Parallel Simulation and Visualization of Blood Flow in Intracranial Aneurysms

Wolfgang Fenz, Johannes Dirnberger,
Research Unit Medical Informatics
RISC Software GmbH
Hagenberg, Austria
{*wolfgang.fenz* | *johannes.dirnberger*}*@risc.jku.at*

Christoph Watzl and Michael Krieger
Austrian Grid Development Center
RISC Software GmbH
Hagenberg, Austria
{*christoph.watzl* | *michael.krieger*}*@risc-software.at*

***Abstract*—Our aim is to develop a physically correct simulation of blood flow through intracranial aneurysms. It shall provide means to estimate rupture risks by calculating the distribution of pressure and shear stresses in an intracranial aneurysm, in order to support the planning of clinical interventions. Due to the time-critical nature of the application, we are forced to use the most efficient state-of-the-art numerical methods and technologies together with high performance computing (HPC) infrastructures. The Navier-Stokes equations for the blood flow are discretized via the finite element method (FEM), and the resulting linear equation systems are handled by an algebraic multigrid (AMG) solver. First comparisons of our simulation results with commercial CFD (computational fluid dynamics) software already show good medical relevance for diagnostic decision support. Another challenge is the visualization of our simulation results at acceptable interaction response rates. Physicians require quick and highly interactive visualization of velocity, pressure and stress to be able to assess the rupture risk of an individual vessel morphology. To meet these demands, parallel visualization techniques and high performance computing resources are utilized. In order to provide physicians with access to remote HPC resources which are not available at every hospital, computing infrastructure of the Austrian Grid is utilized for simulation and visualization.**

***Keywords*-Computational Fluid Dynamics; Parallel PDE solver; Parallel Visualization**

I. Introduction

The various forms of vascular diseases are complex in their origins and their manifestations. In order to diagnose, prevent and treat such diseases, a detailed knowledge of the blood flow is essential. Vessels respond to stimuli and adapt to changes in blood flow and blood pressure. Hemodynamic (blood flow mechanic) factors are strongly correlated with the localization of atherosclerotic plaque. Atherosclerosis is primarily responsible for strokes, heart attacks and aneurysms, which are among the most frequent causes of death in western countries. In many cases the time of reaction is an important factor in the treatment of these diseases and is crucial for the convalescence of the patient. Due to atherosclerosis it is often necessary to dilate vessels to improve the blood flow. Stents are used to support the vessel wall after dilation. We see a major application area in the simulation of blood flow in detected aneurysms and stent supported vessels. Primarily, it is important to get a deeper understanding of how blood flow actually affects the vessel walls and study the effect on aneurysm growth behavior based on simulation. Every patient's physiology forms an individual flow pattern with dedicated locations of higher risk for ruptures of the blood vessel wall. It is important to know where these high risk areas are, in order to decide on an appropriate treatment. Moreover, the decision of the physician, which stent to use and where to place it, can be supported by the resulting flow patterns. As a future outlook, effects of different stent designs can also be observed in silico. These results may help to develop better flow diverting properties of stents. A simulation of the hemodynamics in a 3D geometry based on medical imaging should focus on the following:

- An easy to handle tool for physicians to reconstruct the blood volume and to segment the region of interest.
- Correct modeling of the vascular system (blood lumen and vessel wall) in 3D.
- Fast and robust calculation of the hemodynamics in the relevant area.
- Visualization of the resulting flow patterns in a way that it can support the physician's decision without being an expert in mathematics and/or physics.

A decision support tool for the physician can, therefore, play a major role for recovery. To assure practical relevance, a close participation of clinical partners during the whole project is important.

Apart from the detailed and accurate simulation of the blood flow, the responsiveness of the application also plays an important role. The interactive process of the application for simulating the blood flow can be divided into different schemes, meaning: Calculation intensive parts of the application require batch processing, whereas other parts of the application require highly responsive direct interaction with data sets which may be too big for interactive processing on common work stations in clinical environments. Therefore, we are applying the concept of Grid computing [1] to this application and enable the usage of shared resources in a virtual organization for performing computing and data intensive tasks. Additionally, the Grid middleware provides secure data transfer between resources and secure access to

resources in a controlled environment and between trusted parties, which is of utmost importance in clinical environments. Additionally, as the application and the workflow are designed to work on shared capacity and capability computing resources, new ways of collaboration between different clinical teams and the shared usage of expensive hardware resources will be possible.

II. Application Workflow

To provide quick simulation results and a responsive user interface for visualizing the results we suggest distributing the application workflow to different resources meeting the performance requirements for clinical practice. Considerable computational resources may be necessary to achieve the required response time requested by physicians, whereas in clinical practice the processing power and memory of local work stations are limited. We approach this problem by providing access to remote high performance computing resources from local work stations with the help of Grid middleware.

The workflow of the simulation system basically consists of the following steps (see also Fig. 1), which can be split between resources:

- 3D reconstruction of medical images and extraction of blood and vessel geometry data (mesh generation).
- Computation of a solution for the simulation model on the generated mesh.
- Interactive visualization of the simulation results.

Only the first step, the finite element mesh generation from a 3D volume, is performed on a local workstation. After this step, mesh data is transferred via Grid File Transfer Protocol (GridFTP) to Grid resources to perform the blood flow simulation with the help of a parallel algebraic multigrid (AMG) solver. This work builds on and continues research done on Grid-enabled AMG solvers and fluid-structure interaction in Grid computing environments by different research groups in the Austrian Grid project [2] [3].

One approach for the interactive visualization is to render the data locally. Due to the high resolution requirements [4] of our simulation the visualization on a local workstation as well as the data transfer of the result set between clusters and workstations may exceed the possibilities of clinical standard environments. To avoid the transfer of large result sets (which can amount to gigabytes of data for fine meshes), in an ideal environment the visualization should be computed as close as possible to where the result data resides. Therefore, the optimal setup of the computing resources used for simulation also includes an application for parallel visualization, in our case *ParaView* [5]. We enable the visualization client to connect to the visualization server via port-forwarding with the help of the interactive Grid shell access tool *glogin* [6], thus allowing users to perform interactive visualizations in Grid environments.

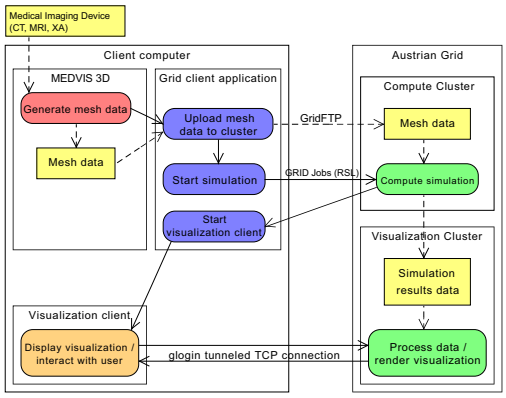


Figure 1. Application workflow

III. Methods

Our main development goal is the implementation of a realistic and physically correct simulation model of the blood flow through the reconstructed blood vessel geometry. The geometry information gained by extracting the isosurface of the blood volume is put into a computer simulation together with blood and tissue parameters, building up a virtual model of the patient's vascular system. Then the simulation procedure initiates and calculates pressure and velocity fields of the whole system at discrete time steps. The results have to be visualized in 3D, in order to quickly provide valuable diagnostic information for the assessment of the aneurysm's rupture risk. The Grid computing approach allows access to high performance computing resources leading to shorter response times for simulation results and faster visualization of large datasets.

IV. Mesh Generation

Commonly used methods in medical imaging are magnetic resonance imaging (MRI), X-ray angiography and computed tomography (CT). All these methods provide series of cross-section pictures (slices) of the patient. Our system can reconstruct the 3D volume from these cross sections and subsequently generate a finite element mesh of a selected region (volume of interest). For surface meshing, the adaptive skeleton climbing algorithm [7] is used, which is superior to commonly used marching cube schemes, whereas for the creation of tetrahedra for the blood lumen we have integrated both the NETGEN [8] and TETGEN [9] libraries. The quality of both meshes is further improved by applying area weighted Laplacian smoothing [10], [11].

V. Blood Flow Simulation

We model the blood in the vessel as an incompressible Newtonian fluid with density $\rho = 1.05$ g/cm^3 and viscosity $\mu = 3.85$ cP, described by the Navier-Stokes and mass continuity equations,

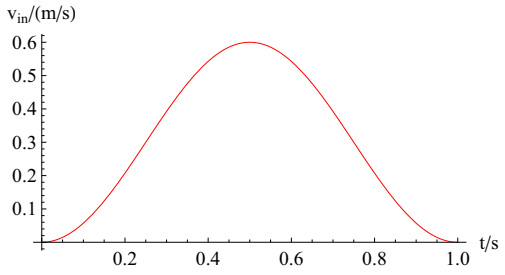


Figure 2. Variation of the peak inflow velocity with time.

$$\begin{aligned} \rho\left[\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\nabla)\mathbf{v}\right] - \mu\boldsymbol{\Delta}\mathbf{v} + \nabla p &= 0, \\ \nabla\mathbf{v} &= 0. \end{aligned} \tag{1}$$

We calculate the time-dependent velocity $\mathbf{v}(\mathbf{r}, t)$ and pressure field $p(\mathbf{r}, t)$ in the whole three-dimensional domain, taking into account boundary conditions specified on the vessel walls and inlet and outlet areas. In particular, we use a no-slip condition for the walls and model the velocity variation of the incoming blood stream during one pulse cycle (1s) with a sine function,

$$v_{in}(t) = v_0\left[1 + \sin(2\pi t - \pi/2)\right], \tag{2}$$

where $v_0 = 30$ cm/s (see Fig. 2). This value is multiplied with a parabolic profile fitted to the cross-section of the vessel at the inlet. At the outlet(s) we prescribe a constant pressure $p = 0$.

A. Numerical Solution

The numerical solution of the Navier-Stokes equations is obtained by applying the Galerkin finite element method [2], which provides a discretization of the corresponding fields on arbitrarily shaped domains, and replaces the partial differential equations with large systems of linear equations of the general form

$$Kx = b. \tag{3}$$

Here, the solution vector x contains the values of the field at the nodes of the FEM mesh, and K is the system matrix, built up from the individual element matrices. Such equations have to be solved by iterative methods, most notably Krylov subspace methods such as conjugate gradient (CG), which requires the coefficient matrix K to be symmetric and positive definite (SPD). The use of a preconditioner, which transforms Eq. (3) into an equivalent system that is easier to solve, can speed up the calculation considerably. The optimal method in terms of arithmetic and memory complexity is algebraic multigrid [12]. In order to apply CG with AMG as preconditioner, the Navier-Stokes equations (1), which yield a non-symmetric system matrix upon discretization, have to be split up in velocity and pressure parts. We are using the so-called characteristic-based split technique [13], [14]. Instead of the coupled equations (1), three separate equations have to be solved successively: one for an intermediate velocity, one for the pressure, and one for the final velocity, corrected via the pressure obtained in the preceding step. Since both velocity equations are fully explicit, only the pressure equation, which is of Laplace type and therefore yields an SPD matrix, needs to be solved iteratively.

Once the velocity field is known, we can calculate derived quantities such as the wall shear stress (WSS) τ_w. It is obtained from the velocity gradient tensor $\nabla\mathbf{v}$,

$$\nabla\mathbf{v} = \begin{pmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} & \frac{\partial v_x}{\partial z} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} & \frac{\partial v_y}{\partial z} \\ \frac{\partial v_z}{\partial x} & \frac{\partial v_z}{\partial y} & \frac{\partial v_z}{\partial z} \end{pmatrix}, \tag{4}$$

and the normal vector at the surface $\mathbf{n}_w$ as

$$\tau_w = \mu \left\| \nabla\mathbf{v} \cdot \mathbf{n}_w \right\|. \tag{5}$$

B. Parallelization

Each simulation step consists of two separate computational tasks that can be accelerated by parallelization, but with different requirements: On the one hand the calculation of the element matrices for pressure and velocity equations, and on the other hand the solution of the linear equation system. While the former can be split into completely independent subproblems by distributing the elements between the CPUs, the latter requires communication between the processes due to the fact that adjacent elements share common nodes with each other. Therefore, we use different approaches in the two cases. For the matrix calculation we use a simple OpenMP `for`-loop over the total number of elements that is processed by n CPUs in parallel. For the linear solver, however, we apply the parallel AMG toolbox by M. Liebmann [15], which uses MPI commands for data exchange between the processors. The solver is invoked via an external system call from our CFD code at each time step, the equation data (element matrices, right hand side vector, initial values for the solution) are passed via binary files. Parallelization affects the CG/AMG solver itself as well as the input of the element matrices and the accumulation of the global system matrix. Also, a file specifying the partitioning of the mesh and the assignment of each element to a CPU has to be provided. We use METIS [16] to generate these partitions.

C. Visualization

In contrast to the numerical results of the vascular flow simulation, the presentation of these results in a medical application should make the images ready for interpretation and verification tasks. Therefore, the intuitive graphical visualization of flow patterns, pressures and stresses within the reconstructed 3D structures of the patient's morphology plays an important role for the acceptance and deployment of computational simulation systems in medical environments.

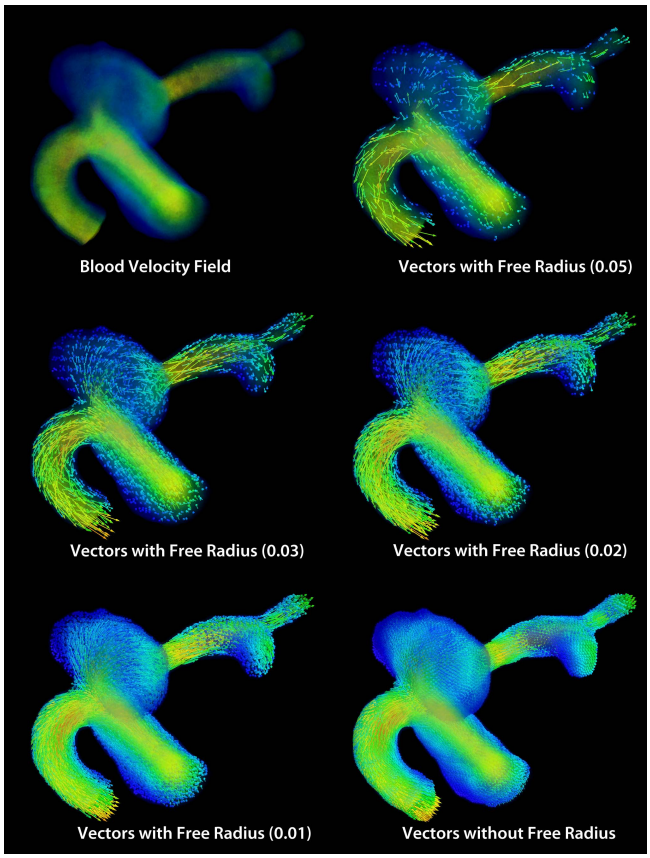


Figure 3. Visualization of blood velocities using our free radius method with different discretization.

Together with our medical partners we develop a blood flow visualization which allows physicians to easily explore a patient's vascular system and to locate parts of enhanced risk. Displaying results with acceptable interaction rates requires fast and flexible strategies in the field of computer graphics, and we have to extensively exploit the capabilities of modern graphics hardware as well as parallel rendering techniques.

1) Pressure/Stress Field: The scalar field of blood pressure calculated by our FEM simulation is visualized as a semi-transparent cloud. Therefore, we have to transform the tetrahedra that form the finite element mesh into a consistent color field based on an RGB rainbow palette without sharp edges. High pressures are visualized in red with high opacity, whereas low pressures are colored blue with low opacity. We use the Projected Tetra algorithm [17] as volume rendering technique.

2) Velocity Field: The 3D vector field representing the blood velocity at every mesh point is visualized as 3D arrows with lighting and shading effects. Larger velocities result in large red arrows, whereas lower velocities are represented as small blue arrows. Since blood velocity is low inside of aneurysms in comparison to the velocity in the vessel, we had to find a trade-off in order to visualize these different velocity domains. ParaView allows choosing between different display settings for vector glyphs that influence the vector's shape. Furthermore, a maximum of displayed glyphs can be defined, which are chosen randomly. These customizations are not sufficient though for our medical application. Thus, we have developed our own visualization module that allows the definition of a free radius, within which only one vector will be displayed. In order to maintain visualization performance, we discretize the velocity field according to the chosen radius and check for the presence of multiple vectors within discrete cubes. Only one vector per cube is rendered, then the cube is flagged as being already visited. By doing so, all locations of the velocity field will be represented adequately without oversampling. We have decided to visualize not only the vector field, but also the scalar field of velocity magnitudes in order to give a quick overview on fast flow regions within the vessel geometry (see Figure 3).

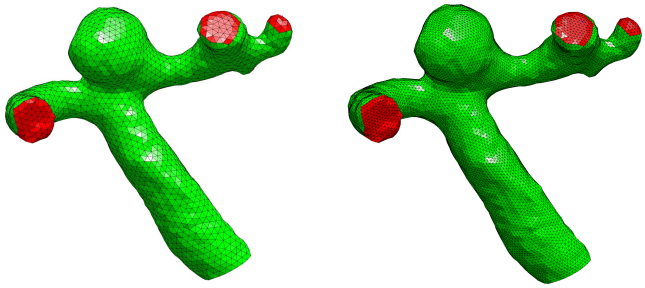
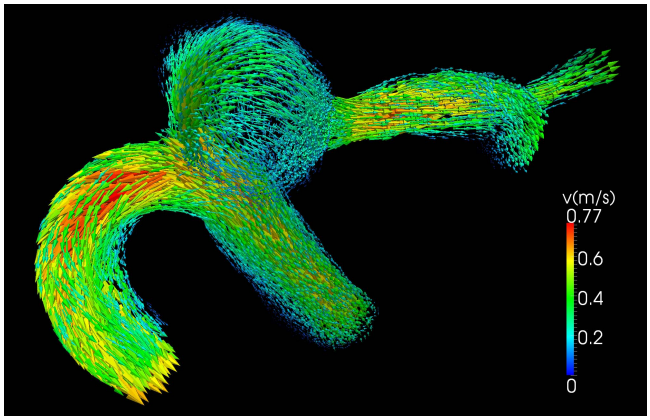


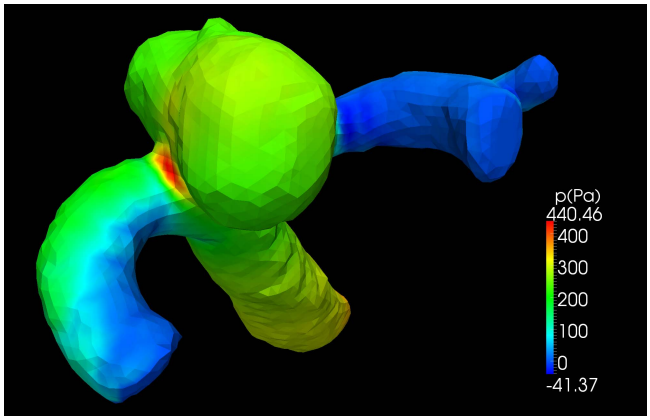
Figure 4. Two FEM meshes used in the simulations (mesh I and II). The red areas are outlet boundaries. (visualized with NETGEN)

VI. Results

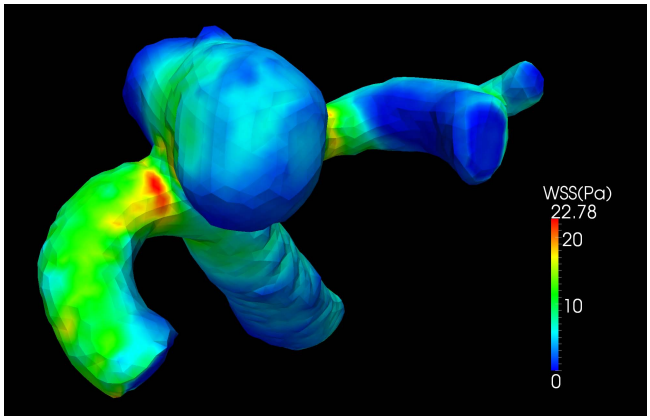
For this paper we performed simulations on three different meshes generated from the same geometry: one with low resolution (Mesh I: 6791 nodes, 30838 elements, 1 MB), one with moderate resolution (Mesh II: 47357 nodes, 246704 elements, 10 MB) and one with high resolution (Mesh III: 353169 nodes, 1973632 elements, 100 MB), where the latter two were built from the first one by mesh refinement [18] (see Fig. 4). The original 2D image data were obtained from a cerebral aneurysm measured on a human patient via X-ray angiography. A full pulse cycle (1s) was calculated for meshes I and II, using a time step size of $\Delta t = 10^{-4}$ s. Results for the finer one are visualized in Fig. 5, showing both the velocity and the pressure fields at the maximum value of the inflow velocity ($t = 0.5$ s), as well as the wall shear stress on the surface of the vessel. It is clearly visible that the peak value of the pressure occurs at the neck of the aneurysm, where it is twice as large as on the surrounding areas [Fig. 5(b)]. The wall shear stress also shows elevated values in the vicinity of the neck [Fig. 5(c)]. We verified our simulation results with existing commercial



(a)



(b)



(c)

Figure 5. Simulation results for the blood flow in the measured aneurysm geometry at peak inflow velocity ($t = 0.5$ s). (a) shows the velocity field, (b) the pressure and (c) the wall shear stress (WSS) on the vessel surface. The elevated pressure and WSS near the neck of the aneurysm are clearly visible. (visualized with ParaView)

simulation systems (FLUENT) in order to check our code for correctness.

In Table I we analyze the calculation times needed for one typical simulation step on the three different meshes.

(a) Mesh I: 6791 nodes, 30838 elements, 1 MB

CPUs	M	CG	CGS	AMG	AMGS	TOT
Desktop (1/1)	0.54	0.33	0.12	0.24	0.03	0.78
Desktop (4/1)	0.15	0.33	0.12	0.24	0.03	0.39
Desktop (4/4)	0.15	0.23	0.16	0.17	0.07	0.32
Cluster (1/1)	0.13	0.11	0.047	0.09	0.02	0.27
Cluster (1/2)	0.13	0.06	0.032	0.06	0.012	0.19
Cluster (1/4)	0.13	0.05	0.026	0.04	0.008	0.17
Cluster (1/8)	0.13	0.05	0.023	0.035	0.006	0.165
Cluster (1/16)	0.13	0.17	0.05	0.13	0.022	0.26

(b) Mesh II: 47357 nodes, 246704 elements, 10 MB

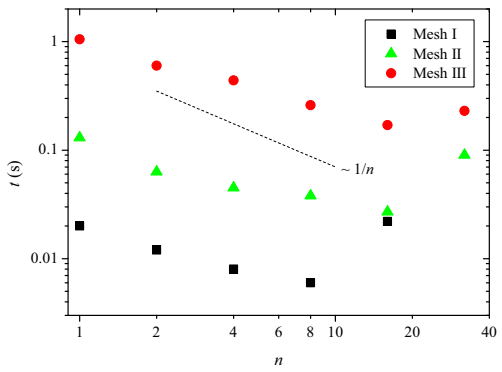
CPUs	M	CG	CGS	AMG	AMGS	TOT
Desktop (1/1)	4.2	3.5	2.6	1.3	0.23	5.5
Desktop (4/1)	1.2	3.5	2.6	1.3	0.23	2.5
Desktop (4/4)	1.2	2.5	2.1	0.75	0.19	1.95
Cluster (1/1)	1.1	1.26	0.78	0.67	0.13	1.77
Cluster (1/2)	1.1	0.75	0.51	0.34	0.063	1.44
Cluster (1/4)	1.1	0.45	0.30	0.22	0.045	1.32
Cluster (1/8)	1.1	0.33	0.21	0.16	0.038	1.26
Cluster (1/16)	1.1	0.25	0.14	0.14	0.027	1.24
Cluster (1/32)	1.1	0.22	0.11	0.25	0.09	1.32

(c) Mesh III: 353169 nodes, 1973632 elements, 100 MB

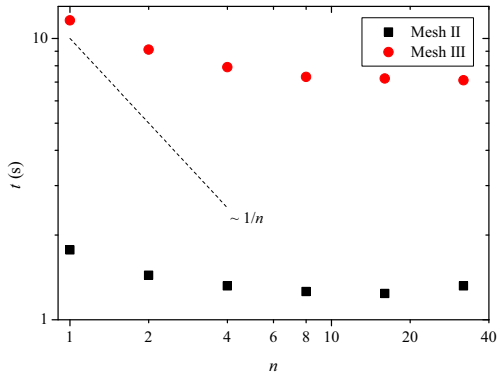
CPUs	M	CG	CGS	AMG	AMGS	TOT
Desktop (1/1)	14	34	23.5	10.2	2.2	24.2
Desktop (4/1)	8	34	23.5	10.2	2.2	18.2
Desktop (4/4)	8	28.4	25	6.1	1.8	14.1
Cluster (1/1)	6.13	18.6	14.6	5.5	1.05	11.6
Cluster (1/2)	6.13	11.5	9.4	3.0	0.60	9.13
Cluster (1/4)	6.13	7.2	6.1	1.77	0.44	7.9
Cluster (1/8)	6.13	5.7	4.3	1.13	0.26	7.3
Cluster (1/16)	6.13	3.3	2.7	1.07	0.17	7.2
Cluster (1/32)	6.13	2.7	1.85	0.93	0.23	7.1

Table I
BREAKDOWN OF CALCULATION TIMES NEEDED FOR ONE TIME STEP OF THE SIMULATION WITH DIFFERENT CONFIGURATIONS OF CPUS AND SOLVERS FOR MESHES I (A), II (B) AND III (C) (TIMES IN S).

Calculations were performed on a standard quad-core desktop (Intel Core2 Quad Q6700 with Windows Vista x64) as well as a cluster in the Austrian Grid (Altix ICE 8200, with 384 CPU cores running Linux). For different CPU configurations (the first number gives the number of CPUs used for the matrix calculation, the second one for the linear solver), we list the times spent for element matrix calculation (column M), the pure solution time of the linear system with and without preconditioning (columns CGS/AMGS) and the total time spent in the linear solver (including file IO and system matrix assembly; columns CG/AMG). The shortest total calculation time per time step for each configuration of CPUs (usually the one using the AMG solver; column TOT)



(a)



(b)

Figure 6. Log-log plot of the calculation time for one simulation time step as a function of the number n of CPUs on the cluster available in the Austrian Grid. (a) running time of the AMG solver, (b) total time for one time step. The slope of the dashed line corresponds to a linear scaling law.

is also included.

Several conclusions can be drawn from these results. Comparing the performance of the two solvers, it is obvious that while the AMG solver is faster than the pure CG for all meshes, the performance gain increases significantly with refining the mesh, rising from a factor of 1.4 to 3.3 for the desktop system, and similarly on the cluster of the Austrian Grid we used. When looking at the time spent in the solver itself, the effect is even more drastic, going from a factor of 4 to 11. For coarser meshes, the advantage of AMG is predominated by the overhead due to IO operations, matrix assembly and setup of the AMG itself (only 12.5% of the time spent in the AMG code is dedicated to the solution of the system, as opposed to 36% in the case of CG). There is still room for improvement in this regard, since currently we call the AMG solver at every time step, and therefore the buildup of the system matrix and AMG setup are repeated each time.

Another interesting point is the effect of parallelization for the different platforms and meshes. While both solvers scale almost linearly with the number of CPUs for mesh II and mesh III on the cluster of the Austrian Grid, the performance gain for the small mesh is much lower (roughly 1.5 for 4 CPUs). On the desktop, the speedup is generally much less (1.4 for CG with 4 cores of a desktop quad-core CPU on meshes I and II, 1.2 for mesh III), and for the smallest mesh the pure solver times even increase when using more than one CPU since the time spent for inter-process communication is larger than the gain from the parallel calculation.

As for the OpenMP parallelization of the matrix calculation, it yields an almost linear performance gain on the desktop, while for the cluster machine no significant effect was observed. Since the calculation time without parallelization is already smaller than the time spent using 4 desktop CPU cores, we ascribe this to automatic compiler optimization and do not include the OpenMP results for the Grid in the tables.

All in all, a speedup factor of roughly 4.5 could be achieved by using HPC cluster hardware and parallelization (5.5s for one time step on mesh II with one core of the desktop CPU compared to 1.24s with 16 cluster CPUs). When looking at the pure linear solver performance, the speedup is significantly larger, especially for the finest mesh, where a factor of 11 was reached. This different scaling behavior is also shown in Fig. 6. In each case, a stagnation or even decrease of the performance was observed when using more than 16 CPUs in parallel. This is probably due to the fact that the compute nodes on the Grid cluster we used consist of 8 CPUs each, and thus data-transfer between the nodes is having an increasing impact on the performance.

VII. Conclusion

We have presented our blood flow simulation of intracranial aneurysms based on the finite element method. We have developed a tool for the complete work flow (see Fig. 7) for medical application fulfilling the following tasks:

- Importing medical images (DICOM format).
- Reconstruction of the blood volume.
- Fast segmentation of sub-volumes of interest.
- Extraction of the vessel wall (isosurface between blood and surrounding tissue).
- Generation of a tetrahedral mesh.
- Automatic determination of inflow and outflow areas.
- Transient simulation of blood flow dynamics.
- Interactive visualization of simulation results (velocity, pressure, stress).

We have shown that our software is capable of simulating a full pulse cycle of the blood flow within a time span of a few hours. First results calculated for a patient-specific vessel geometry already show relevance for clinical

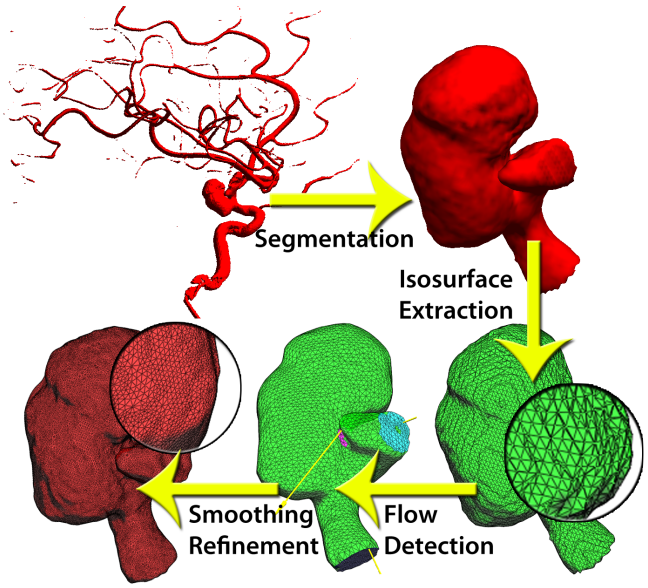


Figure 7. Work flow from vessel volume to FEM mesh.

diagnosis. Using parallelization and computing resources of the Austrian Grid, a speedup factor of 4.5 was achieved. However, the fact that the matrix calculation does not scale with the number of CPUs is an obvious drawback at the moment and will be the focus of future improvements.

As a next step, we are planning to incorporate fluid-structure interaction between the blood and the vessel wall into our simulation. We have already implemented a CSD (computational structural dynamics) module that solves the linear elasticity (Navier-Cauchy) equations for the vessel wall, assumed to be isotropic as a first approximation, once again using the finite element discretization. Since the spatial components of the displacement field are coupled, the AMG algorithm in its present form cannot be used to solve the resulting matrix equation, and we have to resort to the pure CG for now (a solution for this problem is currently being worked on at the Karl-Franzens University Graz). The coupling of CFD and CSD modules has to be done iteratively: The traction forces exerted by the blood lumen on the surrounding wall are applied as Neumann boundary conditions to the structure equation, which in turn yields a new displacement field and thus a new shape for the blood domain. At each time step, this procedure is repeated until an equilibrium is reached [19].

As a future outlook, we also think about simulating stenting, coiling and clipping of aneurysms by changing the geometry (additional mesh from the stent/clip with appropriate boundary conditions and interactions) and the material parameters.

The visualization module we have developed is based on OpenGL and is therefore the next candidate for parallelization and for exploiting Grid technologies. Another possibility for which we already have a prototype is to adjust the rendering parameters for ParaView and use the ParaView client-server framework in order to accomplish parallel visualization, perform rendering and visualization where the data resides, and avoid data transfers when using remote computing resources. For our own visualization module we can think of four options for partitioning simulation results for parallel rendering on remote resources in Grid environments:

- Perform all computationally and data intensive tasks on one supercomputer which is accessible by application users through Grid middleware. The advantage of this approach is that large amounts of result data do not have to be transferred to other computing resources. Although – depending on the hardware – using one resource for solver and visualization may not prove to be the most efficient solution.
- Use separate high performance computers for the solver and the visualization which can be accessed with the help of Grid middleware. Depending on the setup and the location of the two different computing resources, this approach may require time consuming data transfer of the result data. Still the hardware of both clusters could be optimized for performance.
- Create N partitions of the FEM mesh and let cluster nodes render only parts of the whole geometry. This is feasible if we increase the resolution of our simulation from currently about 350,000 nodes to 10^6 nodes or even more. Of course, we still have to render all simulated time steps separately so we even need more computing resources for accomplishing this task within acceptable response times. This approach also has weaknesses regarding the required data transfer and latency between distributed nodes.
- Let every cluster node render another time step of the simulation result. Since we are simulating blood flow (velocity, pressure, stresses) for one complete heart pulse, our FEM model generates at least 1000 result time steps, which have to be rendered and joined together to be available as a video sequence. We want to allow choosing arbitrary camera positions, thus we have to render all time steps again for every distinct camera position. Like the third approach this approach also has weaknesses regarding the required data transfer and latency between distributed nodes.

In any case, the everyday utilization of HPC resources in clinical environments is necessary to minimize the time it takes for a physician to get a detailed picture of future scenarios for patient-specific intracranial aneurysms. For our use-case, Grid computing may prove to be the enabling technology to provide access to these resources and fast and interactive processing for data and computing intensive medical applications.

ACKNOWLEDGMENT

The authors would like to thank Prof. Haase from KF University Graz for providing the AMG Toolbox. We would also like to thank our project partners Simon Schneiderbauer and Peter Fischer for performing comparative calculations with their know-how in commercial simulation systems. This research project is funded by grants of the Austrian Ministry of Traffic, Innovation and Technology (BM:VIT) in the "ModSim - Computational Mathematics" program. More information on project results can be obtained from http://www.medvis3d.at

REFERENCES

[1] I. Foster, "What is the Grid? A three point checklist," June 2002. [Online]. Available: http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf

[2] C. C. Douglas, G. Haase, and U. Langer, *A Tutorial on Elliptic PDE Solvers and Their Parallelization*. Society for Industrial and Applied Mathematics, Philadelphia, 2003.

[3] U. Langer, H. Yang, and W. Zulehner, "Numerical simulation of fluid-structure interaction problems on the grid environment," in *3rd Austrian Grid Symposium*. Wien, Austria: OCG, 2010, pp. 13–27.

[4] S. Prakash and C. R. Ethier, "Requirements for mesh resolution in 3d computational hemodynamics," *Journal of Biomechanical Engineering*, vol. 123, no. 2, pp. 134–144, 2001.

[5] A. Cedilnik, B. Geveci, K. M. J. Ahrens, and J. Favre, "Remote large data visualization in the ParaView framework," in *Eurographics Symposium on Parallel Graphics and Visualization*. Eurographics Association, 2006, pp. 163–170. [Online]. Available: http://www.eg.org/EG/DL/WS/EGPGV/EGPGV06/163-170.pdf

[6] H. Rosmanith and D. Kranzlmüller, "glogin - a multifunctional, interactive tunnel into the grid," in *GRID '04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 266–272.

[7] T. Poston, T. T. Wong, and P. A. Heng, "Multiresolution isosurface extraction with adaptive skeleton climbing," *Computer Graphics Forum*, vol. 17, p. 137, 1998.

[8] J. Schöberl. [Online]. Available: http://www.hpfem.jku.at/netgen/

[9] H. Si. [Online]. Available: http://tetgen.berlios.de/

[10] K. Hildebrandt and K. Polthier, "Anisotropic filtering of nonlinear surface features," *EUROGRAPHICS*, vol. 23, p. 3, 2004.

[11] H. Huang and U. Ascher, "Surface mesh smoothing, regularization and feature detection," *SIAM J. Scient. Comput.*, vol. 31, p. 74, 2008.

[12] K. Stüben, "A review of algebraic multigrid," *J. Comput. Appl. Math.*, vol. 128, p. 281, 2001.

[13] O. C. Zienkiewicz, P. Nithiarasu, R. Codina, M. Vázquez, and P. Ortiz, "The characteristic-based-split procedure: An efficient and accurate algorithm for fluid problems," *Int. J. Numer. Meth. Fluids*, vol. 31, p. 359, 1999.

[14] O. C. Zienkiewicz and R. L. Taylor, *The Finite-Element Method - Volume 3: Fluid Dynamics*. Butterworth-Heinemann, 2000.

[15] M. Liebmann, "A user friendly toolbox for parallel PDE-solvers," 2006. [Online]. Available: http://paralleltoolbox.sourceforge.net/

[16] G. Karypis and V. Kumar, "METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, version 4.0," University of Minnesota, Department of Computer Science and Engineering, Tech. Rep., 1999.

[17] O. Sadowsky, J. Cohen, and R. Taylor, "Projected tetrahedra revisited: a barycentric formulation applied to digital radiograph reconstruction using higher-order attenuation functions," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 4, pp. 461 –473, 2006.

[18] L. Anwei and J. Barry, "Quality local refinement of tetrahedral meshes based on 8-subtetrahedron subdivision," *Mathematics of Computation*, vol. 65, p. 1183, 1996.

[19] U. Küttler and W. A. Wall, "Fixed-point fluid-structure interaction solvers with dynamic relaxation," *Comput. Mech.*, vol. 43, p. 61, 2008.