Bartosz Poławski, Bartosz Sawicki

Warsaw University of Technology

doi:10.15199/48.2017.12.36

Distributed mesh generation for objects with shape variability

Abstract. *The article presents an efficient way to generate a large number of finite element meshes to support statistical analysis of simulation of electromagnetic field inside biological objects which are characterized by a strong shape variability. Demonstrated solution uses capabilities of modern distributed computer systems and open source software. Parametric model of a hen egg has been used as a simple biological object to test the developed solution.*

Streszczenie. *Praca przedstawia efektywną metodę generowania wielu siatek elementów skończonych dla potrzeb modelowania pola elektromagnetycznego w obiektach o pochodzeniu biologicznym charakteryzujących się dużą zmiennością kształtu i ich statystycznej analizy. Do rozwiązania problemu zostały wykorzystane możliwości nowoczesnych systemów informatycznych o architekturze rozproszonej i otwartoźródłowego oprogramowania. Do celów testów zaproponowanego rozwiązania został stworzony parametryczny model kurzego jaja jako przykład prostego obiektu biologicznego.* ***(Rozproszone generowanie siatek dla obiektów o zmiennych kształtach)***

Keywords: meshing, HPC, distributed computing, variability, parameter sweeping
Słowa kluczowe: generowanie siatek, HPC, obliczenia rozproszone, analiza zmienności

Introduction

Traditional methods of computer modeling require strictly determined shapes of objects. Such assumption is justified in case of technical objects which are often produced as a result of repetitive factory process. However, authors interest is placed on objects of natural origin, like human body (see Fig. 1). Growth process is unbelievably complex hence each individual has its own unique shape. In development of new methods of computer modeling shape variability should be taken into account as it is necessary for reliable results of analysis.

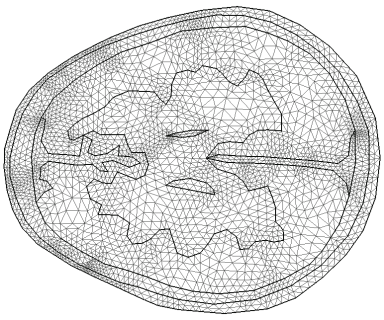

Fig. 1. Human brain is an example of object with high shape variability (uniqueness of the brain folding is similar to fingerprints).

Simulation of electromagnetic field inside of a human body is an example of this kind of problems. Shape variability of the internal body organs is a subject of uncertainty [6]. There are different reasons for the fact that geometry of the computer model is not precisely defined. First of all some organs can naturally change its size and position (e.g. heart or lungs). Another sources of variability are related with differences between individuals. This is particularly important when building a generic model of the body used for numerical verification of safety of the exposure. Shape uncertainty affects also process of creation of a computer model based on a real object. Usually shapes of the body is taken from medical imaging modalities (MRI, CT), using process called segmentation, tissues regions has to be selected on each slice. Segmentation is another source of uncertainty [2]. Shapes of organs are usually more complicated than model resolution, moreover border line between different tissues could be fuzzy what leads into ambiguous distinction. Models created using segmented data could have very high quality, but they are time consuming and expensive, so number of models which have been created that way is relatively small. Large portion of them is based on single body from VisibleHuman project [5], so they can not be used for statistical reasoning.

In order to control the uncertainty related to organ shapes, a procedural generation of a large number of various models with any desired discretization resolution of the same organ can be used. Large number of solutions which could be obtained this way would significantly increase reliability of statistical analysis.

Three basic techniques of generation of biological objects with shape variability can be distinguished depending on the degree of its naturalness:

- simulate natural growth from a variable "seed" and under variable mechanical conditions [8],
- morphological transformations of an averaged real shape [1],
- generation through changes of parameters of a simplified parametric model [7].

However all of the above techniques require a signifi-cant computer resources to get statistical population of ob-jects with acceptable mesh resolution. Essential computer power can be delivered by platforms for distributed computing, for example based on the popular Map-Reduce frame-work [4]. In this article authors present a scalable architec-ture of distributed computing system and a process for its' au-tomatic deployment in a cloud computing environment. The Microsoft Azure services has been chosen as a platform for the proposed computing system. For the purpose of testing the parametric model of a hen egg has been created. Fi-nal conclusions are based on results for 10.000 of biological objects of the same type but different in shape and size.

Infrastructure

In order to distribute mesh generation tasks over networked computers, High Performance Cluster was created. Architecture of the designed system is presented on the Fig. 2. Diagram shows relations between three major tech-nologies which have been used:

- Microsoft Azure Cloud Computing Services to provide server and network infrastructure;
- Docker Container Platform to provide packaged runtime environment which is easy to distribute;
- GNU Parallel tool to distribute and parallelize computations.

Task of producing many meshes of objects with shape variability can be boiled down to a task of carrying out same calculations for different input data multiple times. For such problem the most natural way to assure concurrency of operations within a computer cluster would be parallel executions of computations for next variant of input data as a separate process on a node (or a core of a node) which is not occupied

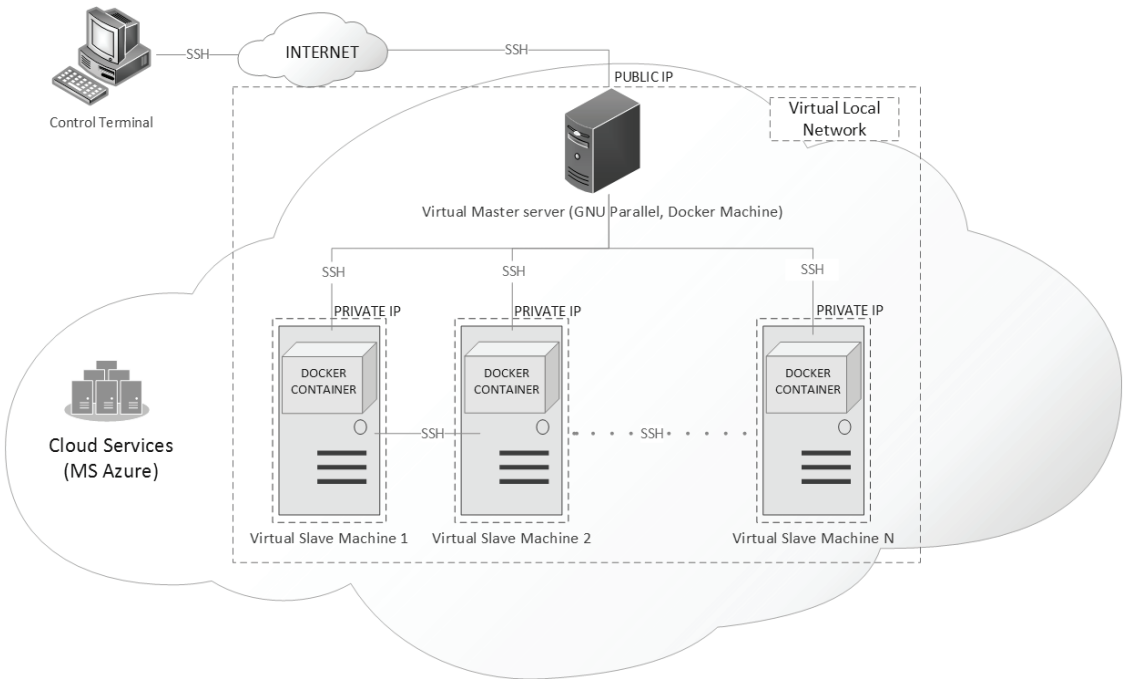

Fig. 2. Architecture of the developed system

at a given moment. In order to keep efficiency, software which is managing tasks execution should guarantee that calculations for next variant of input data will be started immediately when one of the nodes finished calculations.

GNU Parallel [9] is a software thanks to which above assumptions can be achieved. It is a shell tool which allows to execute tasks parallel on one or many computers and provides control of amount of jobs and amount of cores utilized at one time. It is an open source software available under the terms of GPLv3. Parallel is being developed from 2001, in 2010 was adopted as an official GNU tool and name was changed to GNU parallel. Currently it is a very stable software for which packages are available for all major Linux distributions. The version of the program that was available at the time of the article creation was 20170722. A comprehensive documentation, wide possibilities and simplicity of execution of GNU Parallel suggests that it has high utilization potential. For instance, it can be used to replace sequential loops in scripts with parallel execution of body of a loop, as well as it can serve as a job queue manager. It uses SSH protocol to communicate with remote machines. Input data for a serial task that is to be executed by GNU Parallel can be passed via a file, a command line, a standard input or a pipe. Here is a simple example of execution of a serial task parallel on many servers (thousand of executions of the "./mesher.py" command on computers which are listed in the "nodefile" file:

```
$ seq -w 1000 | parallel --sshloginfile nodefile \
  ./mesher.py mesh{}.xml
```

Cloud environment has been used to perform distributed calculations. An architecture based on a cloud environment allows to avoid troublesome and time-consuming activities related to physical administration of computer units and a network infrastructure. Utilization of such platform provides a number of additional benefits such as ease of creation, administration and stability of each of cluster nodes, possibility of elastic change of hardware resources depending on needs at a given moment. To build a computer cluster we used the Microsoft Azure Cloud Computer Services platform which is one of the biggest Cloud Service provider next to AWS, Google and IBM. Microsoft meets now needs of open source community and MS Azure provides plenty of images of preconfigured Linux machines images and supports lightweight virtualization based on Docker [3] containers.

Docker [3] is a software to create and run application image on a given machine in a separate container. Such container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: application itself, runtime environment, file system, system tools, system libraries, specific settings. For each container Docker allocates a separate primary memory area, separate network interface with private IP address and separate disk storage area with OS image that contains all dependencies and libraries required by application. Thanks to that Docker allows to avoid all complicated operations which would need to be performed (or scripted) on each of cluster nodes separately in order to prepare it for a specific task execution. Instead, all of what has to be done is to create a reusable Docker container image which contains all of complex dependencies and configuration only once and distribute it across the cluster nodes. There are plenty of official Docker container images available for free on Docker hub (https://hub.docker.com/explore/) or via websites of interested parties. Those images can be treated as generic images ready for modifications required for a given project. E.g. in our case we started from modification of official Fenics Docker container image (available via the Fenics project website). For our purposes we had to run downloaded image as a container, connect to the container via ssh, modify code and recompile a huge part of the Fenics platform, tune some system configuration and deploy our application. After applying required changes we created a new container image from the modified container ready to distribute across the cluster nodes. Docker containerization may seem similar to virtualization however Docker runs an application in an isolated, dedicated container with no need of emulation hardware layer and operating system. It uses native kernel and normal system calls. Inside a container only processes of a desired application are run and nothing more. This results in minimal overhead and more efficient use of hardware resources compared to virtualization. Docker con-

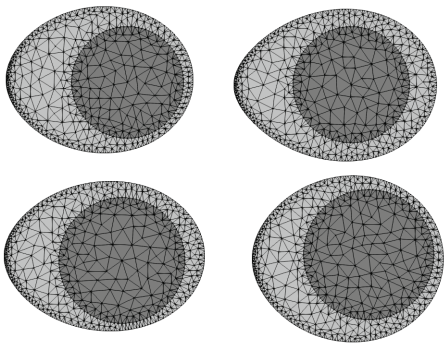

Fig. 3. Different eggs meshes

tainerization provides a way for perfect software portability. This feature is very beneficial also for the described architecture of distributed computing platform as it allows to use cluster nodes with any of up-to-date Linux distribution, especially with those which are very light and optimized for containerization (e.g. CoreOS, Ubuntu Snappy, RancherOS, Red Hat's Atomic Host). Currently there is a strong competition between container-concentric OSs and thanks to container portability the OS which fits best at given moment can be chosen.

Docker Machine is a shell tool to manage Docker Engine and containers on virtual hosts. Authors used Docker Machine to create virtual hosts mostly for two reasons. Firstly it is the most natural choice to automate works with Docker containers, secondly it is an universal solution and allows to apply the same procedure in environments provided by many cloud services providers.

With a usage of the technologies described above authors developed a bash script that automates creation of the computer cluster which is tailored to solve the considered problem. Automatic created cluster is composed of a given number of virtual machines and each of them has a deployed Docker container for the specific applications and their runtime environment which are dedicated to mesh generation for objects with shape variability. Network communication within the cluster takes place via SSH protocol. The script performs following tasks: creation of Docker machine, distribution of SSH Public key, remote GNU Parallel installation, loading container image to Docker engine and starting Docker container. In our case script was executed from the cluster's virtual master server (see Fig. 2). Script can be executed any time when a cluster infrastructure is required for computations. Once cluster is setup it's infrastructure is ready for parallel execution of a given task with a usage of the GNU Parallel and a generated nodefile list. After finish computations resources can be dealocated. Such approach allows to minimize financial costs. According to our tests full time of preparing of one machine takes about 15 minutes where the most time consuming part is loading 2.2 GiB into Docker instance (about 8 minutes). However process and total time of automatic cluster deployment can be quite easy accelerated by introducing quasi recurrent deployment. Virtual machines which have been already deployed during cluster building process can be used to deploy next cluster nodes. E.g. again GNU Parallel can be engaged to support such deployment procedure.

Meshing

A hen egg has been chosen as a simple biological object which varies in shape and size and can be used to examine the proposed solution for distributed calculations for shape generation and meshing. Computer model of an egg shaped

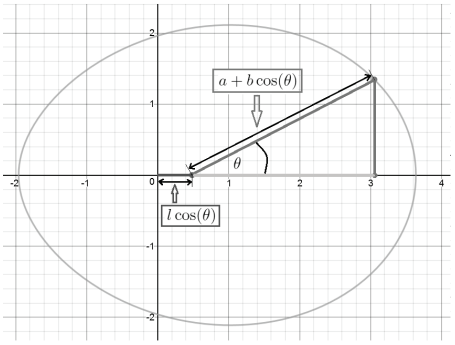

Fig. 4. Egg shaped curve

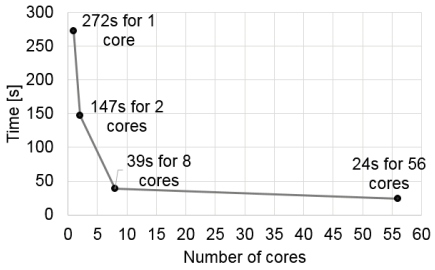

Fig. 5. Computation time comparison for production of 40 meshes for different amounts of cores in a cluster

curve has been created based on the following parametric equations [10] for x and y coordinates:

$$x(\theta) = l\cos(\theta) + (a + b\cos(\theta))\cos(\theta), \tag{1}$$

$$y(\theta) = (a + b\cos(\theta))\sin(\theta), \tag{2}$$

where l, a, b are constants which influence on the value of the above expressions is illustrated on the Fig. 4.

Based on the above equations we created an algorithm which randomly, in accordance with Gaussian distribution, changes parameters in a reasonable range to produce a random egg shaped curve (see Fig. 3). The algorithm generates also a randomly sized and placed yolk as a sub-domain for meshing.

During the tests to build a cluster only Ubuntu 16.04.1 LTS machines with the following hardware parameters were used: 2x2.1GHz cores, RAM 3,5 GiB, Disk 135 GiB.

The Fig. 5 shows a computation time comparison for production of 40 meshes for different amounts of cores in a cluster. As this relationship has non-linear character it can be concluded that the platform for distributed computations generates a certain overhead depending on the size of cluster. Those indirect time costs are most probably caused by the data transfer through the network and the task distribution management. Thus depending on the amount of tasks, complexity and time consumption of a given single computational task there is a certain threshold of number of cluster nodes for which increasing the number of nodes is not much profitable.

Graph on the Fig. 6 shows the time of computations performed on a cluster composed of 28 dual-core machines versus amount of generated meshes. According to expectations it is a linear relationship.

As a final result of performance tests of the cluster composed of 28 dual-core machines we produced 10.000 meshed cross-sections (see Fig. 3) of different eggs in 22 minutes. Computational load was evenly balanced across the cluster (see Fig. 7). The Fig. 8 proves reality of obtained

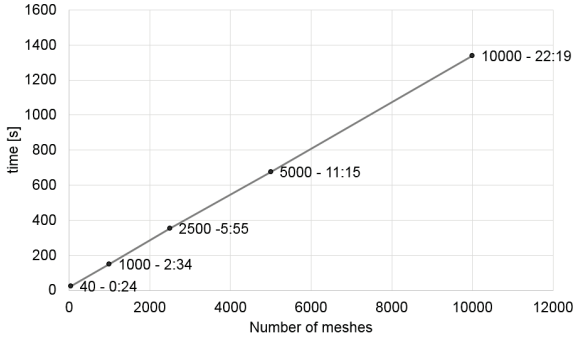

Fig. 6. Time of computations performed on the cluster composed of 28 dual-core machines versus amount of generated meshes

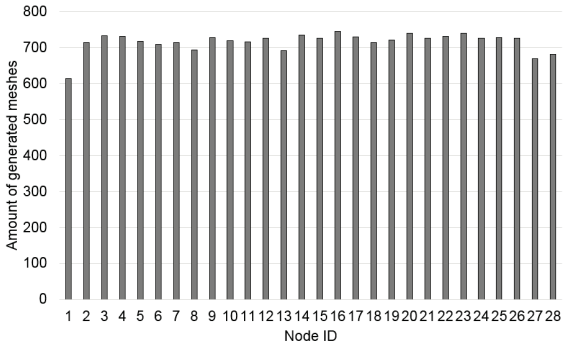

Fig. 7. Distribution of 10.000 generated meshes per cluster nodes

results. It shows that histogram of generated eggs' weight corresponds to normal distribution.

Large number of obtained meshes could significantly increase reliability of statistical analysis of electromagnetic field inside of a biological object, e.g. simulation of eddy currents inside a hen egg placed in a varying magnetic field (see Fig. 9). Such applications are further research objective of the authors.

Conclusions

Developed platform for distributed computations created in the cloud environment with a usage of Docker and GNU Parallel works as expected and significantly accelerate production of meshes for objects with shape variability. However the platform can be used also for other tasks that have serial, independent, repetitive character and can be boiled down to execution of the same algorithm for different input parameters (e.g. series of simulations). The possibility of automation of the deployment makes the cluster provisioning process efficient and scalable. Regarding the scalability, the overhead factor should be taken into consideration to optimize cluster deployment time and avoid additional costs for unnecessary cluster nodes. However from the economic point of view financial costs of utilization of cloud services are negligible (below 0.05 USD per hour of server) comparing to classical clusters investments.

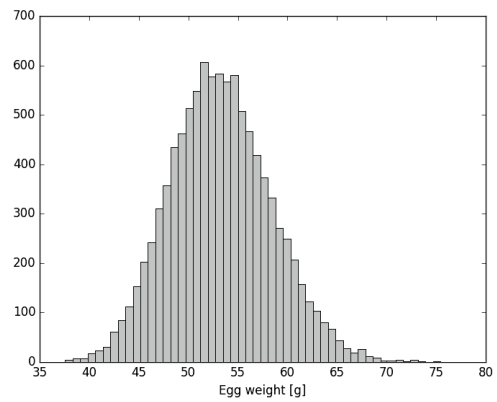

Fig. 8. Histogram of 10.000 generated eggs' weight

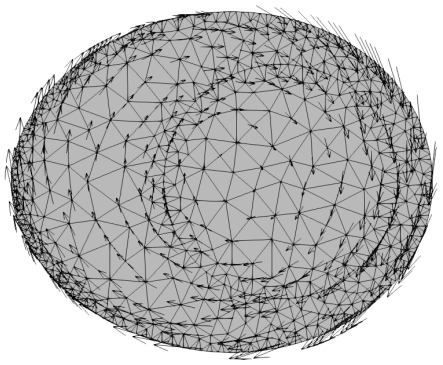

Fig. 9. Example of simulation of eddy currents effect in one of generated egg mesh

***Authors**: Bartosz Poławski, Bartosz Sawicki, Institute of Theory of Electrical Engineering, Measurement and Information Systems, Faculty of Electrical Engineering, Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warszawa, Poland, email: bartosz.polawski@gmail.com, bartosz.sawicki@ee.pw.edu.pl*

REFERENCES

[1] M. Borysiak, Z. Krawczyk, J. Starzynski, R. Szmurło, S. Wincenciak: CUDA accelerated finite element mesh morpher. Przegląd elektrotechniczny. R. 81, 2011

[2] Ivana Despotović, Bart Goossens, Wilfried Philips: MRI Segmentation of the Human Brain: Challenges, Methods, and Applications, Computational and Mathematical Methods in Medicine, 23 pages, 2015.

[3] J. Higgins, V. Holmes, and C. Venters: Orchestrating Docker Containers in the HPC Environment, ISC High Performance, 2015, Germany

[4] Krupa, Artur, Bartosz Sawicki Massive Simulations Using MapReduce Model, Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska, 2015. Vol. 5, p. 45–47

[5] Andreas Pommert, Karl Heinz Höhne, Bernhard Pflesser, Ernst Richter, Martin Riemer, Thomas Schiemann, Rainer Schubert, Udo Schumacher, Ulf Tiede: Creating a high-resolution spatial/symbolic model of the inner organs based on the Visible Human, In Medical Image Analysis, Volume 5, Issue 3, 2001

[6] Bartosz Sawicki: Uncertainty of numerical simulations in bioelectromagnetic problems, Przegląd Elektrotechniczny, R. 91, Nr 7/2015, pp. 49-51

[7] Martin Styner, Guido Gerig: Three-dimensional medial shape representation incorporating object variability, Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR),

[8] Tuomas Tallinen, Jun Young Chung, François Rousseau, Nadine Girard, Julien Lefèvre, L.Mahadevan: On the growth and form of cortical convolutions, Nature Physics, 2016.

[9] O. Tange: GNU Parallel - The Command-Line Power Tool, login: The USENIX Magazine, pp. 42-47, February 2011

[10] Nobuo Yamamoto: Equation of Egg Shaped Curve, http://www.geocities.jp/nyjp07/index_egg_E.html#3 (access: 2017-10-20)