In Proc. of the Third Int. Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering,
Civil-Comp Press, 2015 (DOI: 10.4203/ccp.107.48)

High-performance fluid flow simulations through porous media: coupling micro- and macro-scale on geometrically complex domains

N. Perović, J. Frisch, R.-P. Mundani, E. Rank

Chair for Computation in Engineering
Technische Universität München, Munich, Germany

Preprint

In Proc. of the Third Int. Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering,
Civil-Comp Press, 2015 (DOI: 10.4203/ccp.107.48)

Abstract

High-performance fluid flow simulation through geometrically complex large physical domains appears to be well-known, already studied and often performed case, yet, with numerous challenging issues in the field of the domain generation and discretisation, which affects the physical aspects of the testing phenomena, as well as several issues in the communication patterns which could affect a performance of the programmed routines and accuracy of the resulting data set. In this paper we have chosen a specific approach of coupling two different scales, macro- and micro- scale, in order to be able both to perform analysis on a rather large physical domain of porous media on the macro level without having an excessive data overhead or communication bottlenecks, so as to perform qualitative studies of a particular parameter within the micro model, where the number of computational unknowns goes up to several millions per m^3 of a generated soil sample. With the fact that the communication and computation parts are kept separated, so es both solvers for two different models, this approach can be adjusted to an arbitrary testing case which falls into the category of fluid flow under the pressure.

Keywords: high-performance computing, space-tree data structure, visualisation, porous media, fluid flow, macro scale, micro scale, multigrid like solver, Navier-Stokes model, Darcy model

1 Introduction

The idea of coupling two different scales has risen from the ever greater demand to explore and simulate the behavior of large volumetric areas of underground reservoirs in petroleum industry, as well as precipitation behavior in the agricultural fields, where a rather large testing samples are necessary in order that resulting data sets meet the

In Proc. of the Third Int. Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering, Civil-Comp Press, 2015 (DOI: 10.4203/ccp.107.48)

accuracy and the correct physical representation. Even with a reasonable coarse refinement of an adopted testing domain, which could represent such huge areas, an enormous amount of unknowns points with often several parameters of importance per one calculation points are generated, which quickly leads to the data overhead, far before the necessary precision is reached. Immersing into further below described concept, the main idea was not to neglect or diminish these drawbacks, rather to support such large scale simulations by providing calculated data on the micro level, at particular arbitrarily chosen points and pass them into the large scale model, where the necessary averaging procedures can be done at almost no memory and time cost.

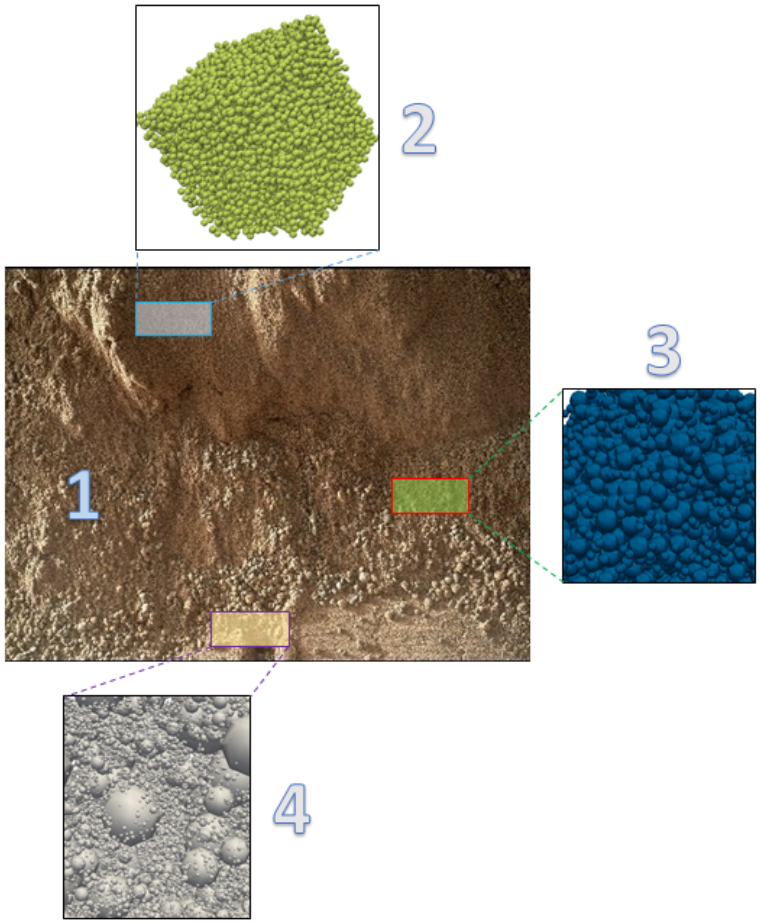

Figure 1: Concept Scheme - 1. Macro scaled model depiction with a single value of porosity throughout the domain. The sand grains' distribution may however vary from location to location, 2. Micro scaled model with a uniform size of the grains across the domain, 3. Micro scaled model with 3-4 grain sizes randomly distributed and 4. Micro scaled model with 10-12 different sizes of the grains

In Proc. of the Third Int. Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering,
Civil-Comp Press, 2015 (DOI: 10.4203/ccp.107.48)

Applying such approach, where the large scale model is treated as a homogeneous field, whereas the small scale model as heterogeneous field (see fig. 1) contributes with different fluid flow parameters such as velocity and pressure field, as well as permeability and porosity data values, which can be then furthermore utilized on the macro level. Having defined the concept pipeline, the focus of this research is set to the generation of a physical and numerical domain on the micro level, so as on the advance computing techniques that can be applied in order to efficiently produce well structured data set, suitable for the further analysis. The interpretation of generated data-set on the macro level is in the concurrence with a large scaled models defined by Darcy law, yet the exact further usability is out of the scope of this particular article. Interested users are referred to the published literature [10] and [7].

To follow the presented structure easily, in the section 2 the generation of the physical micro domain will be depicted, in the section 3 the generation of a numerical domain will be explained together with the data structure that supports highly efficient parallel calculations, preceding the section with depicted case setup and some first results. Finally, two last sections 5 and 6 before the concise conclusion are reserved for the brief analysis on a complexity of data set triggered by adaptive and uniform discretisation of a domain for ten different samples with different values of porosity and rather specific visualization techniques, which enable user to explore already calculated results in the real time.

2 Physical Domain Generation

In order to be as close as possible to the physical representation of a real sand sample, which is used in a regular experimental procedure of fluid flow through porous media, based on the Darcy Law, the granulometric sieve curve which describes the ratio of different sand grains' diameter sizes is used while generating an artificial physical domain.

With this data on disposal (fig. 2 - lower part) the total number of spheres of different size was calculated and randomly inserted into cubic shape of the domain. For the sake of the simplicity only spherical shape of the grain is treated, as otherwise numerous sphere intersection tests would be a mathematically demanding step.
The span of initial parameters that could be set and modified can be seen in the Figure 2 , which affect the shape and the size of the bounding box as well as the number of the spheres and their ratio within the testing volume. Amount of different sand grain distributions is theoretically not limited, although the larger the span is, the numerical discretisation more demanding appears to be. From the physical point of view, variety of different grain diameters is also directly correlated to the largest possible value of porosity, therefore less differ grain fractions can be densely packed into predefined volumetric space (see figures 1 and 5), whereas the presence of the large fractions leaves the inter-grain space empty increasing the porosity and amount of free space for the later simulated fluid flow. Unlike already known techniques of extraction of spheres from the bounded domain, in order to form a cheese-like structure which rep-

# Configuration File for the Random Sphere Generator 2.0				
Dimension of a sample - in one dimension	a			1
Ratio of the bounding box [x, y, z]	r	1	1	1
Starting point of a left bottom corner of the bounding box	xo	0	0	0
Porosity value	**p**			**0.50**
Amount of different grain distributions	**d**			**12**
Diameter of sand grain and its volumetric ratio in the total volume [mm - %]	300			20
	200			15
	100			10
	50			5
	25			2
	15			1
	10			0,1
	5			0,005
	2			0,001
	1			0,0005
	0.5			0,0001
	0.25			0,0001
	0.1			0,00005

Figure 2: Configuration File

resents the fluid flow domain, our approach is based on the generation of a space-tree data structure in so called voxelisation routine. The idea of this routine is to generate a volume-based model from the surface based model (usually triangular representation) using a ”volume pixel” or abbreviated ”voxel” as a primitive. Whether the voxel should be subdivided n-times giving in 3D space the set of new n^3 child voxels that should approximate a triangular continuous representation of a object, the decision is based on numerous geometric intersection tests. Once the volume-based model is generated, in order to define which part of it belongs to the geometry and which one to the cavity (later on treated as a fluid flow domain), a flood filling algorithm can be used. Namely, having had a given seed point in the fluid region, this algorithm checks all connected cells to the cell with this point and flag them as ”F” (fluid) cells. The detailed description of voxelisation process is given in [9].

3 Data Structure

As already briefly mentioned in the section 2, a domain voxelisation leads to the generation of the space-tree data structure, which starts with a logical root grid stored at the level zero that can be refined $r_x^r \times r_y^r \times r_z^r$ times, respectively in the three directions of the Cartesian coordinate system. Such refined logical grids can be furthermore refined $r_x^s \times r_y^s \times r_z^s$ times and this process is recursively repeated at every new level of depth, until the required physical precision is met (see figures 3 and 4). The refinement of a logical grid on the root level and all other logical grids is separated in order to account for the non-cubic shape of initial domain, which is sometimes and surely

In Proc. of the Third Int. Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering,
Civil-Comp Press, 2015 (DOI: 10.4203/ccp.107.48)

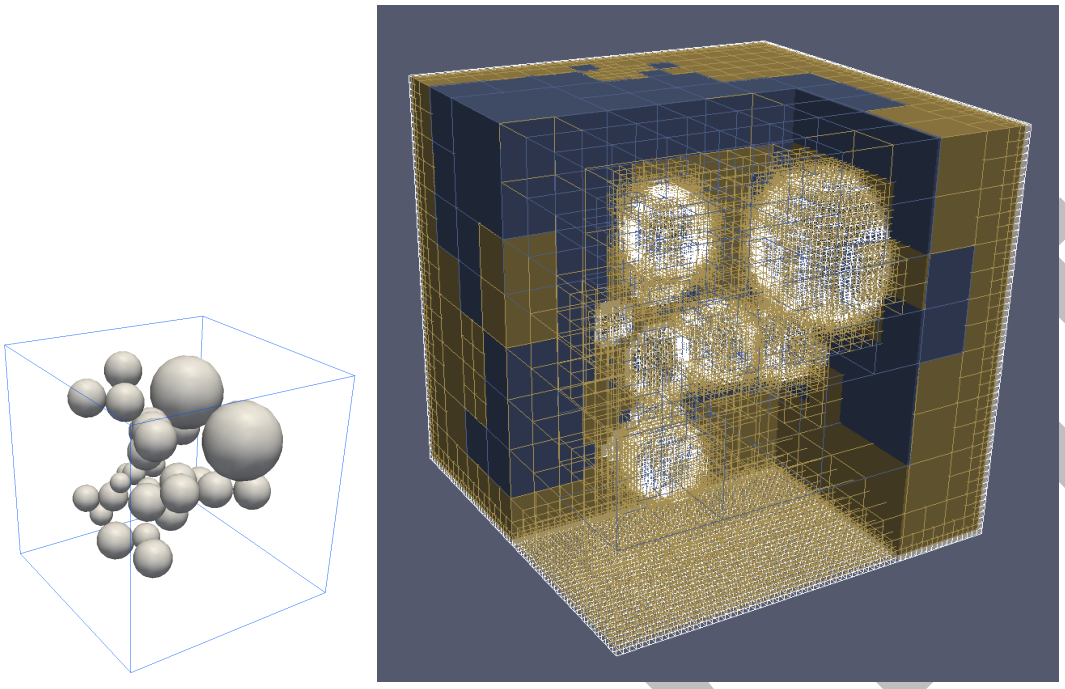

Figure 3: An artificially generated domain and its voxelisation

in this presented case necessary, in order to avoid the influence of applied boundary conditions onto the parameters that are supposed to be measured and analyzed. Furthermore, to every generated logical grid, that describes the topology of the domain, one data grid of a size $s_x^g \times s_y^g \times s_z^g$ is linked, containing all the variables necessary in the calculation and simulation process.

Once the data grids are generated, the communication among themselves is not the problem alone, unless the grids are assigned to the different computing processes during the load balancing. In order to enable a good synchronization of the data between two neighboring cells laying on the two different computing processes, every data grid is surrounded with one layer of so called $ghost$ or $helo$ cells, which are capable of storing the information about neighboring grids' calculated values at their common edge. Generated data grids itself are non-overlapping, block-structured, however, the padded $halo$ cells must overlap. That leaves the space for the implementation of Schwarz Methods [8], that is capable of solving the Laplace equation on two overlapping grids in a iterative process, by simply solving the equation on one grid, fixing these results as the boundary conditions before the equation is solved on the second grid. This process is repeated until a requested convergence requirement is met.
As the calculations on the whole domain must be performed only with the latest updated values, the data exchange and update must be done in a specific order. All types of the communication routines and their specific order of execution is thoughtfully described in the [6].

In Proc. of the Third Int. Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering,
Civil-Comp Press, 2015 (DOI: 10.4203/ccp.107.48)

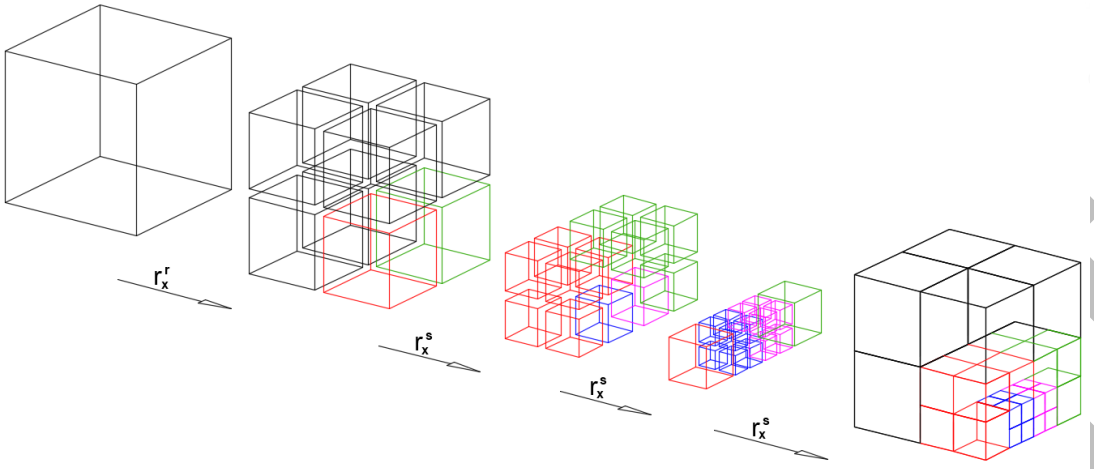

Figure 4: Octree scheme - Refinement of the root level is separately treated than the refinement of all other level of depths, in order to encounter a possible initial non-cubic shape of the domain

4 Case Setup and Data Results

4.1 Mathematical model

As previously introduced, two different mathematical approaches are integrated in the current work, offering a possibility to work with both refined and coarse data representation. Computational Fluid Dynamics (CFD) applied here is based merely on an incompressible, isothermal, Newtonian, single-phase flow described by three Navier-Stokes conservation of momentum equations:

$$\frac{\partial u_i}{\partial t} + \nabla \cdot (u_i \vec{u}) = \nabla \cdot (\nu \nabla u_i) - \frac{1}{\rho} \nabla \cdot (p \vec{e_i}) + b_i \tag{1}$$

for $i \in \{x, y, z\}$, where $\vec{u} = (u_x \; u_y \; u_z)^T$ denotes the velocity field in [m/s] in the three spatial dimensions x, y, z, the current time of the simulation is represented with t in [s], whereas ν is the kinematic viscosity in [m^2/s], ρ the density in [kg/m^3], p the pressure in [Pa], $\vec{e_i}$ the unity vector in the direction i and b_i sums up considered external volume forces such as acceleration due to gravity in [m/s^2].

The momentum equation is complemented with the conservation of mass equation, given in the form (under the assumption of the incompressible flow):

$$\nabla \cdot \vec{u} = 0 \tag{2}$$

The system of above stated equations is discretised according to the finite volume scheme in space and Adams-Bashforth second order explicit scheme in time. The solution is archived by applying the fractional step method also called projection

In Proc. of the Third Int. Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering,
Civil-Comp Press, 2015 (DOI: 10.4203/ccp.107.48)

method, proposed by Chorin [1], which basic idea is to deal with the Navier-Stokes momentum equations without external body forces and to treat its convective and diffusive part explicitly, whereas the pressure term is treated implicitly. As to realize this approach within one time step, the additional intermediate time step between n and $n+1$ was introduced, where the intermediate velocity field was denoted with u^*. Now, to benefit from the properties of the continuity equation (2), which enforces the divergence-free velocity field in the $n+1$ time step, a divergence operator is applied onto the equation in which the pressure is treated implicitly, which is, after a rather straightforward mathematical manipulation, transformed into the Poisson equation for the pressure. This equation can be explicitly solved for the pressure field in the next (n+1) time step, leading to the solution of the velocity field in the same time step and the subsequent time advancement, within which the repetition of the complete just described procedure per advancement is done, until the maximum calculation time is reached. More on this topic, as well as the mathematical notation of all aforementioned steps is given in [2], [3] and [6].

The second mathematical law integrated in this work is based on Darcy Law of fluid flow through porous media, contributing with permeability of the soil bed k in [m^2], having had previously calculated values of pressure and velocity. The Darcy Law is represented with following equation:

$$U_{darcy} = -\frac{k}{\mu} \cdot \frac{\Delta p}{L_i}, \tag{3}$$

where U_{darcy} denotes the Darcy flux in [m/s] in the three spatial dimensions x, y, z, the permeability k is given in [m^2], μ is the dynamic viscosity [kg/(m $\times$ s)], Δp stands for the relative pressure difference between two points laying on the distance L_i given in [m], according to the SI system.

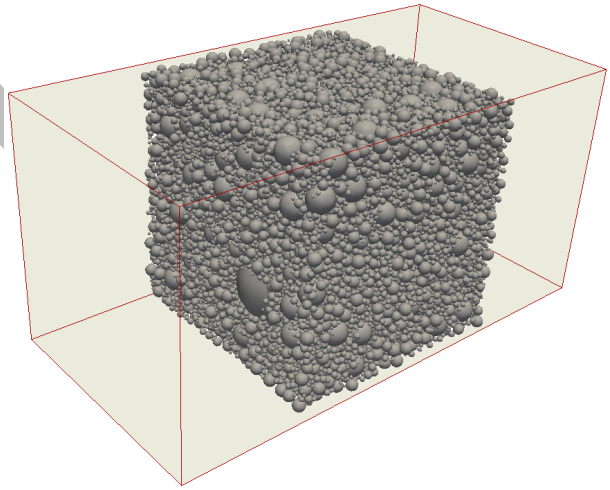

Figure 5: Setup of the sample

In Proc. of the Third Int. Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering, Civil-Comp Press, 2015 (DOI: 10.4203/ccp.107.48)

4.2 Setup of sample and results representation

So far described routines and models are implemented as a part of the framework and as such they can be applied on an arbitrarily chosen sample that corresponds to known assumptions and limitations. In order to check the efficiency of the code, a concrete case is presented in this section, whereas the next section is reserved for the sensitivity analyses with respect to this particular case of porous media. On the page 9 the first results are depicted, showing uniformly (see fig. 6) and adaptively generated mesh (see fig. 7). Numerical domain, that was used for the testing, has following geometrical characteristics:

- Size of the physical domain: $1m \times 1m \times 1m$ (narrow area fulfilled with the porous material - see figure 5),
- Size of the numerical domain: $2m \times 1m \times 1m$, with the physical domain centralized within it,
- Volume of solid particles compared to the total volume of the physical domain is set to 40%, which is defined with the fixed value of porosity $p = 60\%$. Short remark: densely packed samples are also thoughtfully tested, whereas this case is chosen for the representation for the sake of simplicity and visibility,
- Boundary conditions are set in the following order:
    1. Initial velocity field is set to $1m/s$ in order to avoid flow instabilities at the very begin of the calculation
    2. Inlet velocity follows the inflow condition, where all three components should be given. In this specific case $U_x = 0.5m/s$ and other two components are previously set to zero.
    3. Outlet velocity follows the outflow condition, having its derivative in the direction of the normal to the boundary equal to zero
    4. At the side walls and at the surface of sand grains fluid is at rest with both tangential and normal component equal to zero and finally
    5. The pressure value is explicitly defined at the outlet east face of the domain.

The resulting magnitude of the velocity field in a single point on the lowest level of depth, calculated in this case on the adaptively generated mesh shows the discrepancy to the values on the uniformly discetised domain of barely 1%. The calculation is, on the other hand, conducted on the roughly 1.5×10^6 to almost 5×10^6 cells less (2-5 times smaller data set, depending on the geometry file) which provided up to 5x speedup over the original calculation on the uniform domain. The rest of the analyses for the different setup cases are still in progress in the time of writing of this article.

In Proc. of the Third Int. Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering, Civil-Comp Press, 2015 (DOI: 10.4203/ccp.107.48)

5 Sensitivity analyses on adaptive refinement approach

In order to get a better idea, how the type of mesh refinement affects the total number of unknowns within geometrically complex domain such as a porous media sample, the sensitivity analyses are conducted with respect to the porosity value, which essentially change the ratio of fluid to the total number of cells. For the purpose of simplicity, all ten calculated cases have the same grain distribution, accordingly scaled with a amount of available space in which they should are inserted. Among all other parameters that are kept constant, are:

- Initial refinement at the zero level of the depth (root): $r_x^r \times r_y^r \times r_z^r = 2 \times 1 \times 1$
- Refinement ratio at all other level of depths: $r_x^s \times r_y^s \times r_z^s = 2 \times 2 \times 2$
- Level of depth of voxelisation $d = 5$
- Size of the numerical domain: $x \times y \times z = 2m \times 1m \times 1m$
- All boundary conditions, listed in the section 4 remained constant during the testing time.

In the figure 8 the expected difference in the total amount of cell in discretized domain is shown, obtained by applying two different methods of refinement, where theoretically for the very low values of porosity the total number of cells in adaptive scheme should tend to the total number of cells in the uniform scheme. Time efficiency is depicted in the figure 9.

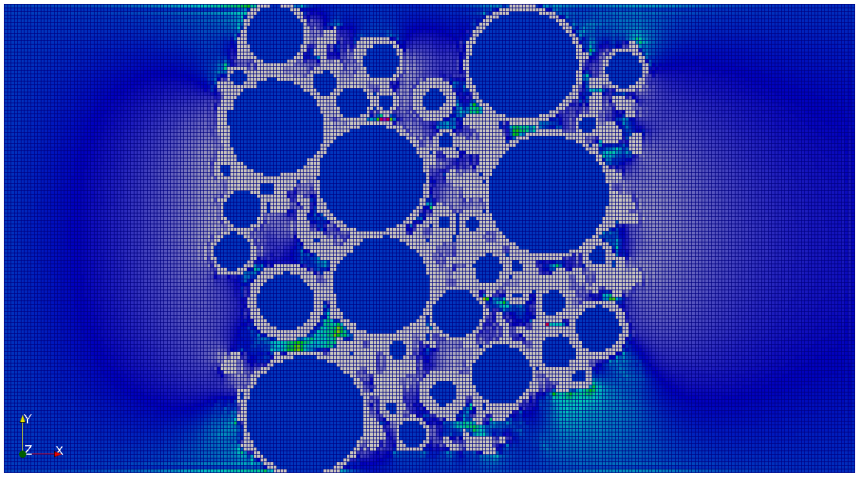

Figure 6: Setup Test Case - Uniform mesh refinement

In Proc. of the Third Int. Conf. on Parallel, Distributed, Grid and Cloud Computing for Engineering, Civil-Comp Press, 2015 (DOI: 10.4203/ccp.107.48)

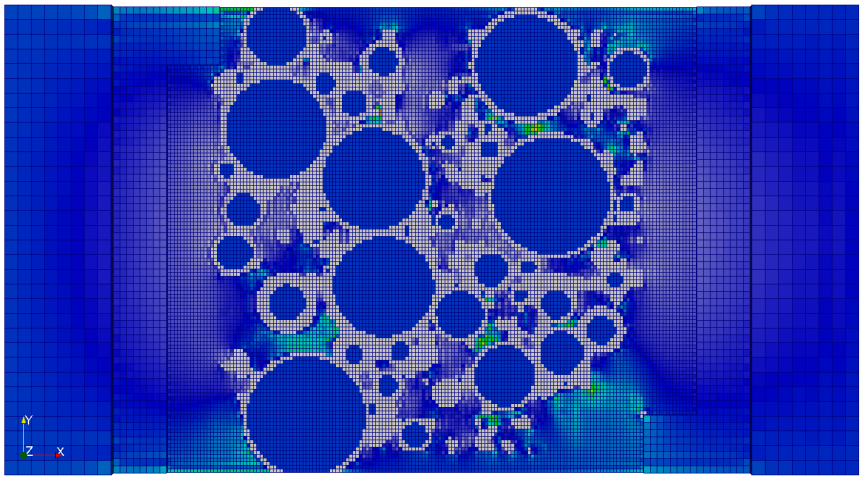

Figure 7: Setup Test Case - Adaptive mesh refinement

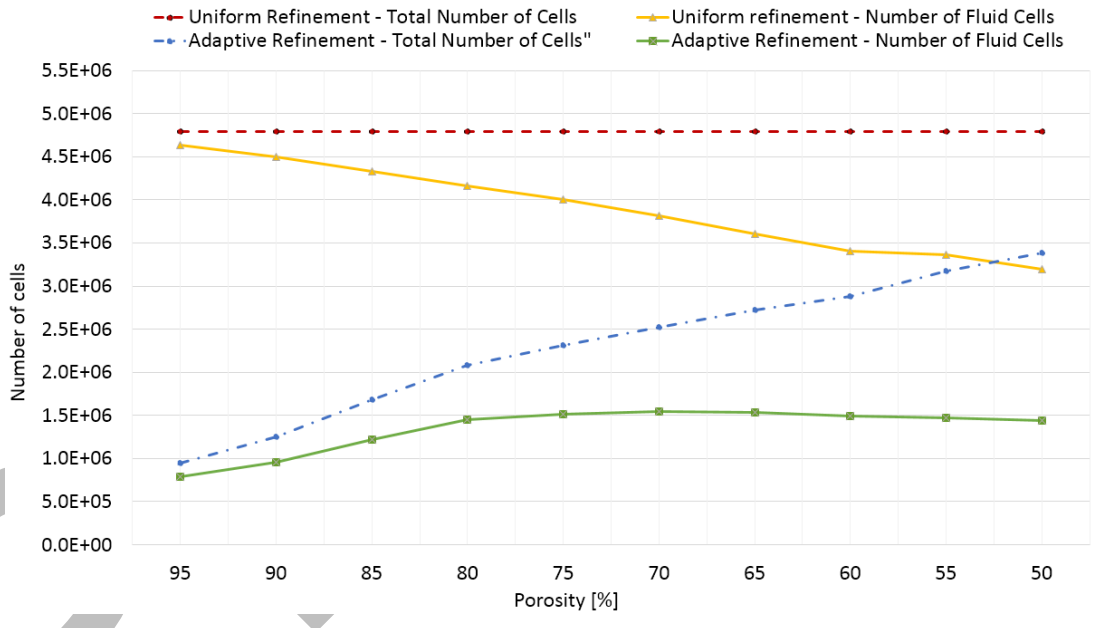

Figure 8: Ratio of total number of cells and fluid cells against the porosity value for two different cases - Uniform and Adaptive Refinement

6 Data Exploration and Visualisation

An integral part of this high-performance parallel code is a so called sliding window or the tool for the interactive exploration and visualization of computed data results in the real time. Bearing in mind that such calculations on a highly complex numeri-

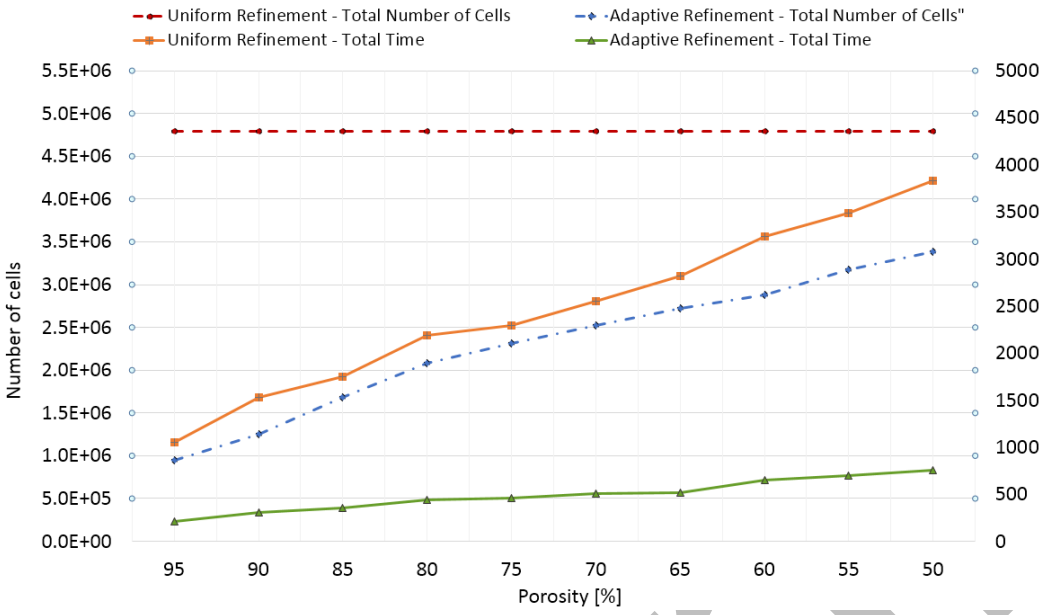

Figure 9: Ratio of total number of cells and total execution time against the porosity value for two different cases - Uniform and Adaptive Refinement

cal domains with a no less than hundreds of thousands of data points multiplied with almost always several unknowns per a point could take long time until they are done. During this time, user would not be able to check either results, or trends of the calculation, without disturbing the occupied calculation processes and therefore influencing the performance rate. Knowing the calculation results before the complete process is done, can lead user to the decision whether such calculation might be stopped and run again without lost of additional CPU time, energy and memory, concerning the nature of super-computer architecture, on which the code is executed. Having had the already implemented complex structure of grids in the form of the space-tree and the communication routines, which are responsible for the update of the data among the grids, instructed by the neighborhood-server (more details on this topic are published under [4] and [5]), the implementation of the sliding window was done essentially in few separate steps (see figure 10). The complete process of obtaining the calculated data from the calculation processes is done over neighborhood server (NBH), which is implemented as a management node, knowing which process is responsible for the requested amount of data from the user's side. The NBH server itself is in charge of checking the grids, where the data-set is calculated and stored, applying this procedure from the root to the next finer level consecutively.

This order of checking is important, as a user has the opportunity to define the size of the domain, which should be visualized as well as the amount of points with which the data should be represented. As soon as this maximum number is reached, the NHB server stops further checking and all finer grids from that point on are neglected. The smaller amount of points is visualized, the coarser the representation is, giving a user

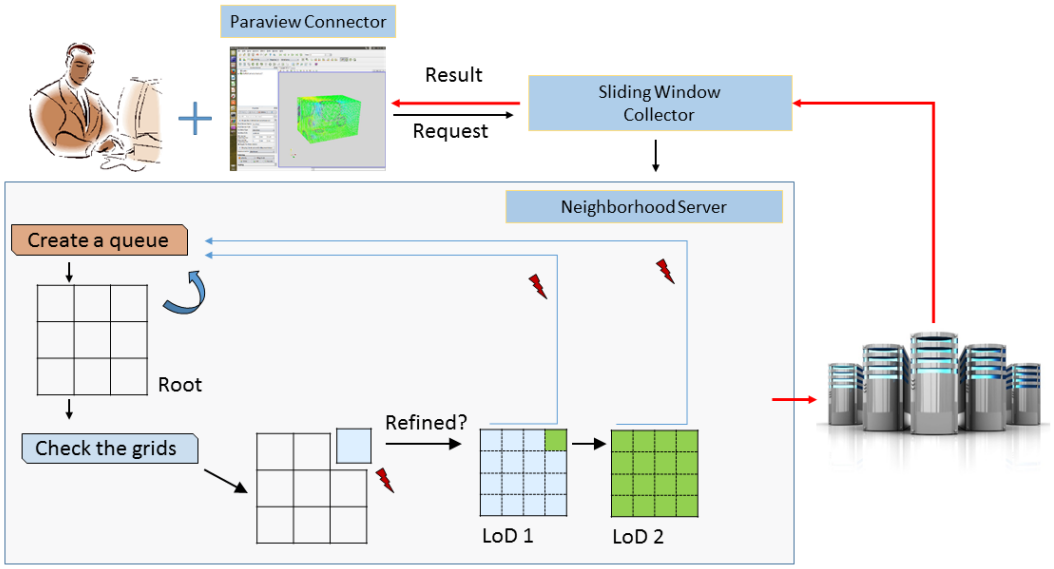

Figure 10: Sliding Window Concept - Interactive real-time exploration of calculated data in five steps

just a general overview of calculated data. On the other hand, the more the points are requested, the higher the disturbance is and the interactivity will be brought into question.

In the figure 10 it can been seen, that the selection of the grids is completely done under the control of NBH, leaving the computation processes intact, until the end of the procedure, when they are instructed to send the information to the sliding window collector using fast interconnect. This information will be then passed to the user at slow bandwidth. Finally the sliding window name comes from the fact that user has a possibility, within the paraview visualisation interface, not just to define the size of the sub-domain that should be visualized, but also to change its position within the complete domain by ”sliding” it interactively in every new time step.

7 Conclusion

This paper presented basic concepts and data structures of the finite volume code developed for the parallel high-performance computing in the field of computational fluid dynamics. After its thorough testing using several different benchmarks, here we have gone one step further and the testing was done using a geometrically complex domain of porous media, which includes coupling of two different scales, rather than staying up to one scale requiring always more computational resources and computational time, as the complexity and size of the physical domain increases. With multi-scale approach a wide range of quantitative and qualitative analyses is enabled, avoiding data overhead and various communication bottlenecks. The presented sensitiv-

ity analyses showed an obvious advantage of adaptive mesh refinement over uniform mesh generation of the complex domain, although testing of this issue is currently still ongoing and more general conclusion are to come. Furthermore, the different solvers are independently implemented, and communication and calculation parts are also separated, giving the code a huge potential to be adjusted to an arbitrarily defined setup within the concept of fluid flows under the pressure, as well as to be upgraded with different solvers and used in wide variety of fluid flow scenarii. This work is currently in progress.

Acknowledgment: This publication is partially based on work supported by Award No. UK-c0020, made by King Abdullah University of Science and Technology (KAUST)

References

[1] A.J. Chorin, "Numerical Solution of the Navier-Stokes Equations", Mathematics of Computation, pg. 745-762, 1968.

[2] Ferziger and Perić, "Computational Methods for Fluid Dynamics", Springer, 3rd Edition, 2002.

[3] C. Hirsch, "Numerical Computation of Internal and External Flows, Volume 1", Butterworth–Heinemann, 2nd Edition, 2007.

[4] R.-P. Mundani and J. Frisch and E. Rank, "Towards Interactive HPC: Sliding Window Data Transfer", Proc. of the 3rd Int. Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering, Civil-Comp Press, 2013.

[5] J.Frisch, R.-P.Mundani, E. Rank, "Resolving neighbourhood relations in a parallel fluid dynamic solver", in Proc. of the 11th International Symposium on Parallel and Distributed Computing, Los Alamitos, CA, USA, 2012, pp. 267-273, IEEE Computer Society.

[6] Frisch, Jérôme, "Towards Massive Parallel Fluid Flow Simulations in Computational Engineering", PhD Thesis, Technische Universität München, 2014.

[7] G.K. Batchelor, "An Introduction to Fluid Dynamics", Cambridge University Press, 2000.

[8] H. A. Schwarz, "Ueber einen Grenzübergang durch alternirendes Verfahren", Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, Vol.15, pp.272–286, 1870.

[9] Ralf-Peter Mundani, "Hierarchische Geometriemodelle zur Einbettung verteilter Simulationsaufgaben", PhD Thesis, Universität Stuttgart, 2006.

[10] United States Environmental Protection Agency EPA, "Method 9100 Saturated hydraulic conductivity, saturated leachate conductivity, and intrinsic permeability", 1986.