HIGH PERFORMANCE COMPUTING AND FINITE ELEMENT STENT DESIGN — A PROMISING COMBINATION

B. Verhegghe*, M. De Beule** and R. Van Impe**

* Departement of Mechanical Construction and Production, Ghent University, Ghent, Belgium
** Laboratory for Research on Structural Models, Ghent University, Ghent, Belgium

benedict.verhegghe@ugent.be

Abstract: Numerical simulation by means of the Finite Element method gives precise insight in the mechanical behaviour of a stent during expansion. The complexity of the threedimensional geometry and the necessity to include nonlinear effects (of both material and geometrical nature) however lead in many cases to very long computation times. One way of reducing them is to use parallel computations on multiple processing units. The authors have built a high performance computing cluster from low cost commodity hardware and used it in studying the mechanical behaviour of a 'first generation' Palmaz Schatz stent. The model includes a plastic material behaviour and large displacement theory. ABAQUS/Standard was used for the computations. Simulations were done using 1 to 5 computational nodes. Different types of elements and element sizes were compared. It is concluded that cluster computing can give an important speedup for a very low cost. But enough care should be taken in the building of the model and the tuning of the many options of the Finite Element software. We have obtained near optimal speedup in most cases. Occassional disappointing performance needs further investigation. The effects of including contact problems into the model also remain to be studied.

Introduction

Stents — threedimensional tubular mesh structures — are increasingly used to reopen (partially) obstructed blood vessels. Their design is evolving rapidly. New designs emerge frequently, while older ones are taken off the market. In addition to a variety of experimental tests, computational models (e.g Finite Element models) may provide interesting insights in the mechanical behaviour of stents [1]. Undoubtedly, the design of future generations of stents will be influenced by such numerical models.

A major obstacle for the realistic simulation of the mechanical behaviour of bodies with a complex threedimensional geometry is the large computational cost involved, especially when nonlinear effects (material and geometrical) are included. Such simulations require a lot of memory and take long runtimes, often several days. These long delays between initating a computation and getting the results are very disrupting in the design process. The only solution here is the use of faster computational hardware and parallel processing on multiple cpus. While many of such schemes (e.g. supercomputers) require vast financial resources, clusters of commodity hardware provide an inexpensive alternative for those with limited financial means.

The present study aims at assessing the possibilities and usefulness of a low cost, high performance computing (HPC) cluster for the simulation of the free expansion of a 'first generation' Palmaz Schatz stent.

Materials and Methods

Stent: The geometry of the stent used in our simulations is shown in Figure 1. For clarity the figure shows only part of the stent in circumferential direction. The full stent has a tubular shape with rectangular cutouts. The length, the inner diameter and the thickness of the tube are 16 mm, 1 mm and 0.1 mm, respectively. There are 5 slots along the length of the stent, 12 along its circumference. The slots have a length of 2.88 mm and an opening angle of 15 degrees. In view of the symmetry only 1/24th

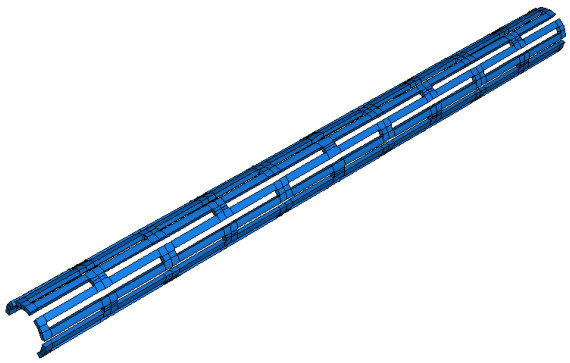

Figure 1: Geometry of the stent (only 2/3 of the circumference is shown).

of the stent has to be modeled for the computations: one half of the length and 1/12th of the circumference. This geometrical model is shown in Figure 2. Symmetry conditions are then applied on one end of the model and on the parts cut by the meridional planes through the middle of two adjacent rows of slots.

When building a complex threedimensional model for

ISSN: 1727-1983

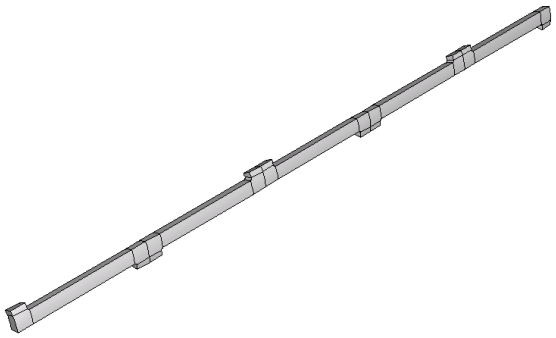

Figure 2: Geometrical model for 1/24th of the stent.

Finite Element analysis, enough care must be taken to the way in which the geometry is constructed. While modern solid modelers provide many ways to get to the same resulting geometrical model, the actual operations that were used can severely reduce the choice of finite element types that can be used with that geometry, as well as the choice of meshing methods. Thus it is very common — even with state of the art software — that general complex domains can only be meshed using tetraeders and free meshing techniques. Tetraeders with linear interpolation however exhibit overly stiff behaviour and require the use of extremily fine meshes, usually making them a far from optimal choice. That leaves only tetraeders with second degree interpolation for use with free meshing.

Using structured meshing techniques and a hexaeder element shape instead opens up a lot more possibilities for the simulation: linear or second order interpolation, reduced or full integration. Structured meshing is also a lot faster than free meshing. But for structured hexahedral meshing to work, it is required that the geometry is carefully built up from parts that allow such meshing. The geometrical model in Figure 2 was assembled from five parts that are (mirrored or not) copies of one basic part, which itself was partitioned in five domains with an adequate geometrical shape to allow structured meshing. These partitions are visible in Figure 2.

The stent is made of 316LN stainless steel. This is modeled adequately by a von Mises plasticity model with isotropic hardening. Young's Modulus was set to $196000\,\mathrm{N/mm^2}$, Poisson's ratio to 0.3, yield stress to $205\,\mathrm{N/mm^2}$. The hardening curve was obtained from Auricchio et al. [2].

Loading was applied as a uniform pressure on the inner surface of the stent. This is an approximation for the pressure exerted by the internal balloon by which the stent is expanded in real life. In a first step the load increases linearly from 0 to $0.7\,\mathrm{N/mm^2}$, matching an inflation of the balloon to 7 atm. After that, the load decreases back to zero. During the first phase, the stent exhibits large plastic deformation. The second phase is merely elastic recoil and involves only small displacements.

HPC cluster: In order to speed up the numerical computations, we use a cluster of computational nodes running in parallel. While high performance computing clusters can be bought from specialised firms, we decided to built our own to keep the total cost minimal. Many low cost clusters have been built in recent years, and there is ample documentation available on the web [3]. For such a cluster Linux is the natural choice as operating system, because it comes free of charge (proprietary systems charge per cpu), scales well, runs on nearly all hardware and is remarkably stable. Furthermore there are several clustering tools (freely) available.

ABAQUS [4], the Finite Element code we use in our numerical simulations, supports parallel computing on clusters of Linux machines through the Message Passing Interface (MPI) specification. We used the LAM-MPI [5] implementation version 7.1 and ABAQUS/Standard version 6.5.

Our cluster consists of a server and five computing nodes, connected with a Gigabit Ethernet network. Since the nodes are intended to be used only for the numerical computations, all they need is a fast processor and (enough) memory. There is no need for the node computers to have a hard disk, a graphics card, a monitor, a mouse or a keyboard. We did put in a hard disk though, because the finite element computations require large disk space for storing intermediate results; storing all files on a central server and accessing them over the network might create a bottleneck, especially if the server and the network are not very performant.

Choosing cost-effective parts, we built our nodes each consisting of a case, a motherboard, an AMD64 3000+ processor, 1GByte of RAM memory and a hard disk of 80 MB, for a price of just over 500 EURO per node [6]. The nodes boot and download their operating system image from the server. This overcomes the need to (re)install system software on each of the nodes. We accomplish this by means of the Warewulf Cluster Toolkit [7]. Though there are ready made solutions available [8, 9] for job scheduling on a cluster, we did not install any of such middleware. Instead, we wrote our own scripts for scheduling and submitting the computational jobs to the cluster, for monitoring the jobs, and for delivering the results back to the users. These scripts involve very little overhead on the nodes and can easily be tuned to the specific jobs running on the cluster (mainly ABAQUS jobs).

Finite element simulations: All simulations comprise the free expansion of the same Palmaz Schatz stent. Different element meshes were used to study the effect of element size and element type. The size of the element is characterized by the mesh seed, taken constant over the whole model. The element type is characterized by its geometrical form and the order of interpolation: linear or quadratic. Where available, we tried both reduced and full numerical integration for the elements. The element types that we used in our simulations are gathered in Table 1. We do not include tetraeders with linear interpolation, because they exhibit a far too stiff behaviour, re-

ISSN: 1727-1983

Table 1: Element types used in the simulations.

type	shape	order	integration
hex8	hexaeder	linear	full
hex8r	hexaeder	linear	reduced
hex20	hexaeder	quadratic	reduced
tet10	tetraeder	quadratic	full

quiring an extremely fine mesh. From a few test runs we found that even the finest mesh that could be processed by our computers did not yield results comparable to those of the other element types.

The quality of the mesh is assessed by comparing the maximal radial displacement in the stent, occurring at the end of the stent. Figure 3 shows the typical deformation of one longitudinal member of the stent. The left end in the figure is the free end of the stent; the right end is the center of the stent. For comparing the results either the value at the end of the loading step or that after unloading could be used: they do not differ much, as the unloading is merely elastic and thus very small.

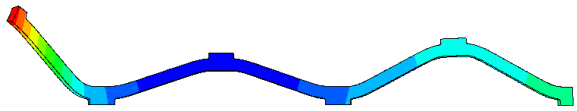

Figure 3: Deformation of (half) a longitudinal member of the stent.

The minimal radial displacement is more interesting from practical standpoint, as it determines the minimal opening of the blood vessel. The ratio of minimal over maximal deformed stent diameter can be used as a measure for the non-uniformity of the stent's expansion [1].

Simulations were run as a single CPU job on one of the cluster's nodes, and as parallel MPI jobs using 2, 3, 4 or 5 nodes. We made sure there were no two jobs running simultaneously on the same nodes. The real run time (wall clock time) was recorded.

An ABAQUS computation starts with a datacheck and job preparation phase, followed by the real analysis phase. In assessing the efficiency of the cluster we neglect the runtime for the datacheck phase, because it is done on the server prior to submitting the job to the cluster. It is only a small fraction of the total runtime anyhow.

Results

Table 2 shows the minimal and maximal radial displacements in the stent after unloading. The results are shown for meshes with different element types and sizes. They are ordered with increasing number of degrees of freedom (DOF). As expected for displacement based elements, the displacement (and accuracy) increase roughly with the number of DOFs.

Table 2: Minimal and maximal radial displacement (mm) of the stent (after unloading) obtained with different element meshes.

type	size (mm)	# of DOFs	displacement min	displacement max
hex8r	0.025	32160	0.753	1.802
hex8	0.025	32160	0.683	1.858
hex8r	0.020	62910	0.692	1.871
hex8r	0.016	113715	0.689	1.892
hex8	0.016	113715	0.690	1.897
hex20	0.025	116883	0.698	1.942
hex8	0.012	253206	0.695	1.912
tet10	0.0315	260808	0.699	1.934

The runtime for some typical computations on 1–5 compute nodes is shown in Table 3. Generally the runtime decreases in accordance with the number of CPUs used. However the table shows some peculiar results. For the quadratic 'hex20' elements e.g., we did not get lower runtimes when using more than 2 computing nodes. Also contrary to expectations, the reduced integration of linear hexaeders takes slightly more time than the fully integrated ones, despite a number of integration points that is eight times lower. Likely this is caused by the hourglass control that is needed for linear interpolation elements with reduced integration.

Table 3: Runtime in seconds for some typical jobs on 1–5 CPUs. Jobs are identified by their element type and size.

CPUs	hex8r 0.025	hex8r 0.016	hex8 0.016	hex20 0.025
1	1592	14640	14456	23065
2	909	7623	7466	12692
3	1197	6310	5780	14660
4	629	5055	4620	15229
5	495	3583	3537	13265

To assess the performance of the cluster, we normalize the runtimes by dividing them by the runtime needed to do the same calculation on a single CPU computer. Figure 4 collects all these relative runtimes in a single graph. The theoretical optimum is $1/N$, where N is the number of CPUs used. Actually recorded runtimes for multiple CPU jobs will always be higher, because there is overhead involved in the communication between the

ISSN: 1727-1983

machines to synchronize their calculations and there are parts of the computation that can not be parallellized.

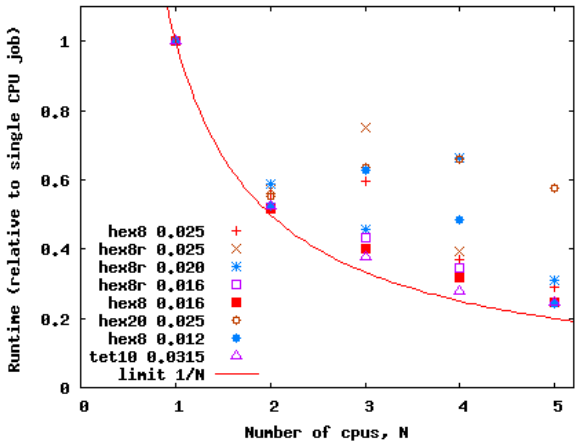

Figure 4: Performance of the cluster with 1 to 5 CPU's.

The figure shows that for a vast majority of our computations, the cluster comes close to the optimal performance. This excellent behaviour can be explained partly by the nature of the computations and the quality of the parallel algorithms, and partly by the dedicated nature of our cluster and minimal overhead of middleware.

Some points in Figure 4 however are lying unexpectedly far above the theoretical limit curve. This occasional misbehaviour has been observed with any type of elements, any size of problem, any number of CPUs involved. We have not found an explanation for the phenomenon, but it certainly is not just an accidental overload of one or more of the nodes: repeating the same job with the same number of CPUs yields the same behaviour, even when run on other nodes.

Discussion

The primary goal of our computations is to predict the expanded shape of the stent. The maximal radial displacement obtained with the different models are close enough to accept any of the models for further investigation of the mechanical behaviour of the stent. Runtime may become a major criterion then. A slight preference for the hexaeder with linear interpolation results. They usually perform very well in the parallel cluster computations.

The fully integrated 'hex8' element yields slightly better results than the 'hex8r' element with reduced integration, while being a little bit faster in large models. The fully integrated is therefore the preferred. However, if there is not just interest in the resulting shape of the stent, but also in the residual stresses in the material, reduced integration may be preferred because the extrapolation of stresses from the integration points to the whole element domain will produce less jagged results. For the same reason hexaeder shaped elements are preferable above tetraeders. For an equal number of DOFs, 'tet10' elements are indeed quite larger than hexaeder 'hex8' or even 'hex20' elements.

We could not find any indications for exceptional network loading or disk access as an explanation for the occasional low performance. It occurs with particular combinations of a finite element mesh (type and size of element) with a number of computing nodes. We suspect the domain decomposition and/or parallellizing software is at fault, but future investigation will have to prove it. Because at any time each of the nodes is involved in only one computational job, it is easy to find out early in the computations whether the job is behaving subpar. With this aim the relative CPU usage of the nodes can be monitored: normally at least one node should have a CPU usage close to 1.0; if none of the nodes exceeds 0.6, there clearly is a problem. Such jobs can then be killed and run again with a different number of nodes.

Conclusions

An HPC cluster reduces the runtime of large computational jobs considerably and reduces the financial cost of finite element studies. The cluster can be built from low cost commoditized components, bringing it within reach of anybody. Nearly optimal acceleration can be obtained with large finite element jobs involving both material and geometrical nonlinearities.

Occasional subpar performance has been observed and needs further investigation. The behaviour of the cluster on problems involving contact also needs further study.

References

[1] De Beule, M., Van Impe, R., Verhegghe, B., Segers, P., and Verdonck, P. Finite element analysis and stent design: Reduction of dogboning. *Proc. of BIOMECH 2005, Regensburg, Germany*, in press, 2005.

[2] Auricchio, F., Di Loretto, M., and Sacco, E. Finite element analysis of a stenotic artery revascularisation through a stent insertion. *Computer Methods in Biomedical Engineering*, 00:1–15, 2000.

[3] LinuxHPC.org. Internet site address: http://www.linuxhpc.org.

[4] Available from ABAQUS,Inc. Internet site address: http://www.abaqus.com.

[5] LAM-MPI. Freely available from: http://www.lam-mpi.org/.

[6] The BuMPer cluster. Internet site address: http://bumps.ugent.be/bumper.

[7] The Warewulf Cluster Toolkit. Internet site address: http://warewulf.lbl.gov.

[8] TORQUE Resource Manager. Internet site address: http://www.clusterresources.com/products/torque/.

[9] Grid Engine. http://gridengine.sunsource.net/.

ISSN: 1727-1983