run the 'Consul' container which is part of the framework that manages the network setup for all of the cluster instances and (ii) to provide a remote entry access point for the cluster. When launching simulations on this remote cloud cluster a user executes an SSH login command using the public IP address for the master node (again obtained manually through the AWS web tool) and a password that is automatically generated within the secure container and emailed to the user. These security measures are all part of the networking image layer in the UberCloud simulation containers. However, once these steps are in place, then running on a cloud cluster is much the same as running on an HPC cluster at a university or national lab.

BENEFITS, End user perspective

- Gained an understanding of the cloud computing philosophy and of what is involved in using a cloud-based solution for computational work.
- Cloud computing using novel [HPC software containers](#) based on [Docker](#) is an easy process compared to building and maintaining your own cluster and software environment.
- Developed an effective workflow for constructing additional HPC cloud containers.

CONCLUSIONS AND RECOMMENDATIONS

For the Phase II proposal based on this case study, Tech-X will add additional codes to the [UberCloud marketplace](#) for targeted industries and applications including those in nanotech and the pharmaceutical industries. We will also investigate ways to add functionality to our STREAMM framework to streamline the setup steps described in the 'end-user perspective' section. We will also check all our current scaling results on the Microsoft Azure cloud platform and compare with AWS and bare-metal. The Azure setup is reported to have ways of streamlining the setup process to make utilizing cloud HPC resources even easier.

Case Study Authors – Dr. Scott W Sides and Wolfgang Gentsch

Team 200

HPC Cloud Simulation of Neuromodulation in Schizophrenia



“Advania’s HPC Cloud servers with Abaqus in an UberCloud container empowered us to run numerous configurations of tDCS electrode placements to explore their complex effects on treatment efficacy.”

Figure 1: Illustration of transcranial Direct Current stimulation device.

MEET THE TEAM

End Users – Dr. G. Venkatasubramanian, G. Bhalerao, R. Agrawal, S. Kalmady (from NIMHANS); G. Umashankar, J. Jofeetha, and Karl D’Souza (from Dassault Systemes).

Software Provider – Dassault/SIMULIA (Tom Battisti, Matt Dunbar) providing Abaqus 2017 software and support.

Resource Provider – Advania Cloud in Iceland (represented by Aegir Magnusson and Jon Tor Kristinnsson), with access and support for the HPC server from HPE.

HPC Cloud Experts – Fethican Coskuner, Ender Guler, and Wolfgang Gentsch from the UberCloud, providing novel HPC software container technology for ease of Abaqus cloud access and use.

Experiment Sponsor – Hewlett Packard Enterprise, represented by Bill Mannel and Jean-Luc Assor, and Intel.

USE CASE: NEUROMODULATION IN SCHIZOPHRENIA

Schizophrenia is a serious mental illness characterized by illogical thoughts, bizarre behavior/speech, and delusions or hallucinations. This UberCloud Experiment #200 is based on computer simulations of non-invasive transcranial electro-stimulation of the human brain in schizophrenia. The experiment has been collaboratively performed by the National Institute of Mental Health & Neuro Sciences in India (NIMHANS), Dassault SIMULIA, Advania, and UberCloud, and sponsored by Hewlett Packard Enterprise and Intel. The current work demonstrates the high value of computational modeling and simulation in improving the clinical application of non-invasive transcranial electro-stimulation of the human brain in schizophrenia.

Transcranial Direct Current Stimulation (tDCS): A new neurostimulation therapy

While well-known deep brain stimulation involves **implanting** electrodes within certain areas of the brain producing electrical impulses that regulate abnormal impulses, [transcranial Direct Current Stimulation](#) (tDCS) is a new form of **non-invasive** neurostimulation that may be used to safely treat a variety of clinical conditions including depression, obsessive-compulsive disorder, migraine, and central and neuropathic chronic pain. tDCS can also relieve the symptoms of narcotic withdrawal and

reduce cravings for drugs, including nicotine and alcohol. There is some limited evidence that tDCS can be used to increase frontal lobe functioning and reduce impulsivity and distractibility in persons with attention deficit disorder. tDCS has also been shown to boost verbal and motor skills and improve learning and memory in healthy people. tDCS involves the injection of a weak (very low amperage) electrical current to the head through **surface electrodes** to generate an electric field that selectively modulates the activity of neurons in the cerebral cortex of the brain. While the precise mechanism of tDCS action is not yet known, extensive neurophysiological research has shown that direct current (DC) electricity modifies neuronal cross-membrane resting potentials and thereby influences neuronal excitability and firing rates.

Stimulation with a negative pole (cathode) placed over a selected cortical region decreases neuronal activity in the region under the electrode whereas stimulation with a positive pole (anode) increases neuronal activity in the immediate vicinity of the electrode. In this manner, tDCS may be used to increase cortical brain activity in specific brain areas that are under-stimulated or alternatively to decrease activity in areas that are overexcited. Research has shown that the effects of tDCS can last for an appreciable amount of time after exposure.

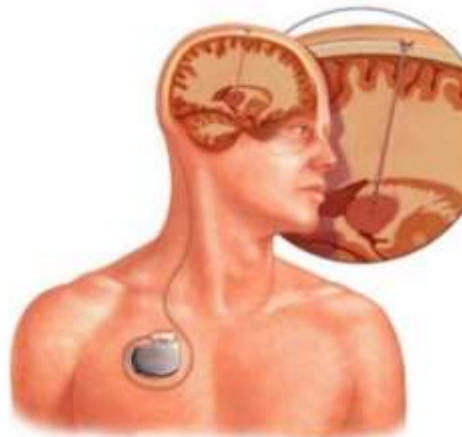
While tDCS shares some similarities with both electroconvulsive therapy (ECT) and transcranial magnetic stimulation (TMS), there are significant differences between tDCS and the other two approaches. ECT, or electroshock therapy, is performed under anaesthesia and applies electrical currents a thousand times greater than tDCS to initiate a seizure; as such, it drastically affects the functioning of the entire brain and can result in significant adverse effects, including memory loss. By contrast, tDCS is administered with the subject fully conscious and uses very small electric currents that are unable to induce a seizure, constrained to the cortical regions, and can be focused with relatively high precision. In TMS, the brain is penetrated by a powerful pulsed magnetic field that causes all the neurons in the targeted area of the brain to fire in concert. After TMS stimulation, depending on the frequency of the magnetic pulses, the targeted region of the brain is either turned off or on. TMS devices are quite expensive and bulky which makes them difficult to use outside a hospital or large clinic. TMS can also set off seizures, so must be medically monitored. By contrast, tDCS only affects neurons that are already active—it does not cause resting neurons to fire. Moreover, tDCS is inexpensive, lightweight, and can be conducted anywhere.

HPC BRAIN SIMULATION IN THE ADVANIA CLOUD

The National Institute of Mental Health and Neuro Sciences (NIMHANS) is India's premier neuroscience organization involved in clinical research and patient care in the area of neurological and psychiatric disorders. Since 2016, Dassault Systemes has been collaborating with NIMHANS on a project to demonstrate that computational modeling and simulation can improve the efficacy of Transcranial Direct Current Stimulation (tDCS), a noninvasive clinical treatment for schizophrenia. Successful completion of the first stage of this project has already raised awareness and interest in simulation-based personalized neuromodulation in the clinical community in India.

Although effective and inexpensive, conventional tDCS therapies can stimulate only shallow regions of the brain such as prefrontal cortex and temporal cortex regions. These therapies cannot really penetrate deep inside the brain. There are many other neurological disorders which need clinical interventions deep inside the brain such as thalamus, hippocampus and subthalamus regions in Parkinson's, autism, and memory Loss disorders. The general protocol in such neurological disorders is to treat patients with drugs and in some cases, patients may be recommended to undergo highly invasive surgeries. This would involve drilling small holes in the skull, through which the electrodes are inserted to the dysfunctional regions of the brain to stimulate the region locally as shown in Figure 2. This procedure is called as "Deep Brain Stimulation", in short DBS. However, DBS procedure has potential complications such as stroke, cerebrospinal fluid (CSF) fluid leakage, bleeding, etc.

Other drawbacks are that not every patient can afford DBS surgery considering their individual health conditions and high cost medical procedures.



courtesy: Mayo Clinic

Figure 2: invasive surgeries involve drilling small holes in the skull, through which the electrodes are inserted to the dysfunctional regions of the brain to stimulate the region locally.

Our project demonstrates an innovative method that can stimulate deep inside the brain non-invasively/non-surgically, using multiple electric fields applied from the scalp. This procedure can precisely activate selective regions of the brain leaving minimal risk and also making it affordable to all.

Background

The method that is adopted here is called “Temporal Interference” (TI), where we are forcing two alternating currents (transcranial Alternating Current Stimulation: tACS) at two different high-frequency electric fields towards the brain via pairs of electrodes placed on the scalp. Neither of the individual alternating fields is enough to stimulate the brain because the induced electric field frequency is much higher than the neuron-firing frequency; hence the current simply passes through tissue medium with no effect. However, when two alternating current fields intersect deep inside the brain, a pattern of interference is created which oscillates within an ‘envelope’ at a much lower frequency i.e. difference between two high-frequencies, which is commonly referred to as “beat frequency”, which would stimulate a neural activity in the brain. With this method clinicians can precisely target regions of the brain without affecting major part of the healthy brain!

It is anticipated that “Temporal-Interference” stimulation has great potential to treat a large number of neurological disorders. However, it is required to be personalized for an individual depending upon type of disease targeted and inter-individual variation in brain morphology and skull architecture. Since each patient’s brains can be vastly different, an optimal electrode placement needs to be identified on the scalp in order to create Temporal-Interference at specific regions of the brain for an effective outcome. For instance, in Parkinson's disease, thalamus and globus pallidus would most likely be the regions to create Temporal-Interference to regulate electrical signals and there by activating neurons to reduce the tremor in the patients.

The power of multi-physics technology on the Advania Cloud Platform allowed us to simulate the Deep Brain Stimulation by placing two sets of electrodes on the scalp to generate Temporal-Interference deep inside the grey matter of the brain, as presented in the Figure 3 workflow. However, a basic level of customization in post processing was required in making this methodology available to the clinician in real time and also reduce overall computational effort, where doctors can choose two pre-computed electrical fields of an electrode pair to generate temporal interference at

specific regions of the grey matter of the brain. Nevertheless, the technique proposed here can be extended to any number of electrode pairs in future.

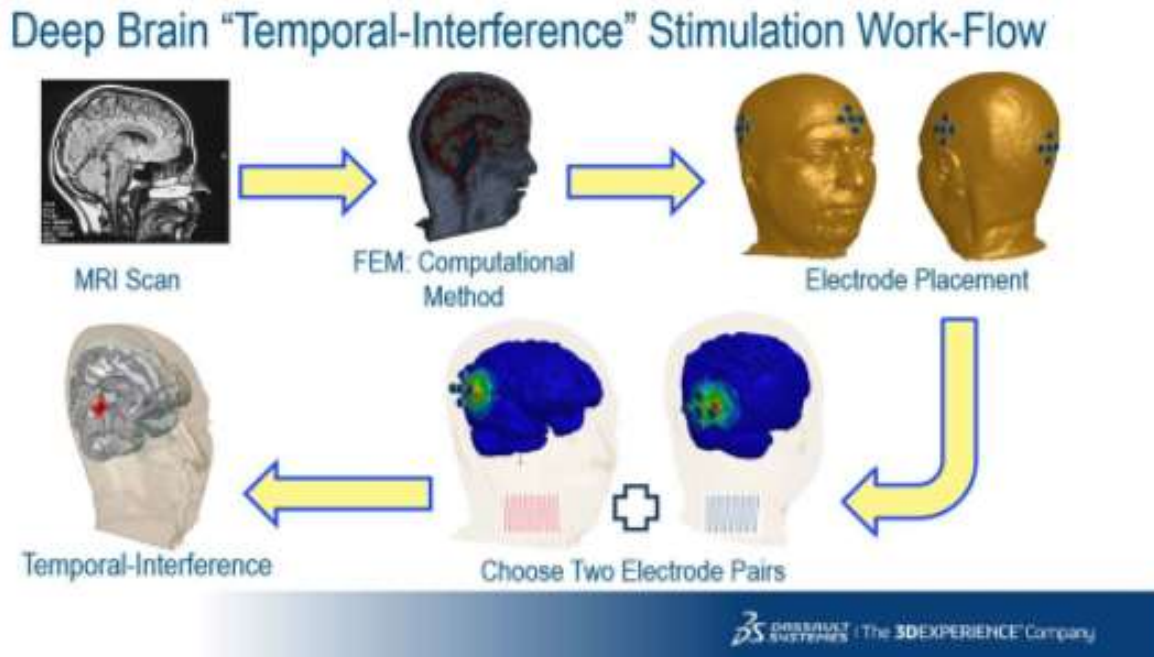


Figure 3: The workflow for the Virtual Deep Brain Stimulation on a human head model.

A high-fidelity finite element human head model was considered including skin, skull, CSF, sinus grey & white matter, which demanded high computing resources to try various electrode configurations. Access to Advania’s HPE’s Cloud and SIMULIA’s Abaqus 2017 code in an UberCloud container empowered us to run numerous configurations of electrode placements and sizes to explore new possibilities. This also allowed us to study the sensitivity of electrode placements and sizes in the newly proposed method of Temporal-Interference in Deep Brain stimulation which was not possible before on our inhouse workstations and HPC systems.

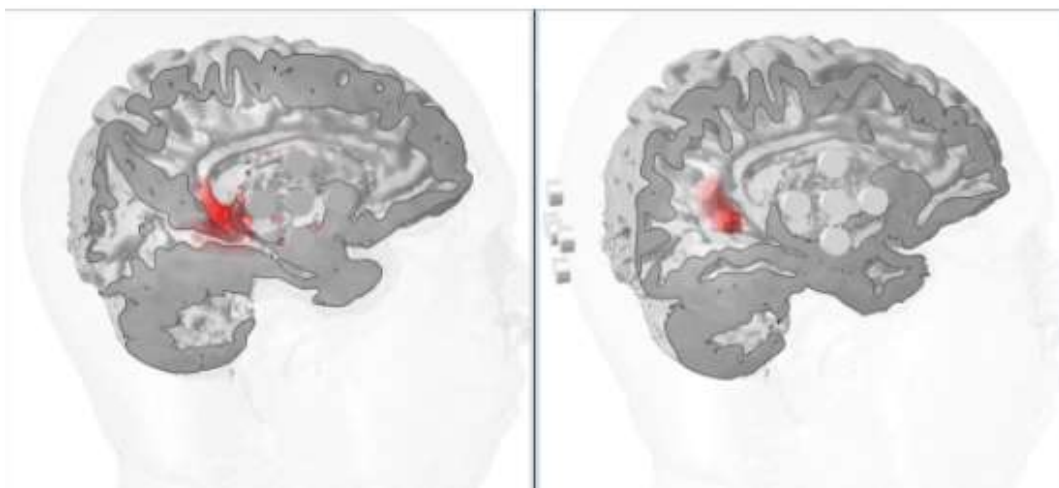


Figure 4: The results show the sensitivity of the temporal-interference region deep inside the brain based on electrode placement on the scalp.

The results demonstrated in the Figure 4 is for two sets of electrical fields superimposed to produce “Temporal Interference”:

- Configuration-1: Electrical fields generated from electrodes placed on the left and right side of pre-temporal region of the scalp.
- Configuration-2: Electrical fields generated from electrodes placed on the left of the pre-temporal and rear occipital region of the scalp.

In Configuration-1, the “temporal interference” was observed at the right hippocampus region, whereas for Configuration-2, the temporal interference” was observed at the subparietal sulcus.

Based on this insight, the team is now continuing to work towards studying various electrode placements in targeting different regions of the brain. While preliminary results look promising, the team will be working closely with NIMHANS in validating the method through further research on this topic and experimentation. In parallel, the team is also working towards streamlining the methodology such that it can easily be used by clinicians.

HPC Cloud Hardware and Results

We ran 26 different Abaqus jobs on the Advania/UberCloud HPC cluster – each representing a different montage (electrode configuration). Each job contained 1.8M finite elements. For comparison purposes, on our own cluster with 16 cores, a single run took about 75min (solver only) whereas on the UberCloud cluster a single run took about 28min (solver only) on 24 cores. Thus, we got a significant speedup of about 2x running on UberCloud.

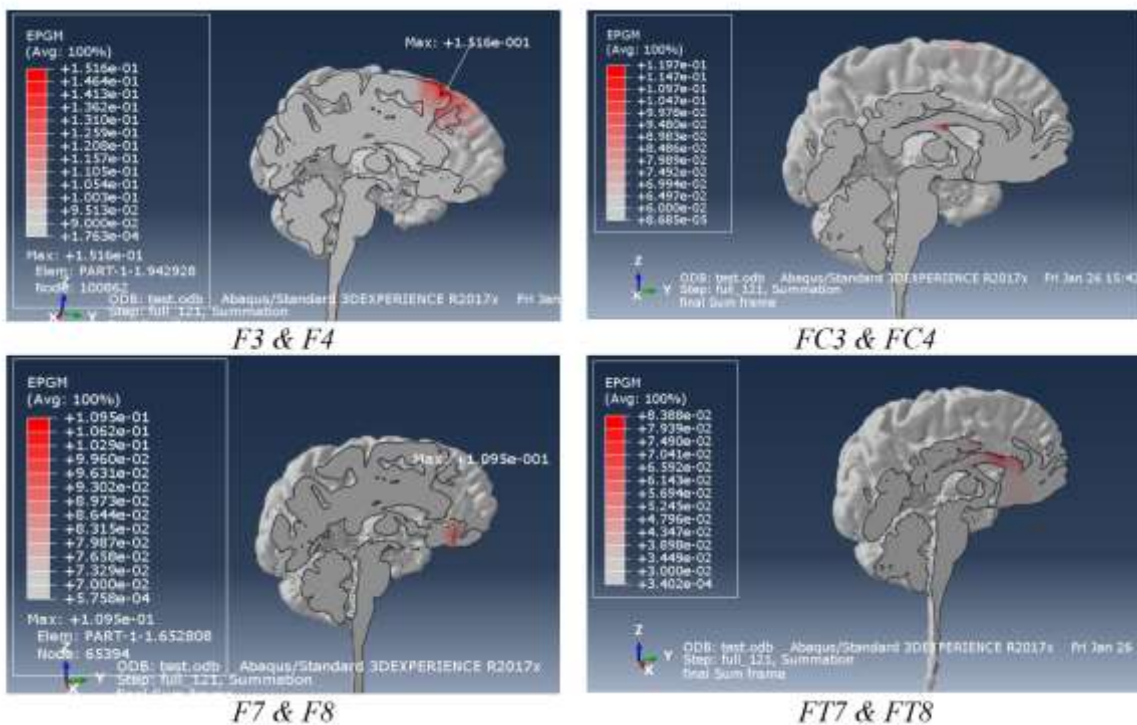


Figure 5: Localization of the peak Electrical Potential Gradient value in Abaqus for different combinations of electrodes.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don’t have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the

increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

CONCLUSION

In the recent times, the Life Sciences community has come together better than ever before, to collaborate and leverage new technologies for the betterment of health care and improved medical procedures. The application discussed here **demonstrates a novel method for "Deep Brain Stimulation" in a non-invasive way which has the potential to replace some of the painful/high risk brain surgeries such as in Parkinson's disorders.**

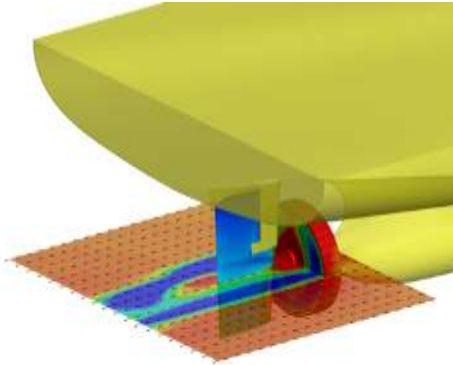
The huge benefits of these computational simulations are that they (i) predict the current distribution with high resolution; (ii) allow for patient-specific treatment and outcome evaluation; (iii) facilitate parameter sensitivity analyses and montage variations; and (iv) can be used by clinicians in an interactive real-time manner.

However, there is still a lot of work to be done in collaboration with the Doctors/Clinicians at NIMHANS and other Neurological Research Centers on how this method can be appraised and fine-tuned for real time clinical use.

Case Study Authors – G. Umashankar, Karl D'Souza, and Wolfgang Gentzsch

Team 201

Maneuverability of a KRISO Container Ship Model in the Cloud



“UberCloud containers provide easy and fast one-click browser-based access to powerful cloud resources, no need to learn anything new, which increases the engineer’s productivity dramatically.”

MEET THE TEAM

End User – Xin Gao, Master Student, Dynamics of Maritime Systems Department, Technical University of Berlin, Germany

Software & Resource Provider – Aji Purwanto, Business Development Director, NUMECA International S.A., Belgium

Technology Expert –Sven Albert, Project Engineer, NUMECA Engineering, Germany; Wolfgang Gentsch, President of UberCloud Inc., USA & Germany.

Use Case

The aim of this experiment was to verify the feasibility of overset grids for direct zigzag tests using an “appended” KRISO Container Ship (KCS) model by means of the NUMECA UberCloud container on the cloud. We used the commercial CFD software FINE™/Marine from NUMECA International S.A. for this experiment. All simulations were run on the latest NUMECA software version 6.2. To accelerate the simulations and achieve highly accurate results we used powerful HPC Cloud resources provided by UberCloud Inc. and NUMECA.



Figure 1: “Appended” KRISO Container Ship (KCS) model.

In order to validate our simulation results with experimental data, in this study, the hull geometry of MARIN (Maritime Research Institute Netherlands) was chosen, which was already published in the 2014 workshop on Verification and Validation of Ship Maneuvering Simulation Methods (SIMMAN 2014: <https://simman2014.dk>). The rudder geometry was identical with the full-scale ship, but only in model scale. A rudder box was also present in this test. However, the propeller force was modeled by an actuator disk to reduce time consumption. All parameters and coefficients used in this experiment are given in Tables 1 and 2 below.

Table 1: Geometry of hull.

Object	Full scale	Model scale
Scale	1.000	37.890
Main particulars		
L_{PP} (m)	230.0	6.0702
B_{wl} (m)	32.2	0.8498
D (m)	19.0	0.5015
T (m)	10.8	0.2850
Disp. (m ³)	52030	0.8565
S (m ²) incl. rudder	9645	6.7182
LCG (m)	111.6	2.945
GM (m)	0.60	0.016
i_{xx}/B	0.40	0.40
i_{zz}/L_{PP}	0.25	0.25

Table 2: Appendages and speed of ship.

Ruder		
Rudder Type	Semi-balanced	Semi-balanced
S of rudder (m ²)	115	0.0801
Lat. area (m ²)	54.45	0.0379
Turn rate (deg/s)	2.32	14.3
Propeller		
Type	FP	FP
No. of blades	5	5
Diameter (m)	7.9	0.208
P/D (0.7R)	0.997	0.997
A_e/A_o	0.800	0.748
Rotation	Right hand	Right hand
Hub ratio	0.180	0.186
Service speed in deep water		
U (kn, m/s)	24.0	2.005
F_n	0.26	0.26

SIMULATION PROCESS AND RESULTS

All simulations have been performed on up to three compute nodes in the cloud with each node consisting of two Intel E5-2697 V2 (2.7 GHz) processors with 24 cores each, 128 RAM, and 200 GB hard disk. The virtual experiment conducted in the NUMECA/UberCloud FINE™/Marine container is part of the author's master thesis. Firstly, a grid independence study was carried out. Afterwards, two static straight-line tests (static drift and static rudder) were performed in order to verify the feasibility of overset grids for the further direct zigzag maneuvering. The figures below indicate only a part of the results. Lack of space forbids further treatment of the uncertainty analysis here. An overview of all 3 simulated cases is shown in Table 3.

Table 3: Overview of simulation procedure.

Case 1. Calm-water resistance						
No.	Case	BS [M]	Rudder [M]	Total [M]	Time [h]	# of Cores
1.1	Medium	5.0	1.5	6.5	26	24
1.2	Coarse	2.6	0.7	3.3	19	22
1.3	Fine	9.2	2.9	12.1	27	48
Case 2. Rudder deflection						
No.	Case	BS [M]	Rudder [M]	Total [M]	Time [h]	# of Cores
2.1	M_5deg	5.0	1.5	6.5	9	22
2.2	M_10deg	5.0	1.5	6.5	8	48
Case 3. Oblique towing/Static drift						
No.	Case	BS [M]	Rudder [M]	Total [M]	Time [h]	# of Cores
3.1	M_10deg	5.1	1.5	6.6	24	22
3.2	M_20deg	5.2	1.5	6.7	27	24

Case 1: Calm-water resistance

As is known, a high-quality grid is the footstone of a precise simulation. With the latest grid generation package of FINE™/Marine, namely HEXPRESS™, the grids were generated automatically through the fully unstructured hexahedra meshes. Since the overset grids were used, the calculated region was divided into background domain and rudder domain, including ship and rudder respectively. The outlines of domain and grid in terms of three refinement levels are indicated in Figure 2.

The calculated resistance coefficients are compared with the experimental data published in the Gothenburg 2010 Workshop on Numerical Hydrodynamics, which are corresponding to case 2.2a. As can be seen from Table 4, a good agreement on the resistance is already observed with the coarse grid. However, on account of the violent oscillation of rudder force using coarse overset grids, medium grid is chosen for the following simulations. Furthermore, its accuracy is more satisfactory as well.

Table 4: Comparison of calculated and experimental coefficients.

Ref. level	$C_T^{CFD} \times 10^3 [-]$	$C_T^{EFD} \times 10^3 [-]$	$\epsilon [\%]$
Coarse	3.620	3.557	-1.77
Medium	3.559	3.557	-0.06
Fine	3.520	3.557	+1.04

Figures 2.1 - 2.3: Outline of grid strategy depending on refinement levels.

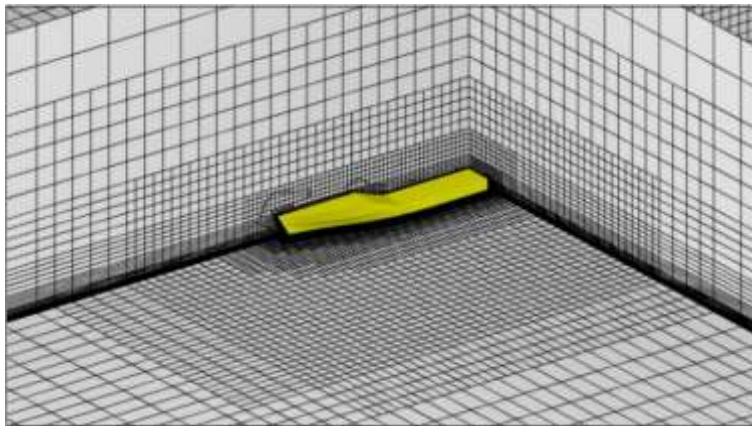


Figure 2.1: Coarse Grid.

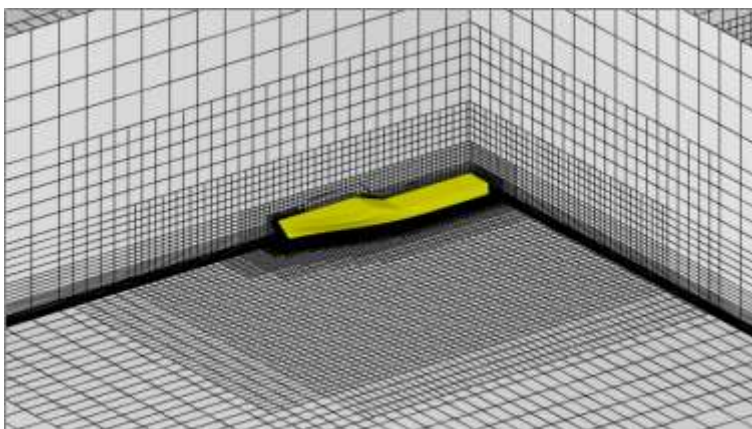


Figure 2.2: Medium Grid.

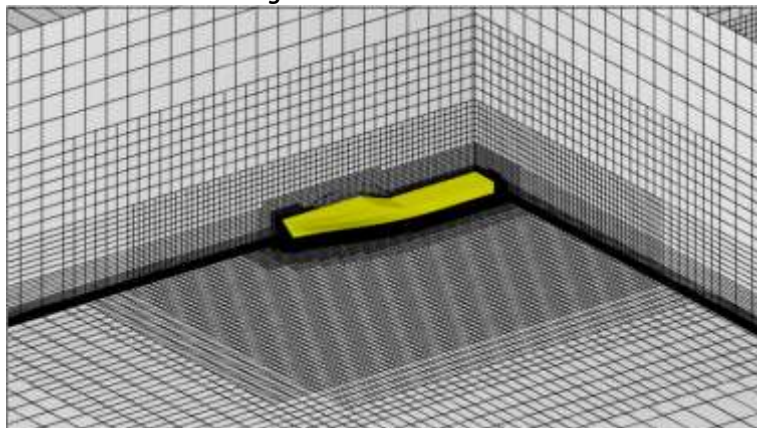


Figure 2.3: Fine Grid.

Figure 3 illustrates a comparison of global wave elevation between calculation and experiment. It is clear to see from this figure that the Kelvin wake can be resolved accurately even one ship length behind it.

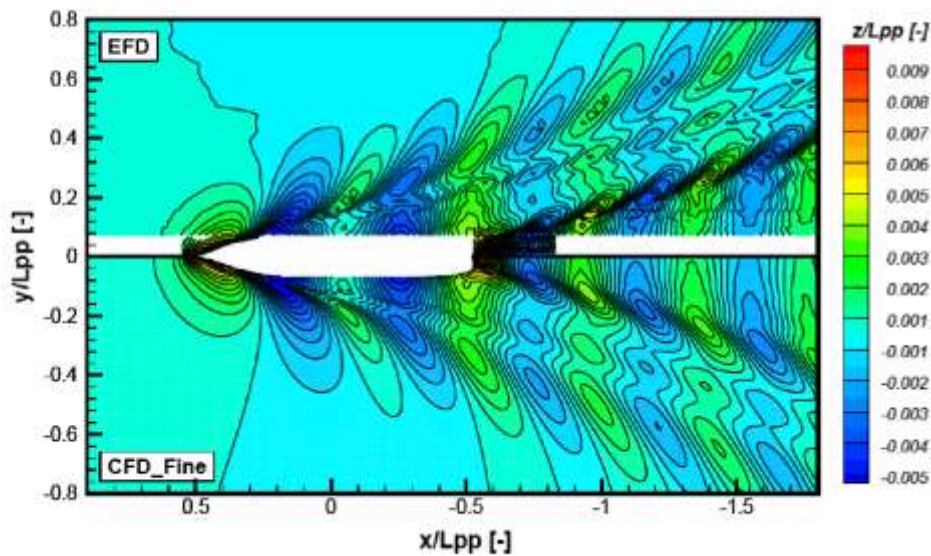


Figure 3: Comparison of wave elevation between calculation and experiment.

Case 2: Straight towing with rudder deflection

During the final zigzag maneuvering, the ship will move under the rudder deflection by means of overset grids. The flow information had to be interpolated on an overlapping interface, which locates in a very thin region for the current case. To verify its feasibility, the rudder was deflected to five and ten degrees respectively, when the ship was towed straightly. Meantime, a body force model was used to provide ship thrust under MSPP (Model Self-Propulsion Point). Figure 4 shows the comparison of dimensionless hydrodynamic coefficients for transverse force (Y') and yaw moment (N') between the current calculation and the experiment carried out in FORCE Technology for SIMMAN 2014. Although only two rudder angles were calculated, the agreement on the hydrodynamic coefficients is already quite satisfied.

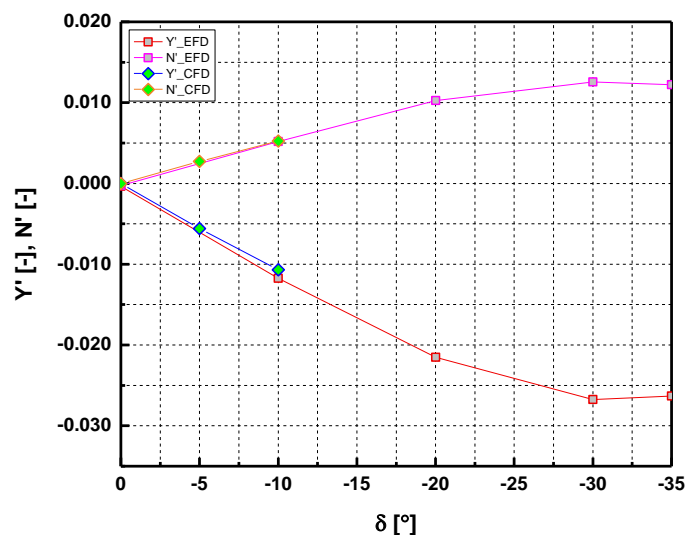


Figure 4: Comparison of normalized hydrodynamic coefficients between calculation and experiment.

The flow field at the stern region is demonstrated in Figures 5 and 6 by using streamlines. A slight change of the flow direction can be seen with the increasing rudder angle.

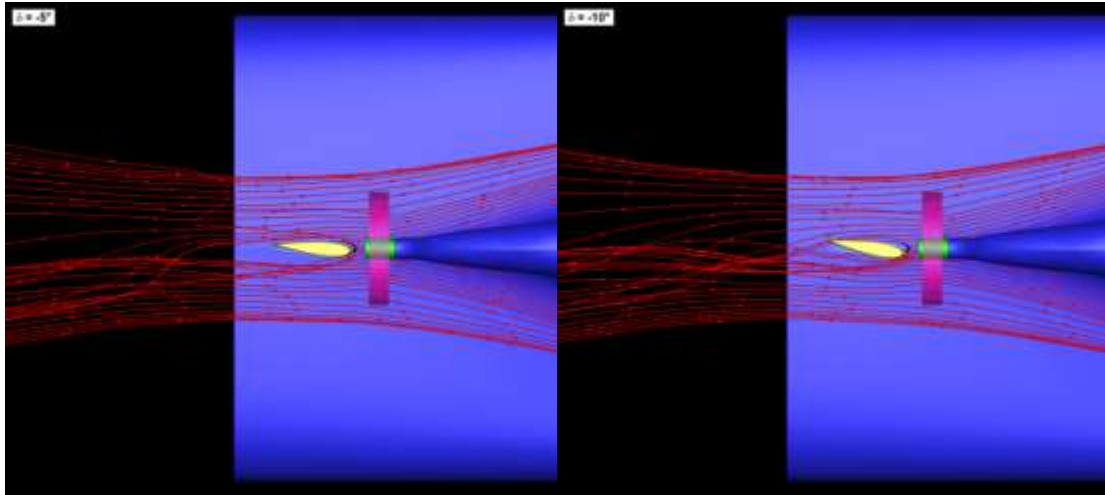


Figure 5: Streamlines at the stern region for 5° (left) and 10° (right) rudder angle to starboard.

Case 3: Oblique towing

For the verification of the large-amplitude drift motion, two oblique towing tests were executed to inspect the feasibility of overset grids. Experimental data for oblique towing test with large drift angle (greater than 10 degree) are not available. All calculated coefficients were compared with the results obtained through the system based CFD methods using the in-house developed RANS code NepIII. Currently, only the corresponding coefficient of yaw moment is presented and compared in Table 6.

The agreement of a 10-degree drift angle is excellent. However, there existed a small difference for the 20-degree condition.

Table 6: Comparison of hydrodynamic coefficients between current calculation and virtual experiment using RANS code NepIII.

Drift angle [°]	N'_{FM} [-]	N'_{NepIII} [-]	$\epsilon_{N'}$ [%]
10	0.0211	0.0211	0
20	0.0484	0.0464	-4.31

Figures 6 and 7 present the wave elevation and the wave pattern for different drift motions. In view of the dense iso-lines at the bow and stern region, these zones are zoomed in for each figure.

In Figures 8 and 9, vortex structures are illustrated using the Q-criterion. At the same time, four slices along the ship length describe the velocity profiles at different locations. The vortices go through each cutting plane.

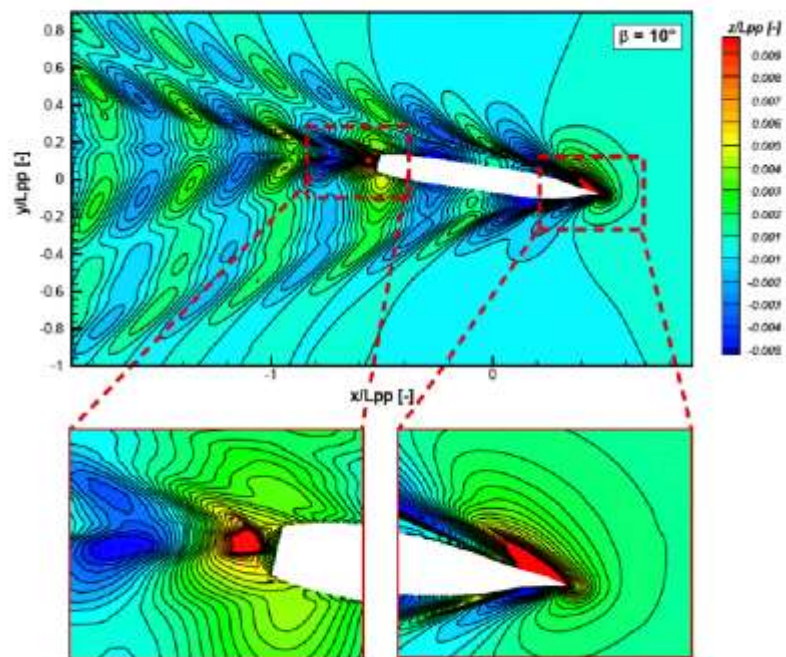


Figure 6: Wave elevation at static drift with 10° drift angle.

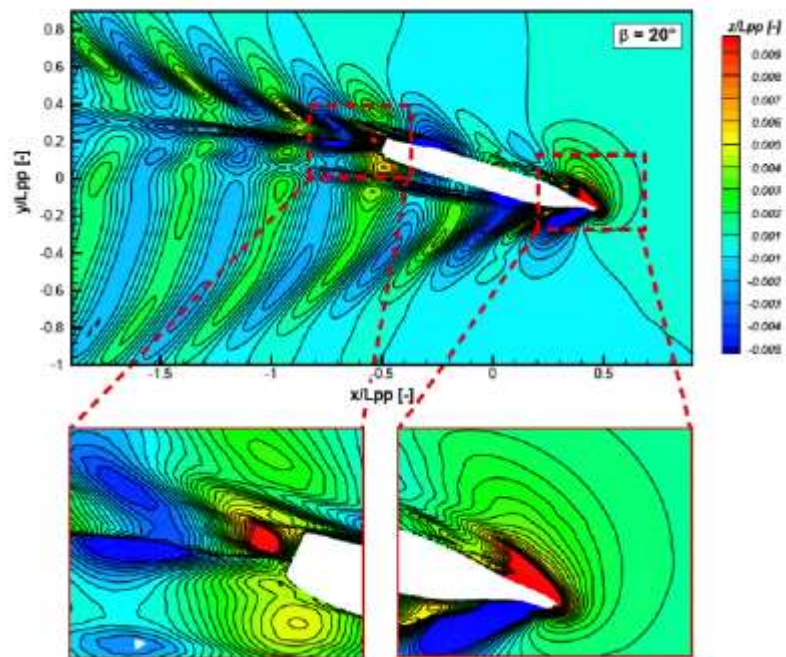


Figure 7: Wave elevation at static drift with 20° drift angle.

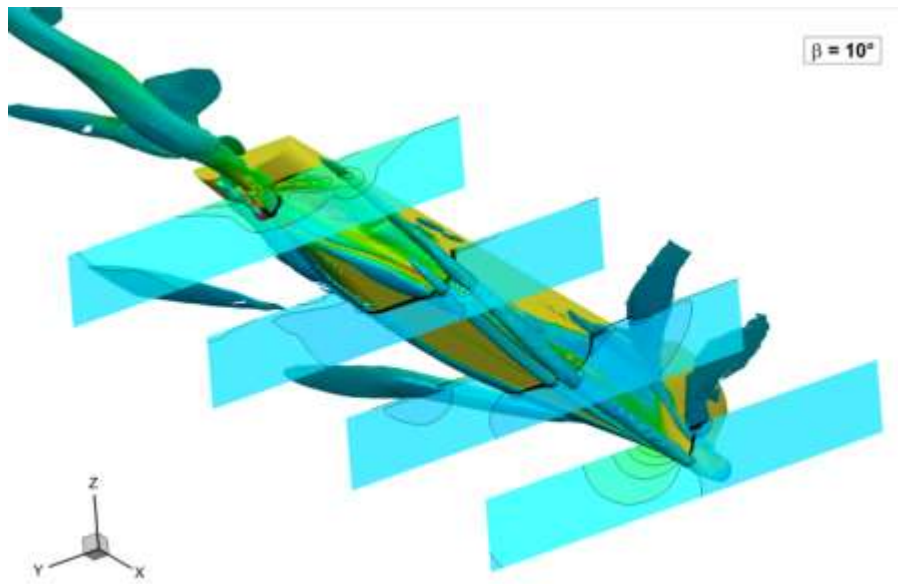


Figure 8: ISO-surfaces of $Q=1$ colored by relative axial velocity at static drift with 10° drift angle.

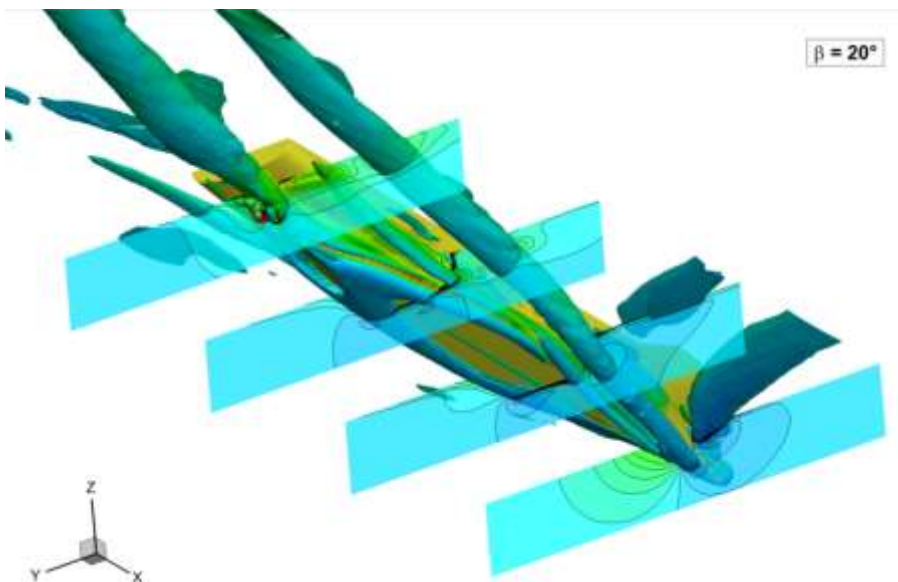


Figure 9: ISO-surfaces of $Q=1$ colored by relative axial velocity at static drift with 20° drift angle.

CHALLENGES AND BENEFITS OF THE CLOUD APPROACH

The overall process of implementing the model, setting up the cloud environment, and running the simulations operations went very smoothly. The FINE™/Marine software was pre-installed in an UberCloud container on a CentOS Linux distribution, which has a friendly Windows GUI. The FINE™/Marine container was always instantly accessible through the browser.

One of the biggest advantages of the simulation in an UberCloud container in the cloud is that hardware resources can be freely chosen depending on the different computing scales, which may be quite different from each other. No acquisition expenses of hardware are required due to the pay-per-use model. Additionally, it is suggested that more storage space per node should be considered if massive data are produced and have to be saved during the simulation.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

CONCLUSION

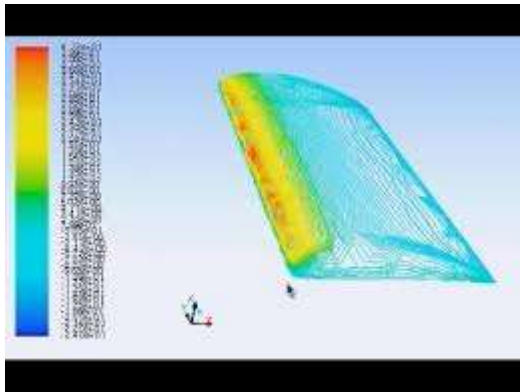
UberCloud containers on cloud infrastructure enable easy and fast simulations, accessible with one click through the browser-based GUI, thus increasing the engineer's productivity dramatically who now can fully concentrate on just the simulation experiment.

There is no need to worry about the cost of buying physical HPC hardware because this cloud model is just pay-per-use.

Case Study Author – Xin Gao, TU Berlin

Team 203

Aerodynamic Study of a 3D Wing Using ANSYS CFX



"I've been using cloud computing for several years now, tried at least four different cloud providers and found the UberCloud service by far the best. I didn't expect it would be SO easy to use."

MEET THE TEAM

End-User/CFD Expert: Praveen Bhat, Technology Consultant, India

Software Provider: ANSYS with computational fluid dynamics (CFD) code CFX

Cloud Resource Provider: Tryggvi Farestveit, Richard Allen, Anastasia Alexandersdóttir, Opin Kerfi, Iceland

HPC Expert and Service Provider: Ender Guler and Ronald Zilkovski, UberCloud.

USE CASE

The aerodynamic study of the aircraft wing provides the air flow and the forces acting on the wing due to the velocity of the air. The study is mainly used to provide in-depth insights on the air flow, pressure and velocity distribution around the wing and also parameters required to calculate the lift and the drag force.

The project involved evaluating the wing aerodynamic performance using the computational fluid dynamics (CFD) approach. A standard wing profile is considered for this experiment. The CFD models were generated in the ANSYS environment. The simulation platform was built in a 128-core HPC cloud server with 125 GB RAM and with ANSYS 19.0 modelling environment dedicated to a single user. The Cloud environment was accessed using a VNC viewer through the user's web browser. The ANSYS software was running in UberCloud's application software containers which enable users to instant interactive access and use of the ANSYS cloud environment. The following flow chart defines the container setup and modelling approach for setting up and running the simulations in the Ansys containerized environment:

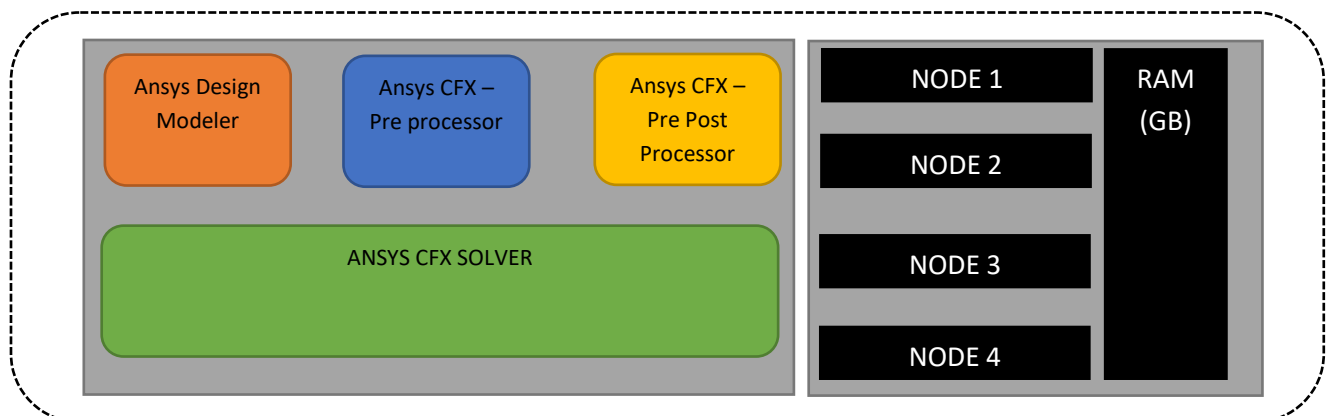


Figure 1: Container environment with Ansys CFX application.

The model construction and setup are done as shown in the following flow chart:

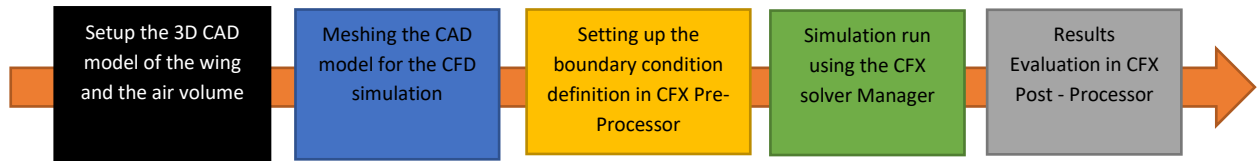


Figure 2: Different stages in Model setup and simulation run in Ansys CFX.

The following defines the step by step approach in setting up the CFD model using the ANSYS Workbench 19.0 Environment:

1. Generate the 3D wing geometry using ANSYS Design Modeler, where the input for the dimension of the wing is the co-ordinate system which is imported in the modelling environment as co-ordinate files (*.csv).
2. Develop the CFD model with atmospheric air volume surrounding the 3D wing in the ANSYS Design Modeler.
3. Import the CFD model in the CFX pre-processing environment.
4. Define the model parameters, fluid properties, and boundary conditions.
5. Define the solver setup & solution algorithm, mainly related to define solver type, convergence criteria and equations to be considered for solving the aerodynamic simulation.
6. Extract the pressure load on the wing surface which is used for calculating lift and drag forces on the wing and evaluate its stability under aerodynamic forces.

The ANSYS CFX simulation setup is solved in the HPC Cloud environment. The simulation model needs to be precisely defined with good amount of fine mesh elements around the wing geometry. The following snapshot highlights the wing geometry considered and CFX mesh model:

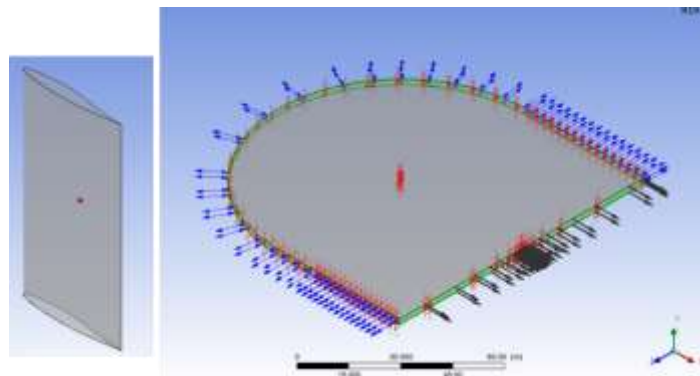


Figure3: 3D geometry of the wing.

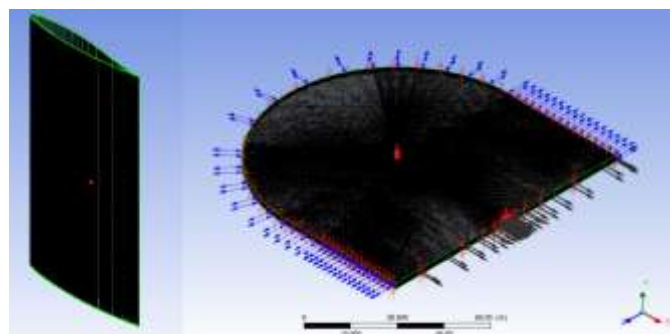


Figure 4: CFD mesh model in Ansys CFX.

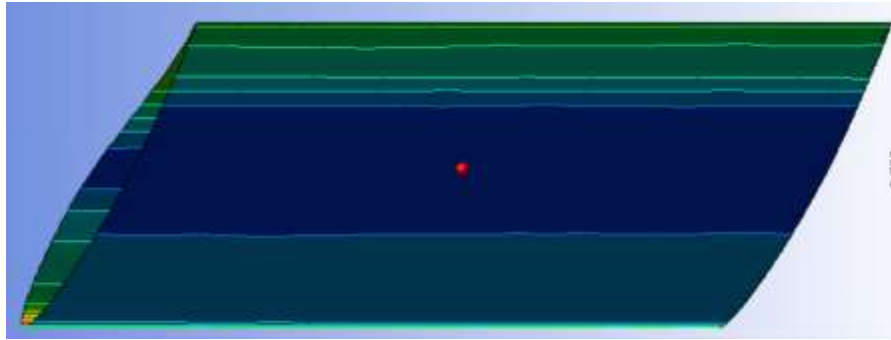


Figure 5: Pressure distribution plot at the mid-section of the wing.

Figure 5 shows the pressure distribution at the mid-section of the 3D wing. The pressure distribution across the section is uniform.

HPC Performance Benchmarking

The Aerodynamic study on the aircraft wing study is carried out in the HPC environment which is built on a 256-core server with CentOS Operating System and ANSYS Workbench 19.0 simulation package. The server performance is evaluated by submitting the simulation runs for different numbers of elements. The finer the mesh size the more is the time required to run the simulation. The run time can be minimized by using higher core systems. The following table highlights the solution time captured for a 128-core system with element numbers up to 100 million.

Table 1: Simulation performance time (sec) for different no of cores.

Model Size	No of Nodes	Cores for each node	No of cores	Solution time (sec)
10 mil model size	1	64	64	81.48
	2	64	80	58.20
	2	64	96	48.50
	2	64	112	44.09
	2	64	128	40.08

Table 2: Simulation performance time (sec) for different no of cores.

Model Size	No of Nodes	Cores for each node	No of cores	Solution time (sec)
50 mil model size	1	64	64	528.59
	2	64	80	377.56
	2	64	96	314.63
	2	64	112	286.03
	2	64	128	260.03

Table 3: Simulation performance time (Sec) for different no of cores.

Model Size	No of Nodes	Cores for each node	No of cores	Solution time (sec)
100 mil model size	1	64	64	1170.4
	2	64	80	836
	2	64	96	696.66
	2	64	112	633.33
	2	64	128	575.75

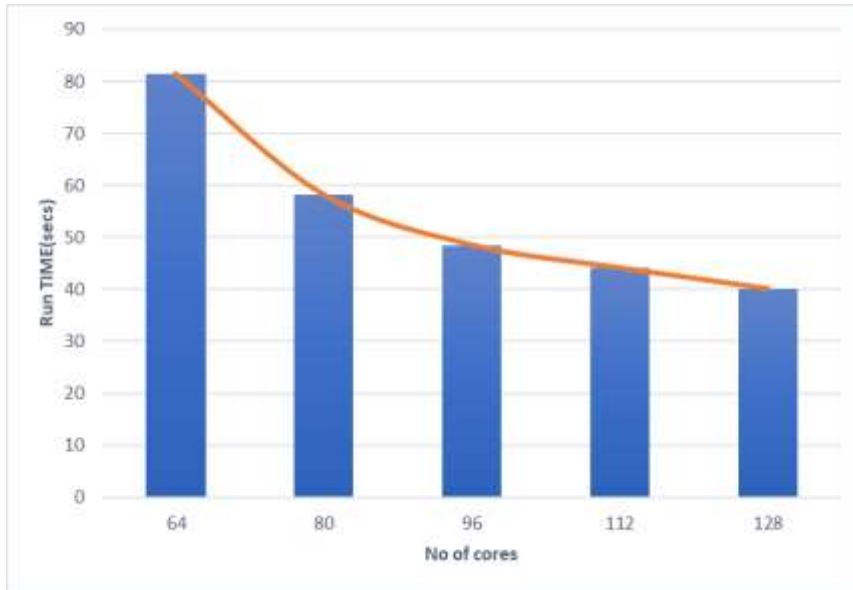


Figure 6: Runtime (secs) vs No. of Cores for 10 million mesh model.

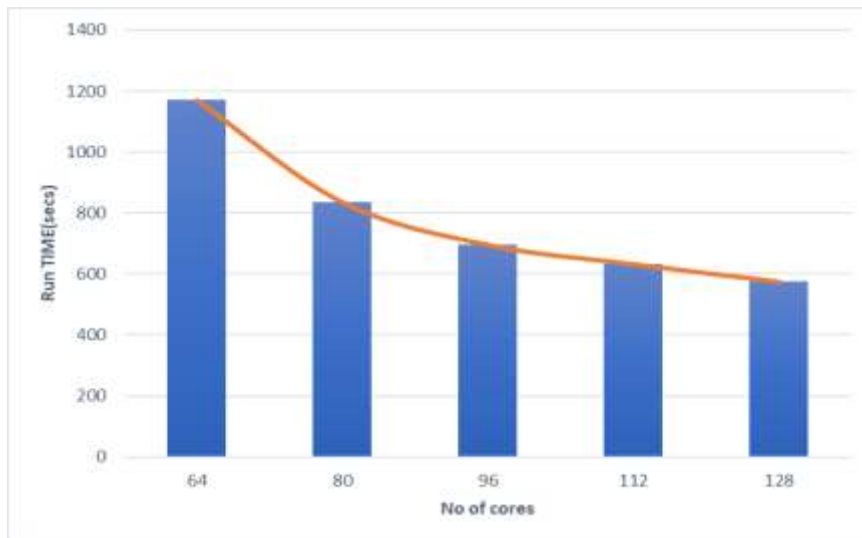


Figure 7: Runtime (secs) vs No. of Cores for 100 mil mesh model.

The simulation time reduces considerably with the increase in the number of CPU units. The solution time required for 64 cores with fine mesh model is 1.8 times higher than the time required for a 128-core server with the same mesh model. For a moderate number of elements (~ 10 mil), the 64-core server performance is 4.5 times better than a normal Quad-core system with respect to total number of simulation jobs completed in a day.

Person-hour Efforts Invested

End user/Team Expert: 120 hours for setup, technical support, reporting & overall management of the project.

UberCloud support: 30 hours for monitoring & administration of host servers and guest containers, managing container images (building & installing container images during any modifications/ bug fixes) and improvements (such as tuning memory parameters, configuring Linux libraries, usability enhancements). Most of the mentioned effort is one-time effort and will benefit the future users.

Resources: 3000 core-hours for performing various iterations in the simulation experiments (the results shown were for a scale down runtime).

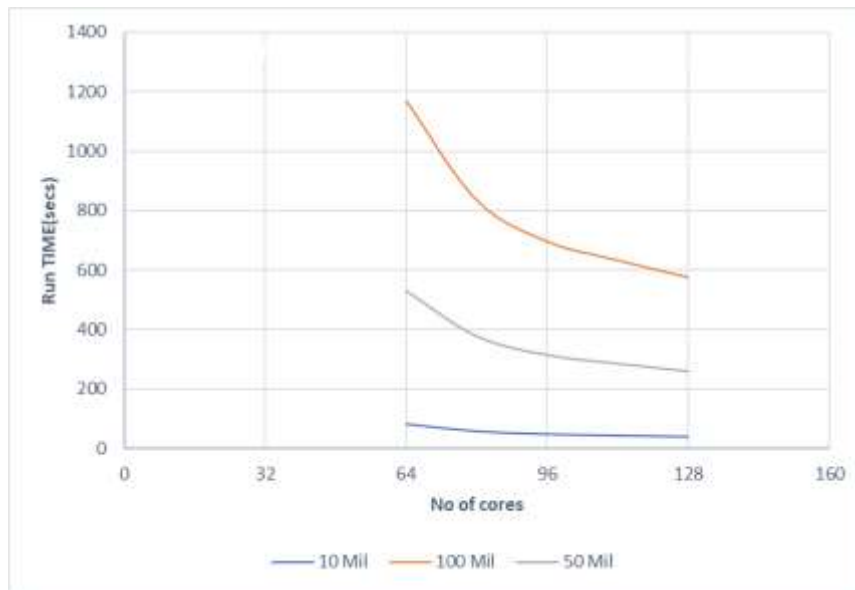


Figure 8: Comparison of Mesh models of different sizes for no. of cores vs runtime (model scalability).

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

CHALLENGES

The project started with setting up ANSYS 19.0 workbench environment with CFX modelling software in the 64-core server. Initial working of the application was evaluated and the challenges faced during the execution were highlighted. Once the server performance was enhanced from the feedback, the next level of challenge faced was scaling the existing system to a multi node container where the container would be using scaled computation environment for simulation run. The key challenge in the project was technical which involved accurate prediction of wing behaviour under the aerodynamic forces which is achieved through defining appropriate element size to the mesh model. The finer the mesh the higher is the simulation time required and hence the challenge was to perform the simulation within the stipulated timeline.

BENEFITS

1. The HPC cloud computing environment with ANSYS 19.0 Workbench made the process of model generation easier with process time reduced drastically because of the HPC resource.
2. The mesh models were generated for different cell numbers where the experiments were performed using moderate fine- to - fine to highly fine mesh models. The HPC computing resource helped in achieving smoother completion of the simulation runs without re-trials or resubmission of the same simulation runs.

3. The computation requirement for a highly fine mesh (100 million cells) is high which is near to impossible to achieve on a normal workstation. The HPC cloud provided this feasibility to solve highly fine mesh models and the simulation time drastically reduced thereby providing an advantage of getting the simulation results within acceptable run time (4 hours).
4. The use of ANSYS Workbench helped in performing different iterations in the experiments by varying the simulation models within the workbench environment. This further helped in increasing the productivity in the simulation setup effort and thereby providing a single platform to perform end-to-end simulation model development and setup.
5. The experiments performed in the HPC Cloud showed the possibility and gave extra confidence to setup and run simulations remotely in the cloud. The different simulation setup tools required were installed in the HPC environment and this enabled the user to access the tool without any prior installations.
6. With the use of VNC Controls in the Web browser, The HPC Cloud access was very easy with minimal or no installation of any pre-requisite software. The whole user experience was similar to accessing a website through the browser.
7. The UberCloud containers helped with smoother execution of the project with easy access to the server resources, and provided huge advantage to the user that enabled continuous monitoring of the job in progress in the server without any requirement to setup the server tools in the desktop.

RECOMMENDATIONS

1. The selected Opin Kerfi HPC environment is a very good fit for performing advanced computational experiments that involve high technical challenges and require highly scalable hardware resources to perform the simulation experiments.
2. There are different high-end software applications which can be used to perform Aerodynamics CFD simulation. ANSYS 19.0 Workbench environment helped us to solve this problem with minimal effort in setting up the model and performing the simulation trials.
3. The combination of Opin Kerfi and ANSYS 19.0 Workbench helped in speeding up the simulation trials and also completed the project within the stipulated time frame.

APPENDIX: About Opin Kerfi

Since 1985, Opin Kerfi has been a leading IT sales and service partner operating both in the Icelandic and international market, providing substantial financial benefits due to the green, low-cost energy grid especially to high-performance computing users. The company has consistently and successfully provided innovative and efficient services to its clients, focusing on consultation, integration, operations and subscription-based cloud- and Software-as-a-Service solutions.

Case Study Author – Praveen Bhat

Team 204

Aerodynamic Simulations using MantiumFlow and Advania Data Centers' HPCFLOW Technology



"After logging into the Advania Data Centers cloud, running a CFD case created by MantiumFlow is just a matter of starting it. This makes an engineer's life very easy."

MEET THE TEAM

End-User/CFD Expert: Andre Zimmer, Managing Director, MantiumCAE

Resource Provider: Jón Pór Kristinsson and Elizabeth Sargent, Advania Data Centers

Cloud Expert: Hilal Zitouni, Fetican Coskuner, Ender Guler, and Burak Yenier, The UberCloud.

ABOUT MANTIUMCAE

Based in Germany, MantiumCAE is an engineering consulting firm dedicated to computational fluid dynamics (CFD) simulations, with a particular focus on aerodynamics, optimization and CFD process automation. They assist manufacturing clients in establishing, enhancing, and optimizing their CFD capabilities and work to create products with greater aerodynamic performance.

As a specialized computer-aided engineering (CAE) consultant, MantiumCAE experiences both large and fluctuating computational demands to work on challenging projects. While browsing for on-demand High Performance Computing (HPC) providers on Cloud 28+, MantiumCAE discovered Advania Data Centers (ADC) and learned about their HPCFLOW service. MantiumCAE reached out to ADC's HPC experts and consulted with them, and subsequently determined that the best approach was to execute a hybrid approach to cloud-based HPC. This allowed them to combine their existing in-house HPC infrastructure with on-demand HPC resources from ADC. The result is a flexible approach which allowed MantiumCAE to make the most out of its existing HPC investments while increasing its ability to scale up HPC resources quickly and efficiently for its customers.

ABOUT ADVANIA DATA CENTERS

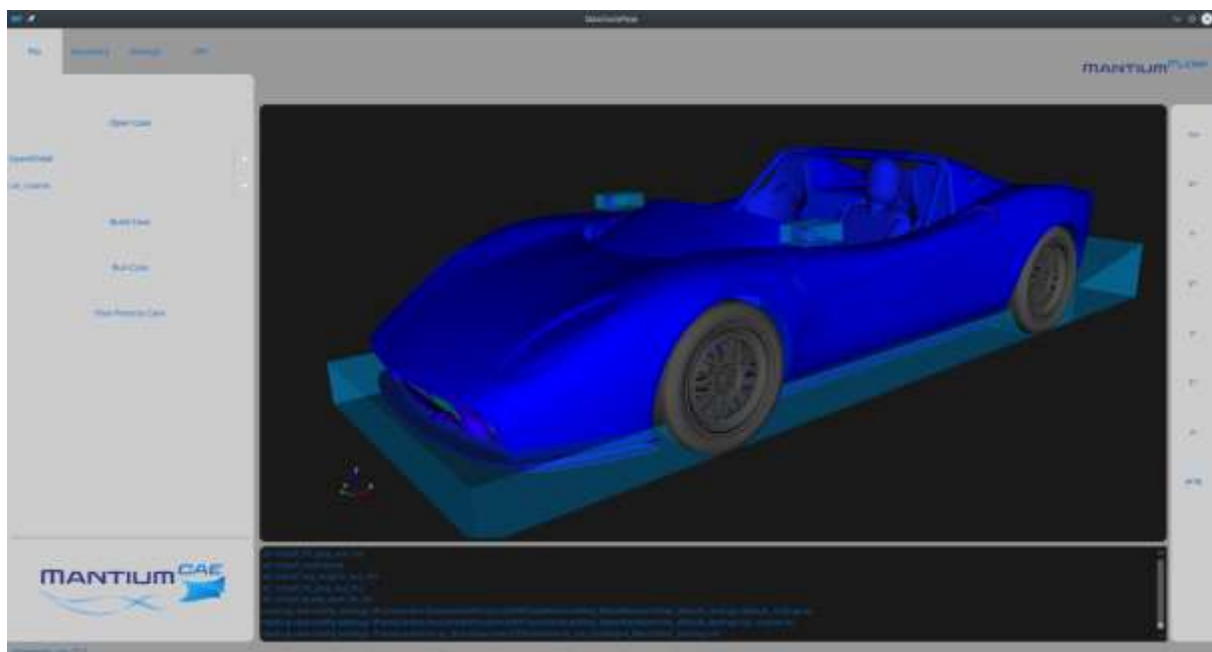
Advania Data Centers is a high-density computing technology company headquartered in Reykjavik, Iceland with operations in Sweden, Norway, Germany and the United Kingdom. Through extreme growth, Advania Data Centers now operate one of Europe's largest datacenter campuses in Iceland that is tailor made for high density hosting such as HPC, blockchain technology and high-density compute, all powered by renewable energy. Advania's HPC team consists of experts that oversee the operation of HPC environments and HPC Jobs of their customers, globally leading organizations in manufacturing, technology, science among other industries. Advania partners with industry leaders in HPC such as Hewlett Packard Enterprise, Intel, Nvidia, and UberCloud to deliver next generation

HPC environments such as HPCFLOW – Advania’s Bare Metal HPC Cloud – where HPC operators can execute simulations in a fast and efficient manner.

USE CASE

This case study shows how ADC’s HPCFLOW computing resources allowed MantiumCAE to create a CFD simulation quickly and efficiently for the Silvermine 11SR sports car. To achieve this, MantiumCAE set up a CAE computing environment in the Advania’s HPCFLOW cloud where simulations could be carried out quickly and efficiently.

A typical external vehicle aerodynamics simulation needs between 2.000 and 10.000 CPU core hours to be processed. Processing this simulation would take weeks to run on a 16-core workstation, but by using the HPCFLOW cloud environment together with MantiumFlow, MantiumCAE is able to deliver results within one business day.

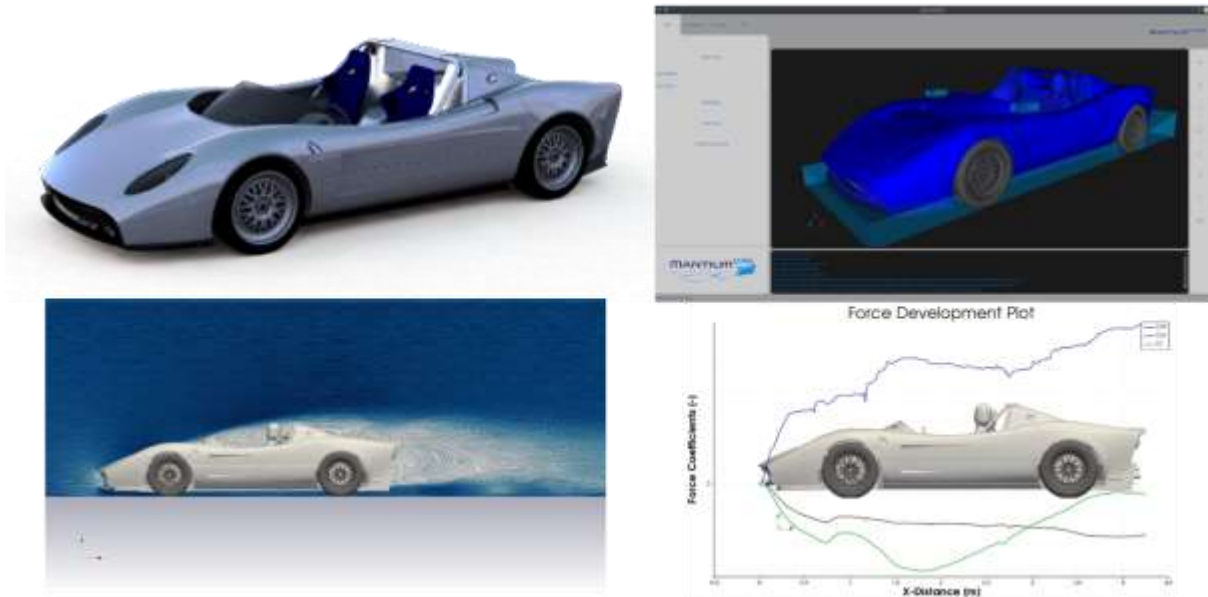


METHOD

In order to successfully create and carry out the CFD simulations for the Silvermine 11SR, MantiumCAE needed the following:

- CFD Engineer with a workstation
- MantiumFlow for the CFD setup
- HPC computing power from ADC
- MantiumFlow for post-processing

The process of running CFD simulations using HPCFLOW is straight forward. First, the engineer creates the CFD case using MantiumFlow, which automates the setup process and uploads it to ADC’s HPCFLOW. The engineer then runs the CFD simulations with a script created by MantiumFlow on the ADC environment.



Afterwards a report containing a series of plots and images is automatically created by MantiumFlow. The almost fully automated approach minimizes user error and ensures that simulations can be repeated. Everything is executed using a desktop-like environment which is easy to use and navigate.

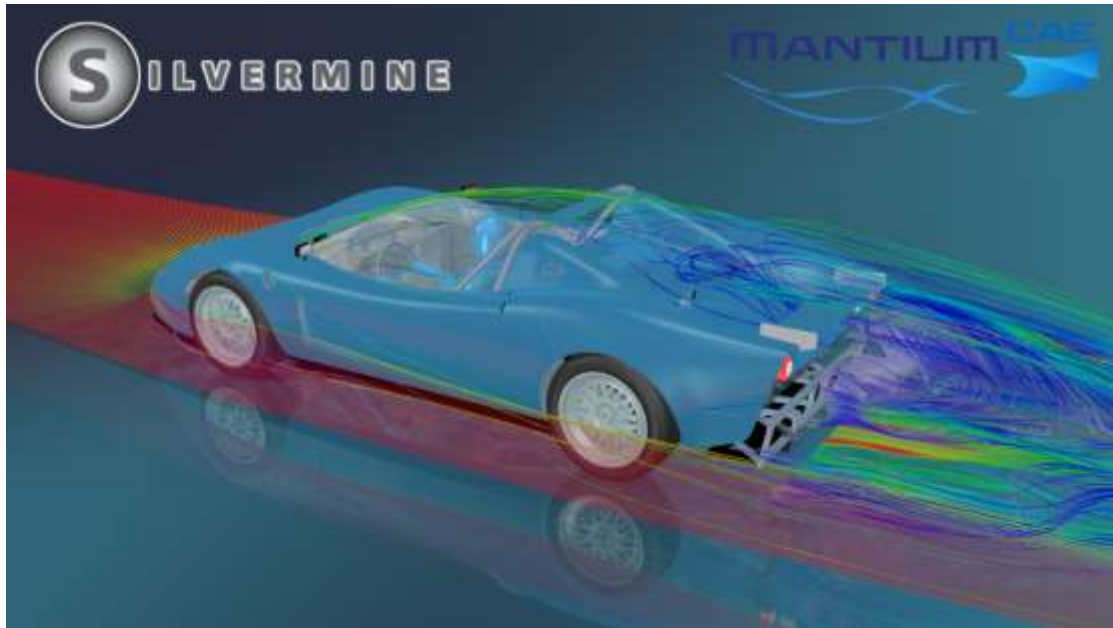
UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

BUSINESS BENEFITS AND NEXT STEPS

By successfully using ADC's HPCFLOW technology, MantiumCAE was able to execute HPC CAE projects on a scale that was previously unattainable, and with a flexibility that allowed them to serve their clients' needs better and faster. This was done without any upfront investment in computers or facilities. MantiumCAE benefitted greatly from the flexibility of the HPCFLOW service, which allowed it to scale its use of HPC resources up and down to meet its changing demands and pay only for what was needed. ADC's HPC nodes proved to be well-suited to CFD, with 8GB of RAM per Intel Xeon E5-2683 v4 core with a total of 256GB of RAM and 32 cores, and we were able to process workloads quickly and efficiently.

By giving MantiumCAE access to a dedicated HPC engineer for technical support throughout the project process, ADC ensured that there was always someone available to answer questions or troubleshoot problems. They listened to MantiumCAE's needs and provided an excellent level of service and support. This, combined with ADC's low cost per hour, made the experience very positive.



As a result of its work with Advania Data Centers, MantiumCAE has greatly strengthened its ability to be more competitive for challenging projects, without high initial investments and high cost of on-demand resources. This has secured their existing business, opened new markets and positioned them well for future growth.

Case Study Authors – Andre Zimmer and Elizabeth Sargent

Team 205

Vehicle Crash Using ANSYS LS-DYNA in the Opín Kerfi Cloud



“Combination of Opín Kerfi Cloud with UberCloud ANSYS LS-DYNA containers provide a powerful platform for accurate simulations that involved impact physics.”

MEET THE TEAM

End-User/CFD Expert: Praveen Bhat, Technology Consultant, India

Software Provider: ANSYS 19.0 with finite element code LS-DYNA

Cloud Provider: Tryggvi Farestveit, Richard Allen, Anastasia Alexandersdóttir, Opín Kerfi, Iceland

HPC Expert and Service Provider: Ender Guler and Ronald Zilkovski, UberCloud.

USE CASE

There are many severe and fatal crashes that result from vehicles colliding each other or to any stationary object. These cause extremely high impact forces and deformation on the frontal area of the car. The objective of the study is to demonstrate the frontal crash simulation of vehicle against a rigid wall to examine injury risk and potential of safety. In particular, various FE models are used to perform contact–impact nonlinear dynamic analysis of rigid wall with vehicle. In this paper ANSYS LS-DYNA Explicit solver is used to numerically simulate the crash of the vehicle with a rigid wall. The main objective of this project is to understand Vehicle crash behavior under dynamic conditions. The simulation framework is developed and executed in the Opín Kerfi Cloud with UberCloud ANSYS containers to achieve good accuracy in result prediction and also with respect to the solution time and resource utilization.

Process Overview

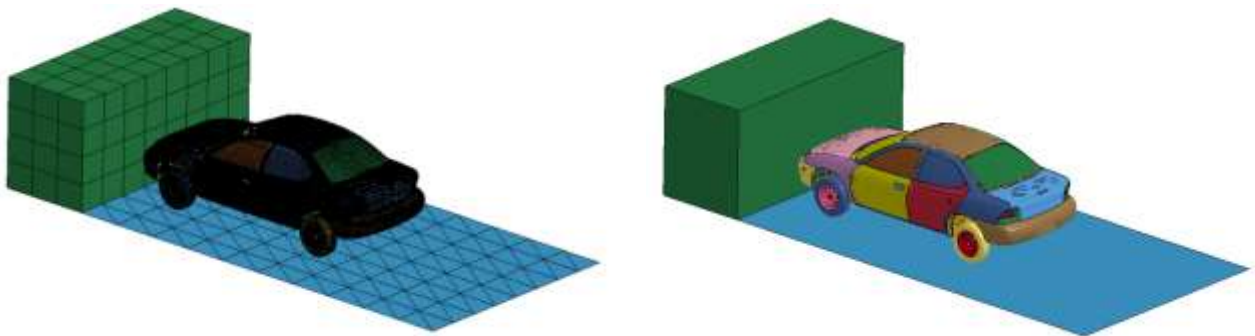


Figure 4: Geometry & Mesh model for a car crash analysis.

1. The car model is meshed using the 2D quad mesh elements. The contacts and interactions between different components in the car assembly is defined.
2. The material properties for different parts of car is defined. The section properties are defined which involved thickness definition for different components in the assembly.
3. The next step in the model setup is defining the model boundary conditions and assigning load curves. The rigid wall is considered for the car assembly impact and the speed for impact for the car is defined as a load curve.
4. The solution algorithm and convergence criteria are defined along with the output parameters and results to be written for post processing.
5. The model is solved in ANSYS LS-DYNA in parallel and once the solution is converged, the final result is used to visualize the output of the simulation result, and the respective result components are captured using the post-processing software tool in ANSYS.

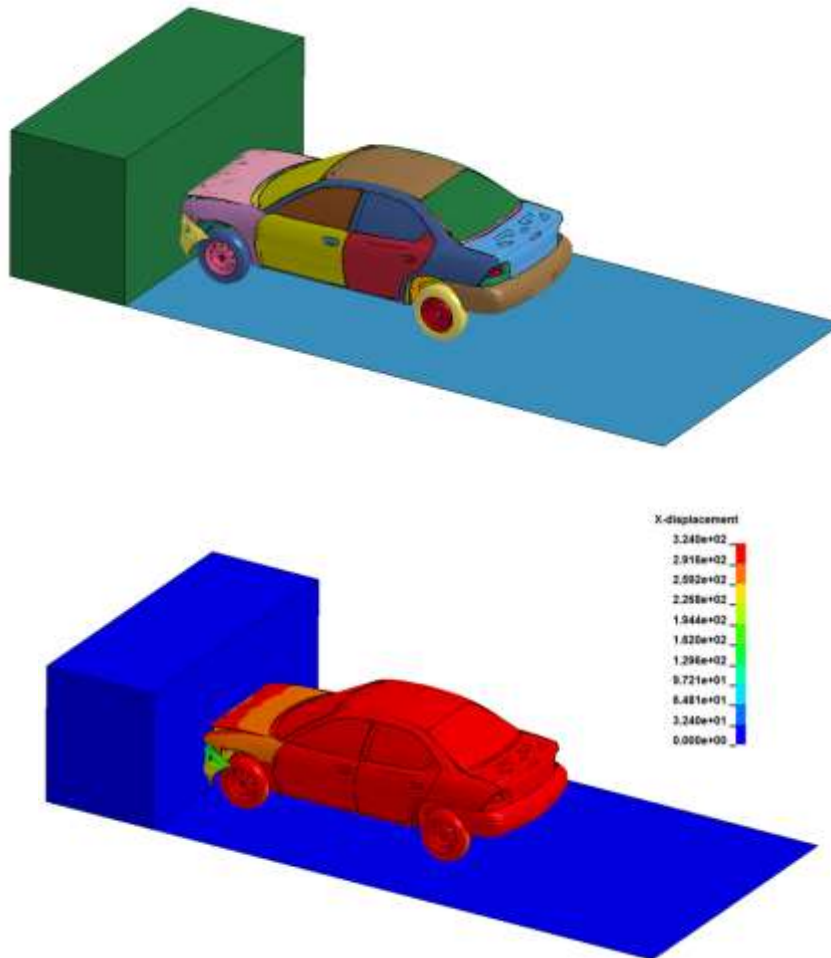


Figure 5: Deformation plot of car assembly and the rate of progressive damage during impact to rigid wall.

The car impacts to the rigid wall, and the damage is progressive and depends on the rate and velocity at which the vehicle impacted the wall. Damage due to impact on the car components is shown in Figure 3 with the rate of damage which can be compared visually:

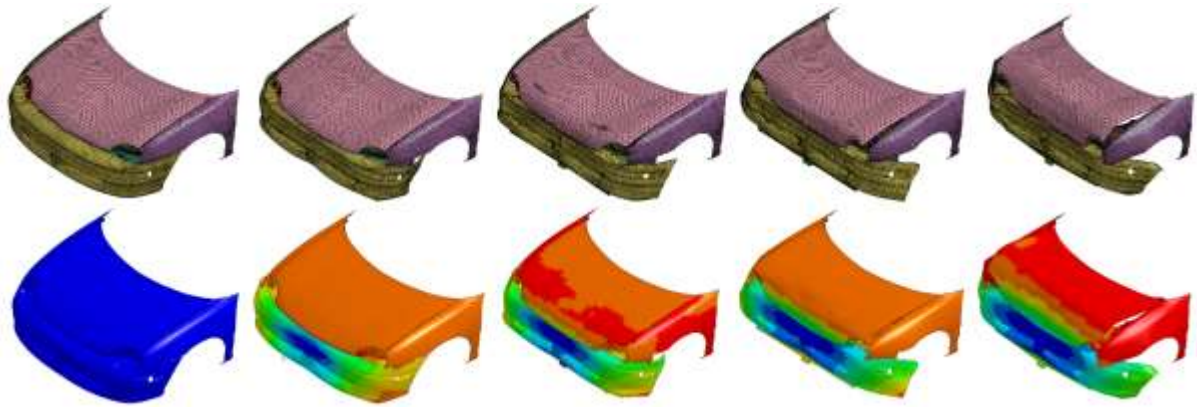


Figure 6: Rate of damage due to frontal impact of the car to the wall.

HPC Cloud Simulation

The HPC system is a 256-core system with 130 GB RAM having centos Operating system. The car assembly is simulated using ANSYS LS-DYNA in the UberCloud HPC container, which is integrated with the Open Kerfi cloud platform. The model is evaluated for the impact behavior of the vehicle and also determine the rate of damage and the stresses developed on the car assembly.

Different finite element models are developed by developing both fine and coarse meshes. The models are submitted to ANSYS LS-DYNA. The time required for solving the model with different mesh intensity is then captured to benchmark the HPC performance in solving high density mesh models. The boundary conditions, solution algorithm, solver setup and convergence criteria remain the same for all the models developed.

Figure 4 & 5 provides the comparison plot of the solution time required for different mesh density model with and without parallel processing. The comparison of the solution time with single core processor and 128-core processor shows that the solution time required is significantly less when compared with running the same simulations with single core.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

Figure 6 shows the comparison plot for the solution time required for a model with 535K elements which are submitted with different CPU Cores. Figure 7 provides the comparison on the solution for different Fine mesh models submitted using different CPU Cores. Parallel computing provides the

advantage of solving highly fine mesh model for a complex simulation like car frontal crash with a very lower solution time.

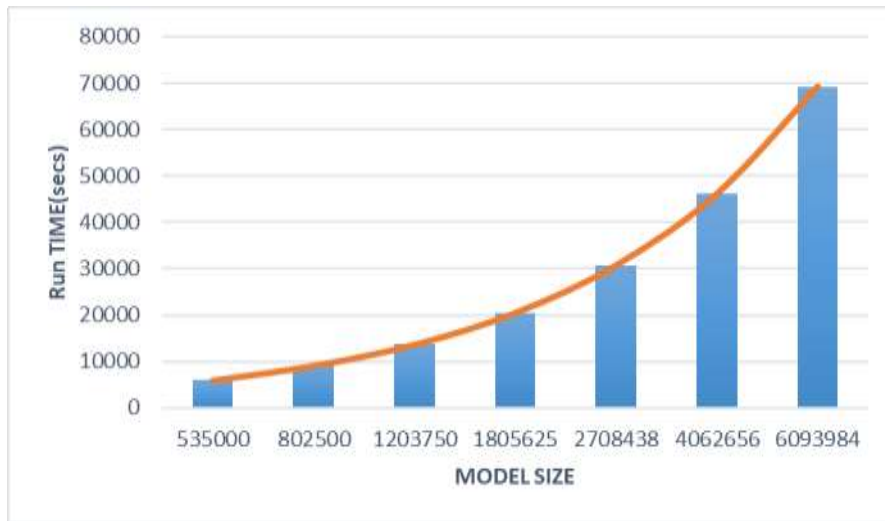


Figure 7: Solution time required for different mesh density with single CPU Core.

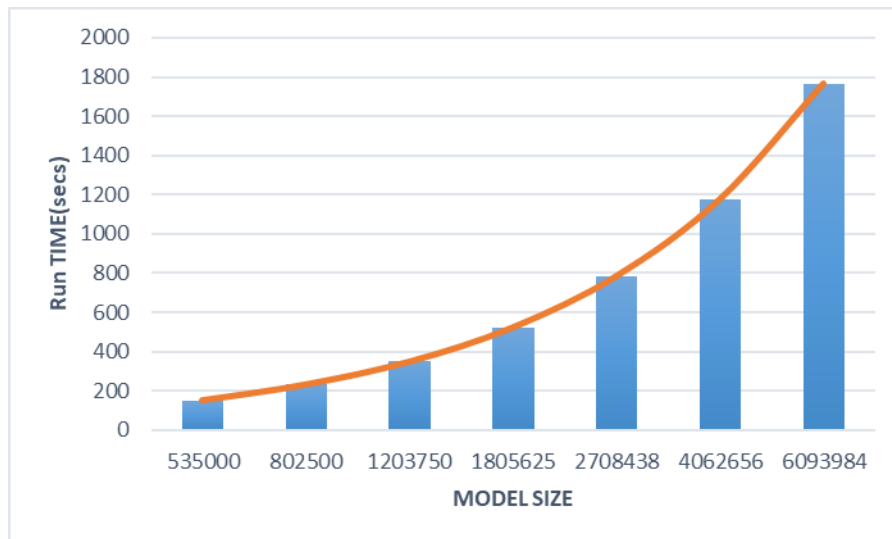


Figure 8: Solution time required for different mesh density using 128 CPU Core.

Effort Invested

End user/Team Expert: 70 hours for simulation setup, technical support, reporting and overall management of the project.

UberCloud support: 1 hours for monitoring & administration of the performance in the host server.

Resources: ~3000 core hours were used for performing various iterations in the simulation experiments.

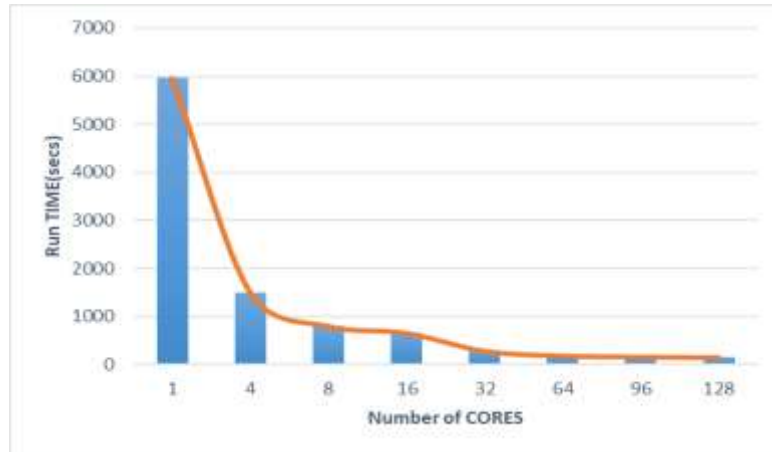


Figure 9: Solution time for a model with 535K elements solved using different HPC Core configuration.

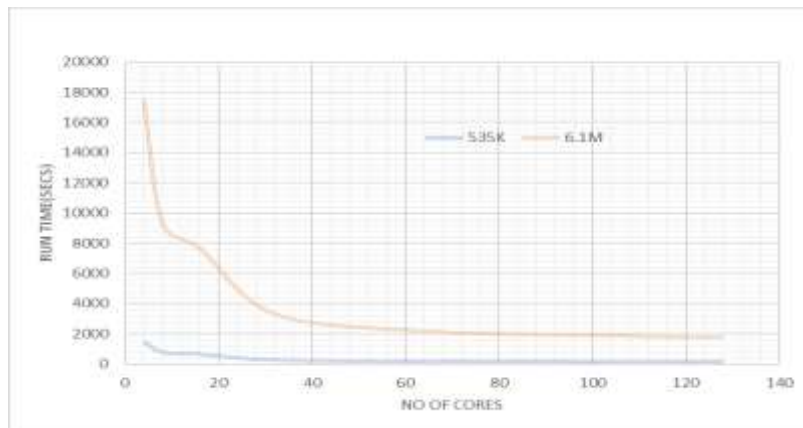


Figure 10: Comparison in solution time for different mesh densities models solved using different HPC core configuration.

CHALLENGES

The project challenges were related to technical complexity and ability to run the dynamic simulation with very short period of execution time. Hence it was necessary to perform trials with different mesh density model to accurately capture the air bag behavior. The finer the mesh the better is the simulation result accuracy, but the higher obviously is the simulation runtime required and hence it was necessary to perform the simulation within the stipulated timeline. Getting exposure to the [UberCloud LS-DYNA container](#) and OpIn Kerfi cloud platform consumed some time as this required learning and understanding how simulation work can be performed in the browser environment.

BENEFITS

1. The HPC cloud computing environment with OpIn Kerfi, UberCloud, ANSYS Workbench & LS-DYNA made the process of model generation easier with process time reduced drastically along with result viewing & post-processing.
2. The mesh models were generated for different cell numbers with using coarse – to – fine to highly fine mesh models. The HPC computing resource helped in achieving smoother

completion of the simulation runs without re-trials or resubmission of the same simulation runs thereby helping the user to achieve highly accurate simulation results.

3. The computation time requirement for a fairly fine mesh (~535K cells) is high, which is near to impossible to achieve on a normal workstation. The HPC cloud provided this feasibility to solve highly fine mesh models and the simulation time drastically reduced thereby providing an advantage of getting the simulation results within acceptable run time (~30 min).
4. The experiments performed in this HPC Cloud environment showed the possibility and gave extra confidence to setup and run the simulations remotely in the cloud. The different simulation setup tools were installed in the HPC environment and this enabled the user to access the tool without any prior installations.
5. With the use of VNC Controls in the Web browser, The HPC Cloud access was very easy with no installation of any pre-requisite software. The whole user experience was similar to accessing a website through the browser.
6. The UberCloud HPC containers helped with smooth execution of the project and with easy access to the server resources. The UberCloud environment integrated in the Opin-Kerfi platform proved to be powerful as it facilitates running parallel UberCloud containers, and the secured data connections for transfer of the simulation data from and to the local system proved to be robust and fast.

CONCLUSION & RECOMMENDATIONS

1. The HPC Cloud environment with UberCloud containerized ANSYS Workbench with LS-DYNA in the Opin Kerfi platform was a good fit for performing complex simulation that involved huge hardware resource utilization with high number of simulation experiments.
2. Opin Kerfi with UberCloud HPC containers was an excellent fit for these advanced computational experiments that involve high technical challenges with complex geometries and cannot be solved in a normal workstation.
3. ANSYS Workbench with LS-DYNA in this HPC environment helped us to solve this problem with minimal effort in setting up the model and performing the simulation trials.
4. The combination of Opin-Kerfi, HPC Cloud, UberCloud Containers, and ANSYS Workbench with LS-DYNA helped in speeding up the simulation trials and also completed the project within the stipulated time frame.

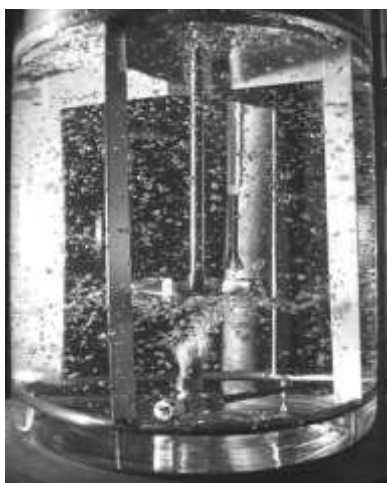
APPENDIX: About Opin Kerfi

Since 1985, Opin Kerfi has been a leading IT sales and service partner operating both in the Icelandic and international market, providing substantial financial benefits due to the green, low-cost energy grid especially to high-performance computing users. The company has consistently and successfully provided innovative and efficient services to its clients, focusing on consultation, integration, operations and subscription-based cloud- and Software-as-a-Service solutions.

Case Study Author – Praveen Bhat

Team 206

Establishing the Design Space of a Sparged Bioreactor on Microsoft Azure



1)

“The combination of Microsoft Azure with UberCloud ANSYS FLUENT Container provided a strong platform to develop an accurate virtual simulation model that involved complex multi-phase flow and tank geometries.”

MEET THE TEAM

End-User/CFD Expert: Sravan Kumar Nallamothu, Sr. Application Engineer, and Marc Horner, PhD, Technical Lead, Healthcare, ANSYS, Inc.

Software Provider: ANSYS, Inc. and UberCloud Fluent Container

Resource Provider: Microsoft Azure

HPC Expert: Shitalkumar Joshi, ANSYS, and Wolfgang Gentsch, UberCloud.

USE CASE

The scale-up of pharmaceutical laboratory mixers to a production tank is not a trivial task as it requires a thorough understanding of complex turbulent and multiphase processes impacting oxygen mass transfer. The interplay between the geometric design of the tank and tank operating parameters are critical to achieving good mixing, esp. at (larger) production scales. In an effort to improve process understanding, international regulators suggest a Quality by Design (QbD) approach to process development and process control. In the Quality by Design (QbD) framework, significant emphasis is placed on the robust characterization of the manufacturing processes by identifying the engineering design space that ensures product quality. There are various geometry and operating parameters influencing oxygen mass transfer scale-up from lab scale to production scale. Understanding the effect of these parameters can lead to robust design and optimization of bioreactor processes.

The main objective of this study is to understand the impact of agitation speed and gas flow rate on the gas holdup and mass transfer coefficient, which are two critical parameters that help process engineers understand mass transfer performance. The general-purpose CFD tool ANSYS Fluent is used for the simulations and the simulation framework is developed and executed on Azure Cloud

¹ Picture from Marko Laakkonen (reference see next page)

resources running the ANSYS Fluent UberCloud container. This solution provided a scalable platform for achieving sufficient accuracy while optimizing the solution time and resource utilization.

PROCESS OVERVIEW

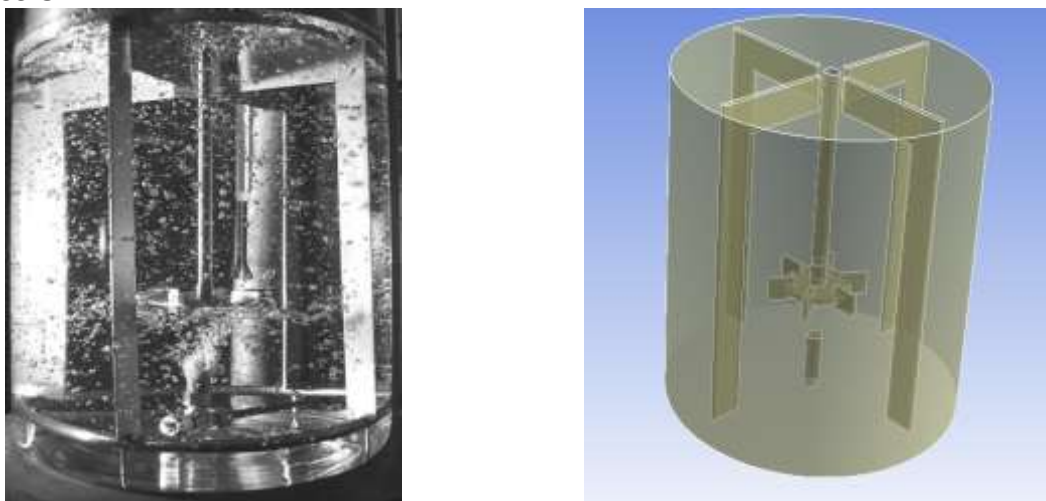


Figure 11: 194L Tank used for experiments¹ and representative CFD Model.

The stirred tank is agitated by a 6-bladed Rushton turbine blade for dispersing the air bubbles generated by the sparger. Four custom baffles are included to prevent vortex formation. Experimental conditions and results are taken from the extensive study performed by Laakkonen².

A full 3D model of the 194 L tank is considered for this CFD study, which is meshed using polyhedral elements. The Eulerian multiphase model is used for simulating the two phases: water and air. The population balance model with quadrature method of moments (QMOM) is used to simulate bubble coalescence and breakup processes. The Ishii-Zuber drag model is used to account for momentum exchange between water and air bubbles. For bubble coalescence, a model based on the Coulaloglou-Tavlarides model is used and the breakup model is based on the work of Laakkonen. It was observed that non-drag forces did not significantly impact gas holdup and mass transfer. A zero-shear boundary condition was applied for the water phase at the upper free surface, and a degassing boundary condition is used to remove the air bubbles.

The steady-state solver is used for running the simulations. Each simulation is solved until gas holdup and mass transfer coefficient reach steady values. The mass transfer coefficient is calculated using a custom field function, formulated based on a correlation derived from penetration theory³. A volume-averaged mass transfer coefficient is defined as an output parameter of the simulations to facilitate comparison of the various process conditions. Specifically, a design of experiments (DOE) study is performed with agitation speed and gas flow rate as input parameters and volume-averaged mass transfer coefficient as the output parameter. ANSYS Workbench with DesignXplorer is used to run the DOE and study the bioreactor design space.

^{1,2} Marko Laakkonen, Development and validation of mass transfer models for the design of agitated gas-liquid reactors, <https://pdfs.semanticscholar.org/c6bd/d98a364a73fecb84468da9352659e475344d.pdf>

³ J.C. Lamont, D. S. Scott, An eddy cell model of mass transfer into the surface of a turbulent liquid, *AIChE J.* 16 (1970) 513-519

RESULTS

As shown in Figure 2, air bubbles undergo breakup near the impeller blades and coalesce in the circulation regions with low turbulent dissipation rates. This leads to bubble size varying throughout the tank. Since interfacial area depends on bubble size, bubble size distribution plays a critical role in oxygen mass transfer.

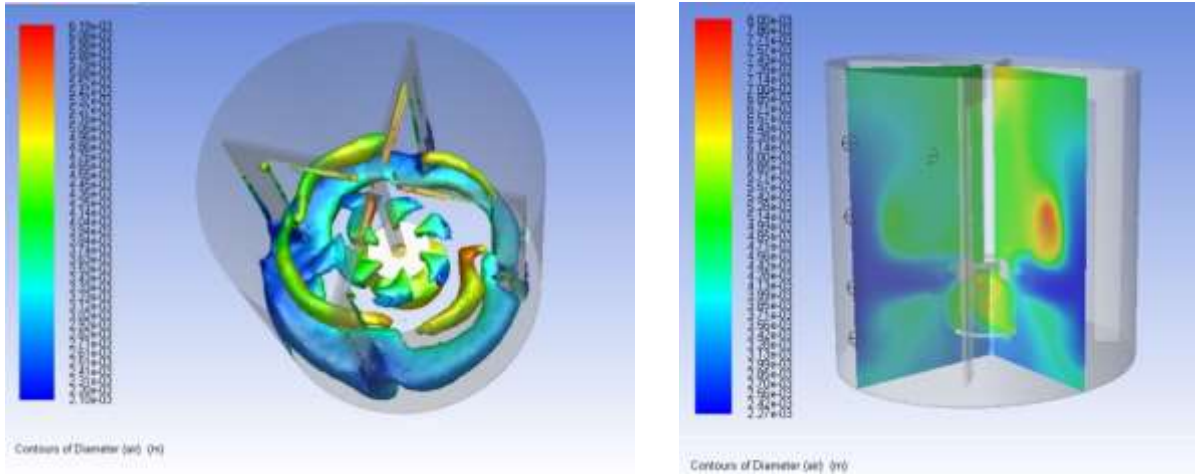


Figure 12: a) Iso-surface of gas volume fraction colored with bubble diameter b) Contour plot of bubble size distribution.

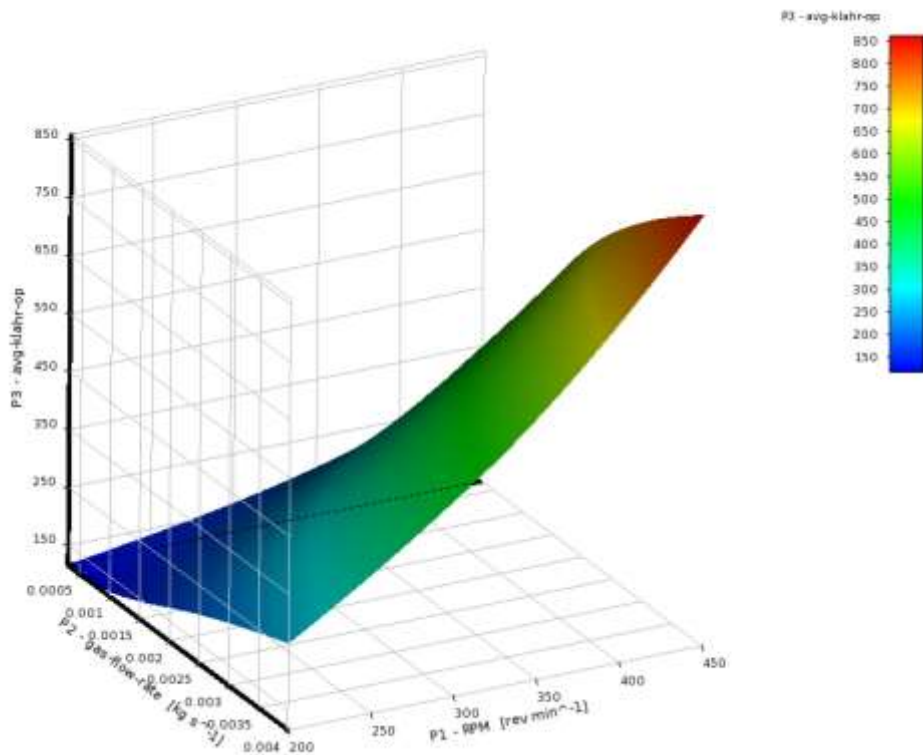


Figure 13: Response surface of average mass transfer coefficient versus gas flow rate and agitation speed.

To study the design space of the bioreactor, a DOE study has been performed to generate the response surface for the average mass transfer coefficient. From the response surface shown in Figure 3, we can see that agitation speed has a greater impact on the mass transfer coefficient versus gas flow rate. Even though we can increase the agitation speed to increase the mass transfer coefficient, there is a limit on maximum speed since some processes involve mammalian cells that are sensitive to hydrodynamic shear. Therefore, studying the design space with several input parameters provides an opportunity to optimize the operating conditions to identify a safe operational range for the bioreactor.

HPC PERFORMANCE BENCHMARKING

We used cloud resources in Microsoft’s Singapore data center because this is relatively close to the ANSYS office in Pune, India. The experiment start date was: 2017-12-27, and experiment finish date was: 2018-01-30. Simulations started on 1 node (16 cores) and the last run was on 16 nodes (256 cores). Instance node type: Standard_H16r; FDR InfiniBand (56 Gbps bandwidth); Azure compute instances: 16 CPU cores (Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz), 112 GB of memory.

The software used to simulate the gas sparging process is ANSYS Workbench with FLUENT in an UberCloud container integrated with the Microsoft Azure cloud platform. The solution methodology is tested with fine and coarse tetrahedral and polyhedral meshes. The time required for solving the model with different mesh densities is captured to benchmark the HPC performance in solving high density mesh models. Boundary conditions, solution algorithm, solver setup and convergence criteria were identical for all models.

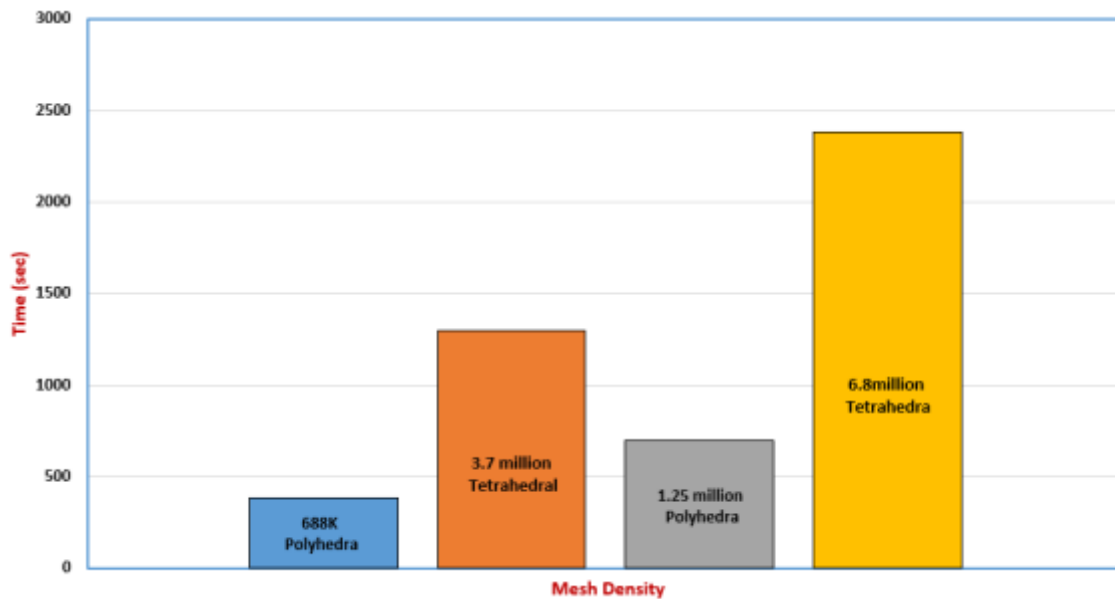


Figure 14: Run time comparison for different mesh densities using 24 CPU cores.

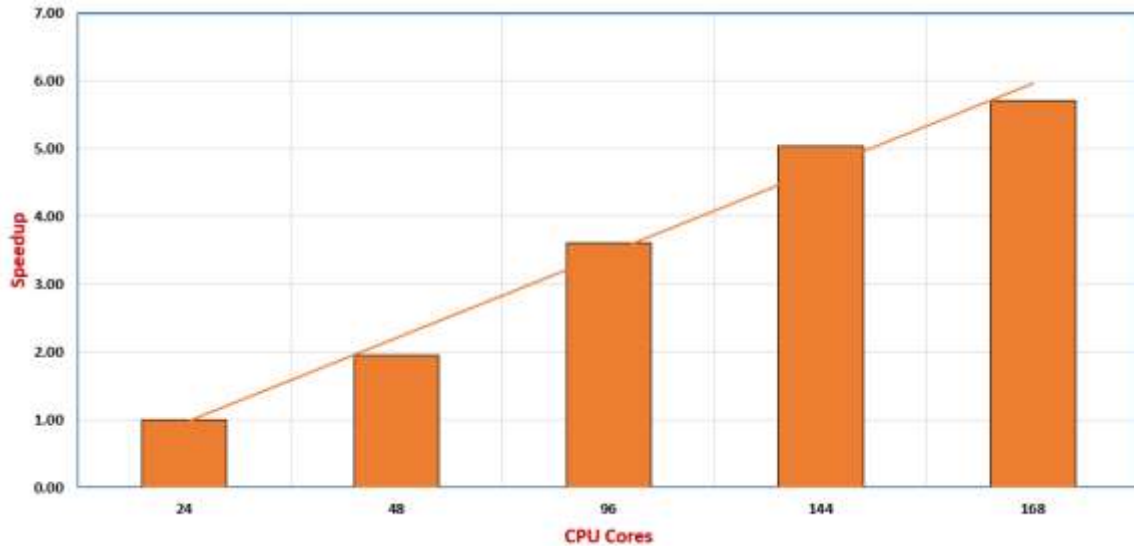


Figure 15: Speedup of 688K polyhedral mesh at different CPU cores.

Figure 4 compares the time required to run 200 iterations for different mesh densities with 24 CPU cores. The comparison of the solution time shows a significant reduction in the solution time when converting the meshes from tetrahedral to polyhedral. This is primarily due to the lower number of mesh elements with minimal impact on solution accuracy. Figure 5 summarizes the scalability study, which was based on the 688K polyhedral mesh. As can be seen from the figure, that the solution speed scales close to linear up to 168 CPU cores. Figure 6 shows the decrease of simulation run time as the number of cores is increased. When using 168 cores, each simulation takes less than an hour, making it possible to run the entire design space of the bioreactor in less than 24 hours.

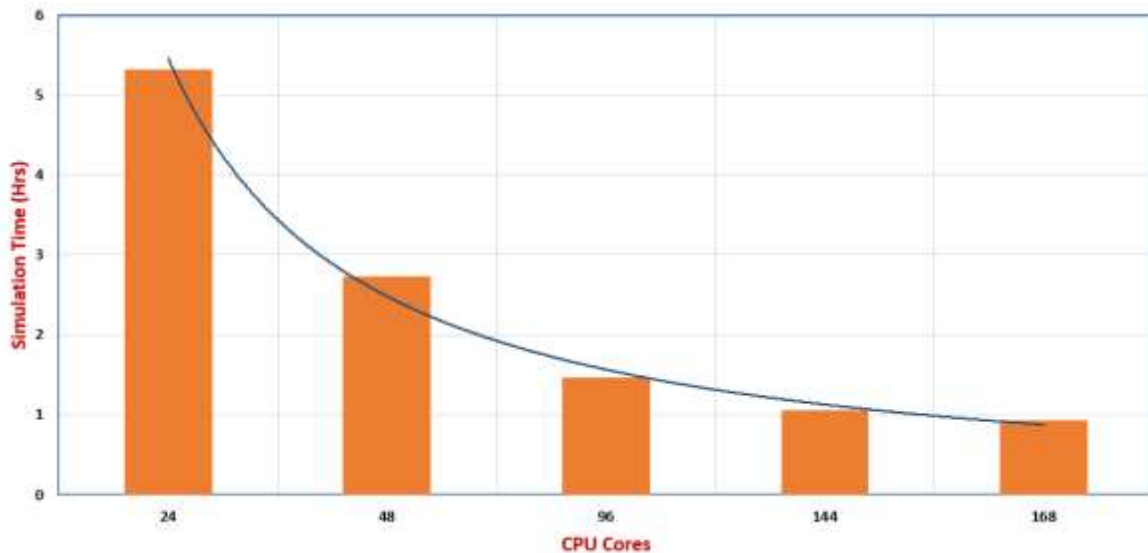


Figure 16: Simulation Run time comparison for 688K polyhedral mesh on different number of CPU cores.

A similar speedup study has been performed for the different types of meshes generated for this study. The solution speed scale-up results are plotted and compared with linear scale-up speed to compare the scale-up at different mesh densities. As shown in Figure 7, the solution speed scale-up is observed to move closer to the linear increase in solution speed as the mesh density increases.

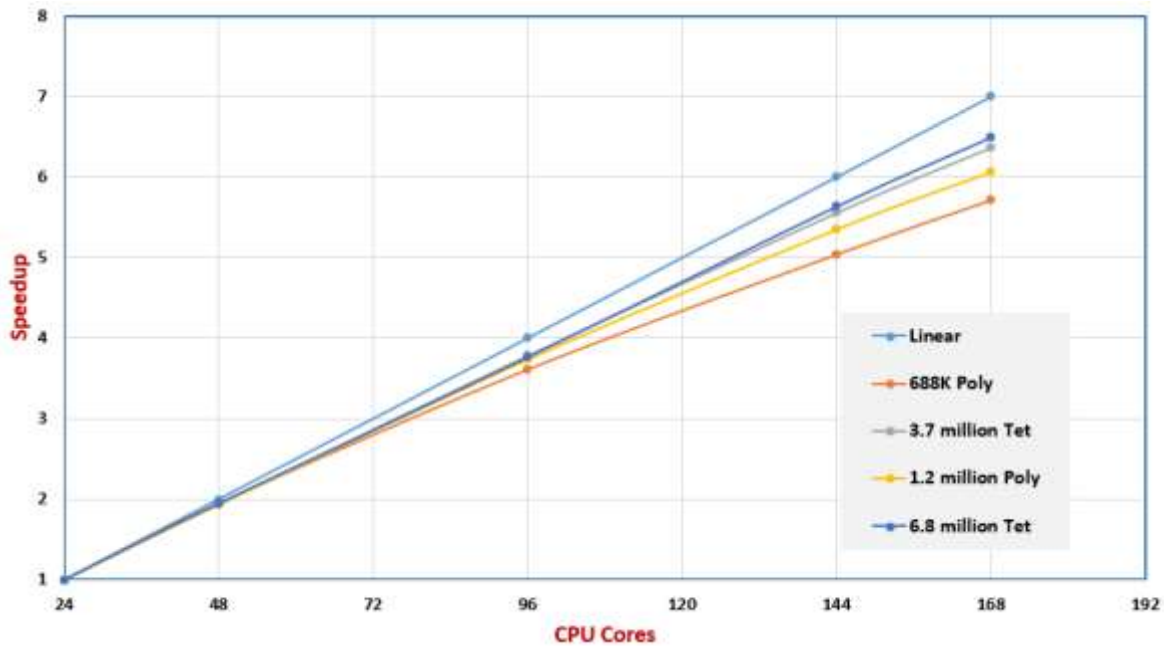


Figure 17: Comparison of solution speed scale-up with different mesh densities.

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux Docker container technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

EFFORT INVESTED

End user/Team Expert: 20 hours for simulation setup, technical support, reporting and overall management of the project.

UberCloud support: 4 hours for monitoring & administration of the performance in the host server.

Resources: ~2500 core hours were used for performing design of experiments study using ANSYS workbench.

BENEFITS

1. The HPC cloud computing environment with ANSYS Workbench with FLUENT and DesignXplorer streamlined the process of running a DOE with drastically reduced process time.
2. Running the 10 design point simulations and generating the response surface took only 24 hours of run time with 144 CPU cores. This means design engineers can quickly execute DOE analyses to study the scale-up behavior of their bioreactors.

3. With the use of VNC Controls in the web browser, HPC Cloud access was very easy with minimal installation of any pre-requisite software. The entire user experience was similar to accessing a website through the browser.
4. The UberCloud containers helped smooth execution and provide easy access to the server resources. The UberCloud environment integrated with the Microsoft Azure platform proved to be powerful as it facilitates running parallel UberCloud containers, with a dashboard in the Azure environment which helped in viewing the system performance and usage.

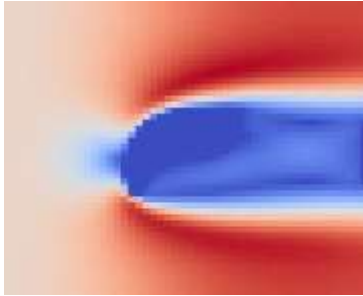
CONCLUSION & RECOMMENDATIONS

1. Microsoft Azure with UberCloud HPC resources provided a very good fit for performing advanced computational experiments that involve high technical challenges with complex geometries and multi-phase fluid flow interactions that would not typically be solved on a normal workstation, reducing the time required to establish a two-parameter design space for a bioreactor to a single day.
2. The combination of Microsoft Azure, HPC Cloud resources, UberCloud Containers, ANSYS Workbench with FLUENT, helped to accelerate the simulation trials and also completed the project within the stipulated time frame.

Case Study Author – Sravan Kumar and Marc Horner, ANSYS Inc.

Team 211

Deep Learning for Steady-State Fluid Flow Prediction in the Advania Data Centers Cloud



“The overhead of creating high volumes of samples can be effectively compensated by the high-performance containerized computing environment provided by UberCloud and Advania.”

1 MEET THE TEAM

End-User: Jannik Zuern, Renumics GmbH, Karlsruhe, Germany

Software Provider: OpenFOAM open source CFD software

Resource Provider: Advania Data Centers Cloud, Iceland

HPC and AI Experts: Stefan Suwelack, Markus Stoll, and Jannik Zuern, Renumics; Joseph Pareti, AI Consultant; and Ender Guler, UberCloud Inc.

2 USE CASE

Solving fluid flow problems using Computational Fluid Dynamics (CFD) is demanding both in terms of computer processing power and in terms of simulation duration. Artificial neural networks (ANN) can learn complex dependencies between high-dimensional variables. This ability is exploited in a data-driven approach to CFD that is presented in this case study. An ANN is applied in predicting the fluid flow given only the shape of the object that is to be simulated. The goal of the approach is to apply an ANN to solve fluid flow problems to significantly decrease time-to-solution while preserving much of the accuracy of a traditional CFD solver. Creating a large number of simulation samples is paramount to let the neural network learn the dependencies between simulated design and flow field around it.

This project between Renumics GmbH (Karlsruhe) and UberCloud Inc. was therefore established to explore the benefits of additional cloud computing resources that can be used to create a large amount of simulation samples in a fraction of the time a desktop computer would need to create them. In this project, we want to explore whether the overall accuracy of the neural network can be improved when more samples are being created in the UberCloud Container and then used during the training of the neural network. UberCloud kindly provided the cloud infrastructure, a CentOS Docker container with an OpenFOAM installation, and additional tech support during the project development.

3 WORKFLOW OVERVIEW

In order to create the simulation samples automatically, a comprehensive workflow was established.

As a **first step**, random two-dimensional shapes are created. These shapes have to be diverse enough to let the neural network learn the dependencies between different kinds of shapes and their respective surrounding flow fields.

In the **second step**, these shapes are meshed and added to an OpenFOAM simulation case template (Fig. 1). This template is simulated using the steady-state solver OpenFOAM solver simpleFOAM.

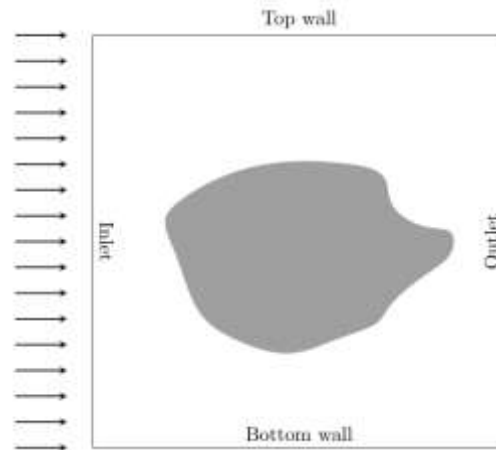


Figure 1: Simulation case setup. The flow enters the simulation domain through the inlet, flows around the arbitrarily shaped obstacle (dark grey shade) and leaves the simulation domain through the outlet.

In the **third step**, the simulation results are Post-Processed using the open-source visualization tool ParaView. The flow-fields are resampled on a rectangular regular grid to simplify the information processing by the neural net.

In the **fourth and final step**, both the simulated design and the flow fields are fed into the input queue of the neural network. After training, the neural network is able to infer a flow field merely from seeing the to-be-simulated design.

In Figure 2, a visualization of the four-step Deep Learning workflow is shown.

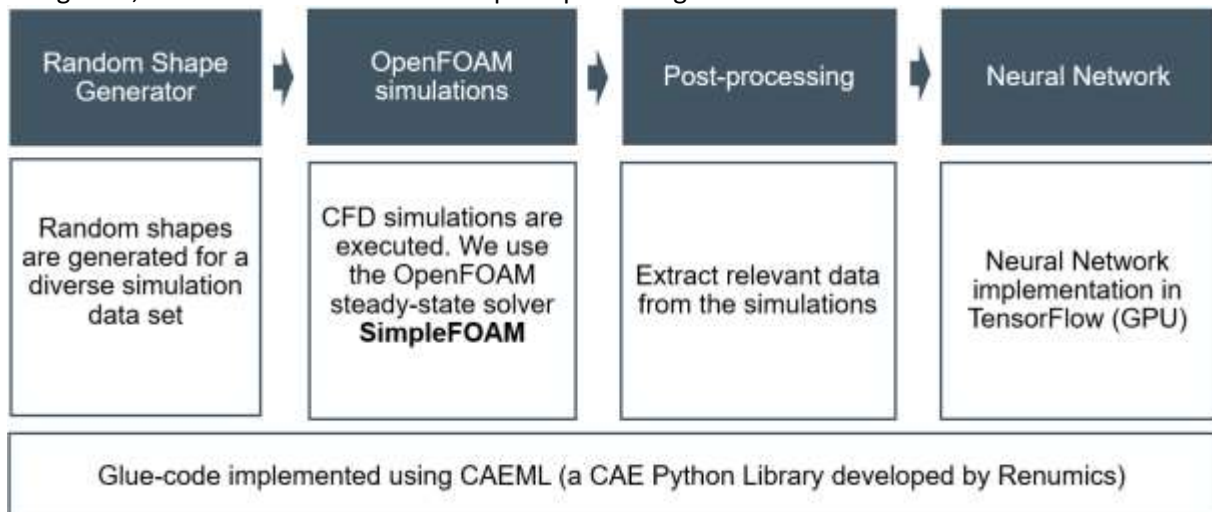


Figure 2: Deep Learning workflow.

Hardware specs

The hardware specs of the Advania Data Centers compute node hosting the UberCloud container are as follows:

- 2 x 16 core Intel Xeon CPU E5-2683 v4 @ 2.10 GHz
- GPU: none
- Memory: 251 GB

The hardware specs of the previously used desktop workstation are as follows:

- 2 x 6 core Intel i7-5820K CPU @ 3.30 GHz
- GPU: GeForce GTX 1080 (8GB GDDR5X memory)
- Memory: 32 GB

4 RESULTS

Time needed to create samples

As a first step, we compared the time it takes to create samples on the desktop workstation computer with the time it takes to create the same number of samples on the UberCloud container. Figure 3 illustrates the difference in time it took to create 10,000 samples

On the desktop computer it took 13h 10min to create these 10,000 samples. In the UberCloud OpenFOAM container in the Advania Data Centers Cloud, it took about 2h 4min to create 10,000 samples, which means that a speedup of 6.37 could be achieved using the UberCloud container.

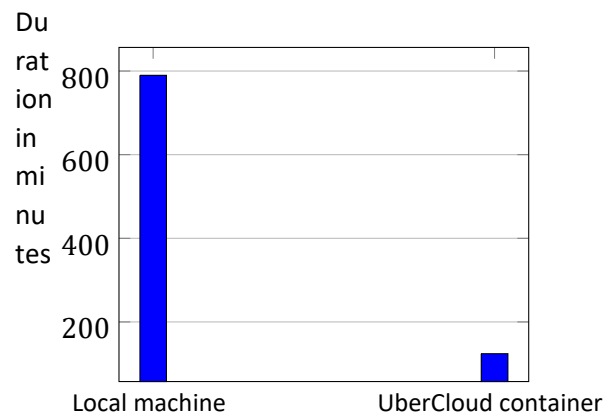


Figure 3: Comparison between Local machine and UberCloud container.

Neural Network performance evaluation

A total of 70,000 samples were created. We compare the losses and accuracies of the neural network for different training set sizes. In order to determine the loss and the accuracy of the neural network, we first must define, what these terms actually mean.

■ Performance for generating the flow field data set and Tensorflow training

Setup	2-D external flow	3-D internal flow
Time for 10.000 simulations	13.2 h	152.5 h
Time for training	23.7 h	48.5 h

■ Neural network prediction of flow field

Setup	2-D external flow	3-D internal flow
Time for CFD solver	4.7 s	55.0 s
Time of neural network prediction	3 ms	120 ms
Speedup factor with deep leaning	1566	458

Figure 4: Performance and speedup of flow simulations with neural network prediction.

Definitions

Loss: The loss of the neural network prediction describes how wrong the prediction of the neural network was. The output, or prediction, of the neural network in our project is a $N \times M \times 2$ tensor since the network tries to predict a fluid flow field with N elements in x -direction, M elements in y -direction, and two flow velocity components (velocity in x -direction and velocity in y -direction). A mean-squared-error metric was used to calculate the loss l :

$$l = \frac{1}{2} \sum_{d=0}^1 \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (v_{dij} - \bar{v}_{dij})^2 \quad (1)$$

where v_{dij} denotes the ground-truth velocity component in dimension d at the grid coordinates (i, j) , \bar{v}_{dij} denotes the predicted velocity component at the same position and in the same dimension. The goal of every machine learning algorithm is to minimize the loss of the neural network using numerical optimization schemes such as Stochastic Gradient Descent. Thus, a loss of 0.0 for all samples would mean that every flow velocity field in the dataset is predicted perfectly.

Accuracy: In order to be able to make sensible statements about the validity of the prediction of the neural network, metrics have to be defined that describe the level of accuracy that the neural network achieves. In general, the accuracy of a neural network describes how accurate the prediction of the neural network was. While the loss of a neural network is the metric that is being minimized during training, a small prediction loss does not necessarily mean that the corresponding prediction is physically meaningful. In general, however, a small prediction loss usually corresponds with a high accuracy. Different measurements of how accurate the outputs of the neural network are needed to express the validity of the predictions. A highly accurate prediction should have high values for all formulated accuracy measurements and a low loss at the same time. These accuracies can have values between 0.0 and 1.0, where an accuracy of 0.0 indicates that the prediction of the neural network does not at all coincide with the ground truth flow metric that is examined, and an accuracy of 1.0 means that the prediction coincides perfectly with the ground truth flow metric. Bear in mind that a low loss does not necessary cause high accuracy and vice versa. However, the two measurements are typically correlated.

In this study, two different accuracies were evaluated: Divergence accuracy and Drag accuracy:

- **Divergence accuracy:** Numerical CFD solvers aim to find a solution to the continuity equation and the momentum equation. For an incompressible fluid, the continuity equation dictates that the divergence of the velocity vector field is zero for every point in the simulation domain. This follows the intuition that at no point in the simulation domain fluid is generated (divergence would be greater than zero) or ceases to exist (divergence would be smaller than zero). By design, the Finite Volume Method preserves this property of the fluid even in a discretized form. A data-driven approach should as well obey this rule.
- **Cell accuracy:** The number of correctly predicted grid cells in the two- or three-dimensional grid yields an intuitive metric for how well the neural network predicts fluid flow behavior. As the network will never be able to predict the fluid flow velocity down to the last digit of a floating-point number, the following approach is proposed: If the relative error between the network prediction and the actual flow velocity is smaller than 5%, the respective grid cell is declared as predicted correctly. The cell accuracy can be calculated by counting the number of correctly predicted grid cells and dividing the results by the total number of grid cells.

5 TRAINING RESULTS

The generated samples are divided into the training and validation datasets. The training- and validation loss for different numbers of training samples was evaluated. Concretely, the neural net was trained three times from scratch with 1,000, 10,000, and 70,000 training samples respectively. The following training parameters were used for all neural network training runs:

- Batch size: 32
- Dropout rate: 0.5
- Learning rate: 5×10^{-4}
-

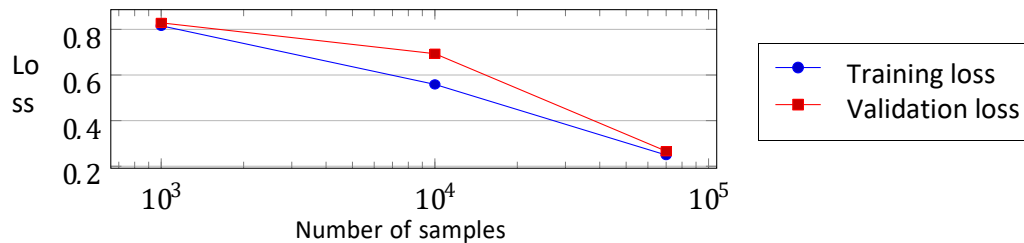


Figure 5: Loss after 50,000 training steps.

It can be observed that both training- and validation losses are lowest for the 70k samples training and are highest for the 1k training samples. The more different samples the neural network processes during the training process the better faster it is able to infer a flow velocity field from the shape of the simulated object suspended in the fluid. The validation loss tends to be higher than the training loss for all tested numbers of samples, which is a typical property of machine learning algorithms. Figure 6 shows the loss after 300,000 training steps:

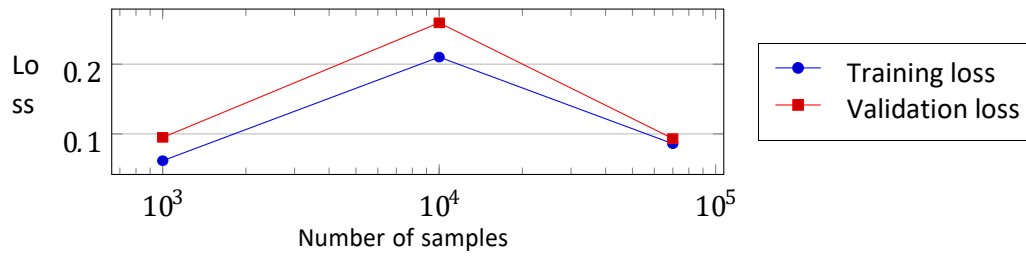


Figure 6: Loss after 300,000 training steps.

Surprisingly, the final training- and validation losses for the 70k samples training session are as low as the losses for the 1k samples training session. Generally speaking, no clear tendency towards lower losses when increasing the set of the training samples could be observed. This result is somewhat surprising since we expected the final losses at the end of the training process to show a similar tendency towards lower losses for higher numbers of samples. We assume that the number of samples does not heavily influence the final loss for extensive training sessions with many hundreds of thousand training steps. Finally, in Figure 7 the divergence and grid accuracies are visualized.

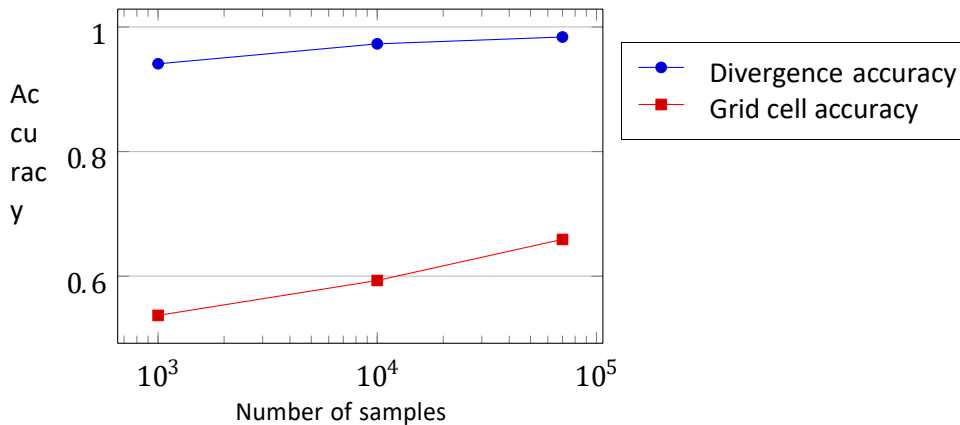


Figure 7: Validation accuracies after training.

Both the divergence accuracy and the grid cell accuracy show higher values for larger numbers of samples. While the divergence accuracy shows overall high values going from 0.94 for 1,000 samples to 0.98 for 70,000 samples, the grid cell accuracy also increases from a value of 0.53 for 1,000 samples to a value 0.66 for 70,000 samples. To recap: a grid accuracy of 0.66 means that approximately two thirds of all velocity grid cells were predicted correctly within 5% relative error to the correct value.

Figure 8 illustrates the difference between the ground truth flow field (left image) and the predicted flow field (right image) for one exemplary simulation sample after 300,000 training steps. The arrow direction indicates the flow direction and the arrow color indicates the flow velocity. Visually, no difference between the two flow fields can be made out.

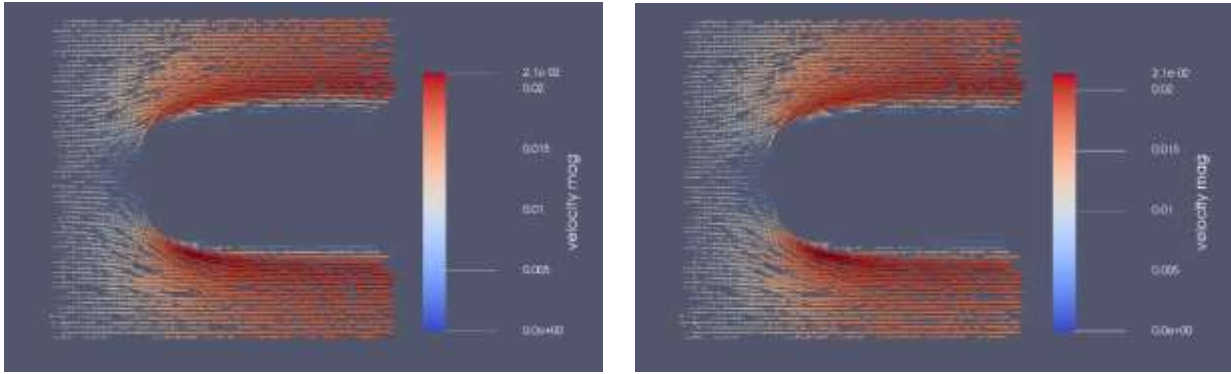


Figure 8: Exemplary simulated flow field (left image) and predicted flow field (right image).

UBERCLOUD HPC SOFTWARE CONTAINERS

In 2015, based on our experience gained from the previous cloud experiments, we reached an important milestone when we introduced our new UberCloud HPC software containers based on Linux [Docker container](#) technology. Use of these containers shortened project times dramatically, from an average of three months to just a few days. Containerization drastically simplifies the access, use and control of HPC resources, applications, and data, whether on premise or remotely in the cloud. Essentially, users are working with a powerful remote desktop in the cloud that is as easy and familiar to use as their regular desktop workstation. Users don't have to learn anything about HPC, nor system architecture, nor cloud, for their projects. This approach will inevitably lead to the increased use of HPC for every engineer's daily design and development, even for novice HPC users. That's what we call democratization of HPC.

CONCLUSION

We were able to prove a mantra amongst machine learning engineers: *The more data the better*. We showed that the training of the neural network is substantially faster using a large dataset of samples compared to smaller datasets of samples. Additionally, the proposed metrics for measuring the accuracies of the neural network predictions exhibited higher values for the larger numbers of samples. The overhead of creating high volumes of additional samples can be effectively compensated by the high-performance containerized (based on Docker) computing node provided by UberCloud on the Advania Data Centers Cloud. A speed-up of more than 6 compared to a state-of-the-art desktop workstation allows creating the tens of thousands of samples needed for the neural network training process in a matter of hours instead of days.

In order to train more complex models (e.g. for transient 3D flow models) much more training data will be required. Thus, software platforms for training data generation and management as well as flexible compute infrastructure will become increasingly important.

Join our free and voluntary UberCloud Experiment

If you, as an end-user, would like to participate in this Experiment to explore hands-on the end-to-end process of on-demand Technical Computing as a Service, in the Cloud, for your business then please register at: <http://www.theubercloud.com/hpc-experiment/>

If you, as a service provider, are interested in promoting your services on the UberCloud Marketplace then please send us a message at <https://www.theubercloud.com/help/>

Annual UberCloud Compendiums:

- 1st Compendium of case studies, 2013: <https://www.theubercloud.com/ubercloud-compedium-2013/>
- 2nd Compendium of case studies 2014: <https://www.theubercloud.com/ubercloud-compedium-2014/>
- 3rd Compendium of case studies 2015: <https://www.theubercloud.com/ubercloud-compedium-2015/>
- 4th Compendium of case studies 2016: <https://www.theubercloud.com/ubercloud-compedium-2016/>
- 5th Compendium of case studies 2018: <https://www.theubercloud.com/ubercloud-compedium-2018/>

International Awards for UberCloud Technology & Community Contributions:

- 2013 HPCwire Readers Choice Award: <http://www.hpcwire.com/off-the-wire/ubercloud-receives-top-honors-2013-hpcwire-readers-choice-awards/>
- 2014 BioPharma Research Council D2D Innovation Award for UberCloud for its “Most Innovative Technology”.
- 2014 HPCwire Readers Choice Award: <https://www.theubercloud.com/ubercloud-receives-top-honors-2014-hpcwire-readers-choice-award/>
- 2015 Gartner Names The UberCloud a Cool Vendor in Oil & Gas: <https://www.hpcwire.com/off-the-wire/gartner-names-ubercloud-a-cool-vendor-in-oil-gas/>
- 2017 HPCwire Editors Choice Award: <https://www.hpcwire.com/2017-hpcwire-awards-readers-editors-choice/>
- 2017 IDC/Hyperion Innovation Excellence Award: <https://www.hpcwire.com/off-the-wire/hyperion-research-announces-hpc-innovation-excellence-award-winners-2/>
- 2018 HPCwire Editors Choice Award: <https://www.theubercloud.com/ubercloud-receives-top-honors-2018-hpcwire-readers-choice-award/>
- 2018 IDC/Hyperion Innovation Excellence Award: <https://insidehpc.com/2018/11/ubercloud-wins-hyperion-hpc-innovation-excellence-award-neuromodulation-project/>
- 2018 IDC/Hyperion Innovation Excellence Award: <https://www.digitalengineering247.com/article/ubercloud-wins-three-awards-for-cae-in-the-cloud>

If you wish to be informed about the latest developments in technical computing in the cloud, then please register at <http://www.theubercloud.com/> and you will get our free monthly newsletter.



Please contact [UberCloud help@theubercloud.com](mailto:UberCloud_help@theubercloud.com) before distributing this material in part or in full.
© Copyright 2019 TheUberCloud™. UberCloud is a trademark of TheUberCloud Inc.