Hierarchical Parallel Approach in Vascular Network Modeling – Hybrid MPI+OpenMP Implementation

Krzysztof Jurczuk[1], Marek Kretowski[1], and Johanne Bezy-Wendling[2,3]

[1] Faculty of Computer Science, Bialystok University of Technology, Wiejska 45a, 15-351 Białystok, Poland
[2] INSERM, U642, Rennes, F-35000, France
[3] University of Rennes 1, LTSI, Rennes, F-35000, France
{k.jurczuk,m.kretowski}@pb.edu.pl

Abstract. This paper presents a two-level parallel algorithm of vascular network development. At the outer level, tasks (newly appeared parts of tissue) are spread over processing nodes. Each node attempts to connect/disconnect its assigned parts of tissue in all vascular trees. Communication between nodes is accomplished by a message passing paradigm. At the inner level, subtasks concerning different vascular trees (e.g. arterial and venous) within each task are parallelized by a shared address space paradigm. The solution was implemented on a computing cluster of multi-core nodes with mixed MPI+OpenMP support. The experimental results show that the algorithm provides a significant improvement in computational performance compared with a pure MPI implementation.

1 Introduction

The continuously increasing need for computing power and physical and economic limitations of processor frequency scaling (i.e. significant increase of costs and energy usage) caused that parallel machines have become the only known alternative to improve computing performance [1]. Firstly, the need of high performance arises from the necessity to solve problems of ever-rising size at the limits of available computing resources. Moreover, parallel computing seems to be more suitable to mimic natural world processes that may happen in the same time and are quite often interrelated with each other. Thus, parallel processing appears to be one of the most relevant issues in modern scientific computing [2].

In our previous studies, we developed a physiological model reflecting both morphology and functions of vascular networks in clinical images [3]. The solution consists of a macroscopic model (vascular network and pathological anomalies) and a microvascular model (blood flow simulation in capillaries and contrast agent diffusion processes [4]). Moreover, we coupled this two-level vascular model with imaging CT and MR simulators. Such a model-based approach represents a non-invasive way to control physiological parameters, what would be difficult or even impossible to do in real experiments. The whole solution constitutes a good

R. Wyrzykowski et al. (Eds.): PPAM 2011, Part I, LNCS 7203, pp. 376–385, 2012.

way to improve the interpretation of dynamic medical images by linking image descriptors with morphological and functional perturbations, thus offering the potential to reveal early image indicators of pathologies.

In the model, the structure of vascular networks is obtained in the process of vascular development. Initially, we proposed a sequential algorithm [3]. Although we applied many algorithm and code optimizations, the simulation was still a time expensive process. Later, we introduced the distributed memory algorithm that parallelized the vascular development [5] (message passing interface (MPI) [6] implementation on computing cluster). Moreover, we proposed an advanced modeling framework [7] able to efficiently simulate vascular development on a computing cluster (distributed memory approach) as well as on low-cost multi-core desktop machines (shared memory approach - OpenMP [8] implementation).

Although in all our previous distributed memory algorithms we were able to gain substantial speedups on computing clusters of nodes with single-core chips, nowadays, the multi-core chips in clusters seem to be an industrial trend. Moreover, a combination of shared memory and message passing paradigms in one application may provide a better efficiency than e.g. pure MPI version [9]. Therefore, in this paper, we propose a two-level hybrid parallel algorithm of vascular development, that employs both shared address space and message passing paradigms on a cluster of nodes with multi-core chips (mixed MPI+OpenMP implementation). The main aim of this work is to further accelerate the simulation process, which will increase the possibility to create more elaborate and precise vascular models. Furthermore, our intention is to bring the model closer to reality in which processes of vascular development are performed inherently in a parallel way [10]. Many other vascular system have been proposed (e.g. [11], [12]), however, as far as we know, all the previous solutions, published by other authors, have been using only sequential approaches of vascular development. On the other hand, one can find at least several other applications of hybrid parallel modeling in computational biology and medicine, e.g. in PET image reconstruction algorithms [13] or in cardiac simulations [14].

In the next section, the vascular model and sequential algorithm of vascular development are recalled. In Sect. 3 the hybrid parallel algorithm of the vascular development is explained. An experimental validation is performed in Sect. 4. Conclusion and some plans for future research are sketched in the last section.

2 Vascular Model Description

In this paper, we focus on vascular development algorithms on the macroscopic level. Therefore, this section describes basic features of the macroscopic model. Then the sequential algorithm of vascular development is recalled.

In the macroscopic model we can distinguish two main components: the tissue and the vascular network [3]. The tissue is represented by a set of Macroscopic Functional Units (MFU) that are distributed inside the specified organ shape. An MFU is a small, fixed size part of the tissue and is characterized by a class that determines most of its structural/functional properties (e.g. size, probability of mitosis and necrosis) and also physiological features (e.g. blood pressures, blood

flow rate). Several classes of MFUs can be defined to differentiate functional or pathological regions of tissue (e.g. normal, tumoral). Moreover, the class of MFU can be changed over time, which makes it possible to simulate the evolution of a disease (e.g. from benign nodule to malignant tumor).

2.1 Vascular Network

The model expresses the specificity of the liver, although most of its features are not linked with any specific organ. The liver stands out from other organs by the unique organization of its vascular network that consists of three vessel trees. Hepatic arteries and portal veins deliver blood to tissue, whereas, the hepatic venous tree is responsible for blood transport back to the heart.

In the model, each vascular tree is composed of vessels that can divide creating bifurcations. A vessel segment (part of vessel between two consecutive bifurcations) is represented by an ideal, rigid tube with fixed radius, wall thickness and position. The model distinguishes vessels larger than capillaries, while the capillaries themselves are hidden in the MFUs (micromodel [4]). Blood is transferred from hepatic arteries and portal veins to hepatic veins through MFUs. Vessel intersections (anastomosis), which occur particularly among vessels with very small radii or in pathological situations, are not taken into account. As a result, each vascular tree forms a binary tree.

In the model, the blood is treated as a Newtonian fluid (with constant viscosity) and its flow is governed by Poiseuille's law. Moreover, the vessels' parameters (e.g. radius, blood flow) are calculated according to two following physical laws. At each bifurcation the law of matter conservation is observed, i.e. the quantity of blood entering and leaving a bifurcation is the same. Second constraint deals with the decreasing vessel radii in the vascular tree, creating a morphological dependency between the radius of a vessel and radii of its two descendants.

2.2 Sequential Algorithm of Vascular Network Development

The algorithm begins with the model initialization [3]. Few vessels are placed in the 3D shape of an organ whose size is a fraction of the adult one. Afterwards, in discrete time moments (called cycles), the organ enlarges its size (growth phase) until it reaches its full, mature form. Additionally, each cycle consists of subcycles during which each MFU can divide and give birth to a new MFU of the same class (mitosis process) or die (necrosis process). The processes of mitosis and necrosis are repeated in consecutive subcycles until spaces appearing during the growth phase are filled by new MFUs.

The appearance and development of new vessels are induced by new MFUs that are initially not perfused by the existing vascular system. As a result, for each new MFU a fixed number of the closest/candidate vessels (in each tree) is found. Then each candidate vessel creates a new bifurcation that temporarily perfuses the MFU, i.e. new temporary vessels are sprouted. The spatial position of the bifurcation point is controlled by local minimization of additional blood volume needed for the MFU perfusion.

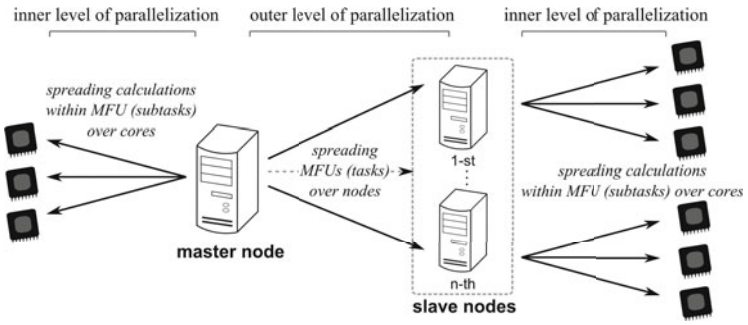


Fig. 1. The two-level parallel perfusion process. At the outer level, new MFUs are spread over nodes and each node is responsible for searching optimal bifurcation points. At the inner level, within each MFU calculations concerning different vascular trees are delegated to different cores.

Only one candidate vessel from each tree can finally be designated to perfuse the new MFU. Therefore, the algorithm tests all possible combinations of candidate vessels (a single combination consists of one vessel from each tree). Since only non-crossing vessels can be accepted, the algorithm firstly detects intersections between vessels coming from the same tree or from two different trees (e.g. between arteries and veins) and rejects theses vessels. Finally, the combination with the lowest sum of volumes is chosen to permanently perfuse the MFU and a recalculation of vessels' characteristics in all vascular trees is performed.

In each subcycle, after the reproduction (mitosis and perfusion processes), the algorithm goes to the degeneration phase. Based on the necrosis probabilities of individual MFUs, some of them can die (necrosis process). Next, all the vessels supplying these MFUs retract and disappear (retraction process), and the algorithm goes back to the reproduction phase again.

3 Hybrid Parallel Algorithm of Vascular Development

At the beginning, the general idea of the algorithm is described. Then we present in more detail parallel perfusion and retraction algorithms. The main attention is focused on the perfusion phase since it is the most time consuming part of the algorithm (from 70% to 95% of the total simulation time).

The hybrid algorithm applies a hierarchical (two-level) parallelism. At the outer level, tasks are carried out by pool of processes running on different processing nodes. Interactions between the nodes are accomplished by the message passing paradigm with the master-slave model [2]. At the inner level, parallel subtasks are spread over threads running on different cores and the shared memory space paradigm is exploited. For the sake of explanation clarity, we neglect the mapping of threads/processes to cores/processing nodes and assume that one node is identified with one process and one core is identified with one thread.

During the whole simulation, each processing node has its own copy of the organ. Therefore, at the beginning, the master node broadcasts the initial vascular

system and tissue providing the same starting information for all slave nodes. After that, cycle and subcycles are iterated until the adult organ is obtained.

Each subcycle starts from a sequential mitosis performed at the master node. Next, the two-level parallel perfusion is performed (see Fig. 1). The new MFUs that are created during the mitosis are spread over slave nodes (outer level of parallelization). Moreover, calculations within each single MFU are divided between cores (inner level of parallelization). After the perfusion process, the degeneration phase follows. At the beginning, at the master node the sequential necrosis is carried out. Next, the two-level parallel retraction is performed.

3.1 Two-Level Hybrid Parallel Perfusion Algorithm

The outer level parallelization of the perfusion is based on the distributed memory approach of vascular development [5]. After the sequential mitosis, the master node spreads new MFUs (tasks) over slave nodes (see Fig. 1) and then this node is responsible for managing the perfusion process. When it receives a message with an optimal bifurcation of one of the new MFUs, it takes a decision about permanent perfusion. Communication latency and independent work of slave nodes cause that vascular networks at individuals nodes can be slightly different (tree nonuniformity). Therefore, the master node searches, in its vasculature, the vessels related with the proposed optimal bifurcation (vessels to form the optimal bifurcation). If at least one of these vessels cannot be found, then the MFU is rejected. But in the other case, the new MFU is permanently perfused and all organ changes related with the new MFU are broadcasted over slaves.

As regards the slave nodes, each of them is responsible for searching optimal bifurcation points to perfuse the received new MFUs. Each time, when the search ends successfully, the optimal bifurcation parameters are sent to the master node. Next, if there are any queued messages with permanent organ changes broadcasted by the master node, the slave node applies these changes and continues to perform its remaining tasks. Moreover, when the master node is under-loaded (e.g. as a result of small number of slave nodes), it can also perform calculations to find parameters of optimal bifurcation points [7].

The inner level parallelization of the perfusion, both at the master and slave nodes, introduces a possibility to divide calculations concerning single MFU (see Fig. 1). Individual cores are responsible for the calculation concerning different vascular trees (i.e. hepatic arteries, portal veins or hepatic veins in liver).

In the case of the master node, there are two algorithm blocks (i.e. making decision about permanent perfusion and permanent perfusion) during which individual cores are responsible for calculations in different vascular trees. When the master node receives a message with an optimal bifurcation to perfuse a new MFU, each core tries to find, in its assigned tree, the vessel that may create the proposed optimal bifurcation. If all cores find such vessels, the new MFU is permanently perfused in a parallel way in all vascular trees based on the information from the optimal bifurcation. However, if at least one of the cores cannot find the vessel, the other cores abandon their jobs and the MFU is rejected.

In the case of the slave nodes, the inner parallelization is applied to spread calculations within each new MFU (see Fig. 1) during following subtasks: searching of optimal bifurcations, choice of one optimal bifurcation and permanent perfusion. At first, each core of a slave node searches the candidate vessels in its vascular tree, then it creates one optimal temporary bifurcation to each found candidate vessel and recalculates tree characteristics, taking into account the new vascular structures for these temporary bifurcations. Afterwards, all possible combinations of candidate vessels (a single combination consists of one vessel from each tree) are created, sorted according to increasing volume and then tested. Each time, to determine the volume of a verified combination, calculations in different vascular trees are divided between cores. Moreover, the inner parallelization is applied when slave nodes perform permanent perfusions as a result of organ changes broadcasted by the master node.

3.2 Two-Level Hybrid Parallel Retraction Algorithm

In comparison to our previous solutions [5], [7], in the hybrid one, we decided to pay more attention to the retraction phase. The profiling results showed that the time needed for that part of the algorithm is too short to apply a sophisticated message passing strategy. However, in order to create a possibility to gain higher performance (especially in the context of Amdahl's law of a maximum attainable speedup [15]), we decided to apply here a naive message passing strategy at outer level and shared address space paradigm at inner level.

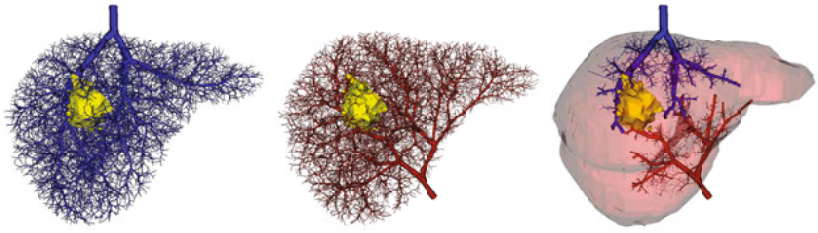
After the sequential necrosis, the master node broadcasts to all other nodes identifiers of the MFUs that have to be removed. Then, the entire algorithm of retraction is performed at each node. As regards the inner parallelization, both the master and slave nodes spread calculations concerning different vascular trees over cores. Hence, the MFUs are disconnected concurrently in all vascular trees.

4 Experimental Results

This section focuses on performance analysis. The presented mean results come from experiments on the vascular development algorithm starting from small size configurations (about 1000 MFUs) and ending on large configurations (about 50000 MFUs and consequently 300000 vessel segments). Figure 2 shows a visualization of one of the simulated vascular networks.

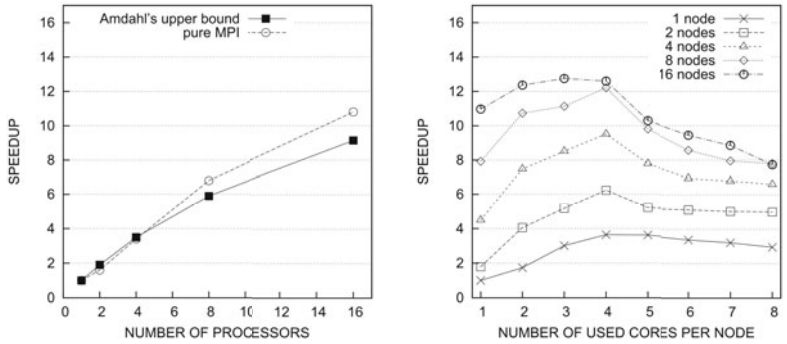
The solution was implemented in C++ with support MPI [6] and OpenMP [8] interfaces. At the outer level of parallelization, the MVAPICH2 version 1.6.1 as MPI2 implementation over infiniband networks was used. The OpenMP was exploited at the inner level of parallelization. The Intel C++ Compiler 10.1 was used. For performance measuring we made use of the Multi-Processing Environment (MPE) library with the graphical visualization tool Jumpshot-4 [6].

Two computing clusters were used. The first one consisted of sixteen SMP nodes running on Linux and connected by an Infiniband network. Each node was equipped with two single-core chips (two 64-bit Xeon 3.2GHz CPUs) with



a) b) c)

Fig. 2. Visualization of an adult liver (about 50000 MFUs and 300000 vessel segments): a) hepatic veins with a tumor shape, b) portal veins with a tumor shape, c) main hepatic arteries, portal veins and hepatic veins with liver and tumor shapes



a) b)

Fig. 3. Speedup of the pure MPI version on the cluster of: a) single-core dual-socket SMP nodes, b) multi-core dual-socket SMP nodes

2MB L2 cache, 2GB of RAM and an Infiniband 10GB/s HCA connected to a PCI-Express port. The second cluster was also built with sixteen SMP nodes but each node was equipped with two multi-core chips (two 64-bit quad-core Xeon 2.66 GHz CPUs) with 2MB L2 cache and 8GB of RAM.

Figure 3a shows the mean speedup of the distributed memory algorithm [5] (pure MPI implementation) on the computing cluster of single-core nodes. It is clearly visible that good performance was obtained. Moreover, we were able to obtain the speedup better than its upper bound value based on Amdahl's law [15]. It results from the introduced periodical memory reallocation mechanisms improving memory cache usage. However, the same algorithm running on the computing cluster of multi-core nodes tends to decrease its performance if too many cores inside each node are involved in computations (see Fig. 3b). Profiling results showed that it is caused by intra-node memory traffic that increases the mean time of message passing and the time of searching optimal bifurcations.

The performance of the proposed two-level hybrid solution is presented in Fig. 4. The outer level parallelization is accomplished by MPI processes, while the inner level one by OpenMP threads. We verify two cases: i) one MPI process (three

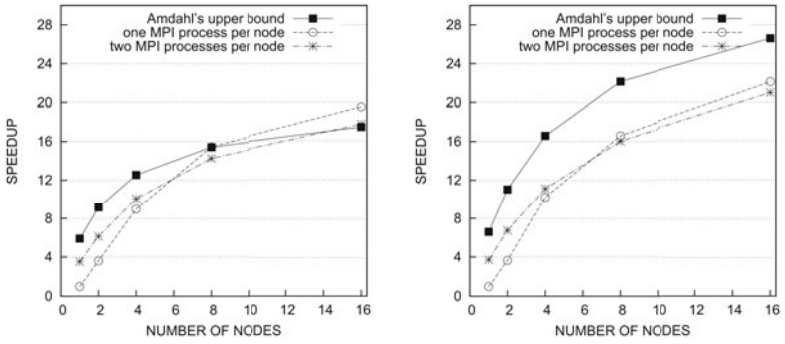


Fig. 4. Speedup of the two-level hybrid version on a cluster of multi-core SMP nodes: a) without and b) with two-level parallelization during retraction

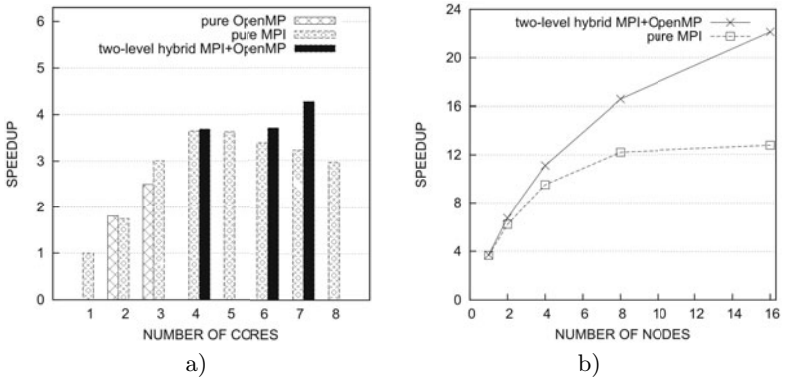


Fig. 5. Performance comparison of pure MPI, pure OpenMP [7] and hybrid MPI-OpenMP versions on the cluster of multi-core SMP nodes: a) speedup within one node, b) speedup across nodes. Different configurations in the hybrid version within one node: four cores (master process - one thread, slave process - three threads), six cores (master process - three threads, slave process - three threads), seven cores (master process - one thread, two slave processes - each three threads)

OpenMP threads) per node and ii) two MPI processes (six OpenMP threads) per node. Figure 4a shows the mean speedup if the two-level hybrid parallelization is applied only during the perfusion process. It is clearly visible that the obtained efficiency is better than in the pure MPI version. If also the retraction process exploits the two-level hybrid parallelization, the gained speedup still remarkably increases. Although in the presented results the retraction process takes approximately only 2% of total CPU time during sequential algorithm execution (in contrast to 95% for the perfusion), it is enough to further accelerate the solution, especially in terms of Amdahl's law.

It can be also observed that the number of MPI processes per node has an influence on the performance (see Fig. 4). When the number of cluster nodes

increases (i.e. eight and more), only one MPI process and consequently three OpenMP threads should be run per node even though each node is equipped with eight cores. The performance reduction, in the case of two MPI processes per node, comes from the higher load of a master process having more slaves to manage. Profiling results indicated that an overloaded master process can be inefficient in broadcasting permanent changes, i.e. time between sending the message with an optimal bifurcation by a slave and making a decision about permanent perfusions till broadcasting related changes by a master is lengthened excessively. Such a situation increases the tree's nonuniformity between slave nodes, which causes more MFUs rejections and finally more algorithm iterations.

In Fig. 5a, the performance of our all parallel solutions running within one cluster node is summarized. The best speedup can be gained with the two-level hybrid algorithm, especially in the case of seven cores and three MPI processes (master process - one thread, two slave processes - each three threads).

On the other hand, Fig. 5b shows the summary comparison between the best results of pure MPI version and hybrid one across nodes. It is clearly visible that the proposed hybrid solution provides better speedup than the pure MPI version. The improvement rises with the increase in the number of cluster nodes.

The hybrid solution was tested on the cluster of nodes with two quad-core chips. Hence, it may seem a waste of computational power since only three or six cores in each node can be arranged in computations. However, due to limited memory bandwidth, it can be even advantageous (e.g. in terms of power consumption) to use fewer treads than available cores [16]. On the other hand, there also exist six-cores processors (e.g. AMD Phenom II X6) that could be used with better efficiency. Moreover, most of internal human organs are supplied by two vascular trees (i.e. arterial and venous) and then the proposed approach would be more suitable for the most widespread two and quad-core processors.

5 Conclusion

In the paper a two-level parallel algorithm of vascular development is presented. The algorithm employs shared memory and message passing paradigms (mixed MPI+OpenMP implementation). Experimental results on a multi-core cluster show that a significant improvement in computational efficiency has been obtained. As a result, it helps us to extend the vascular model and to test multiply sets of parameters in reasonable period of time.

In the future, we will continue to work on the hybrid approach, among others, to investigate the influence of communication and calculations overlapping model, e.g. splitting one OpenMP thread off only to handle communication and the others to perform useful calculations.

Acknowledgments. This work was supported by the grant W/WI/3/11 from Bialystok University of Technology.

References

1. Gebali, F.: Algorithms and Parallel Computing. Wiley, NJ (2011)
2. Hager, G., Wellein, G.: Introduction to High Performance Computing for Scientists and Engineers. CRC Press, Boca Raton (2010)
3. Kretowski, M., Rolland, Y., Bezy-Wendling, J., Coatrieux, J.-L.: Physiologically Based Modeling for Medical Image Analysis: Application to 3D Vascular Networks and CT Scan Angiography. IEEE Trans. Med. Imaging 22(2), 248–257 (2003)
4. Mescam, M., Kretowski, M., Bezy-Wendling, J.: Multiscale Model of Liver DCE-MRI Towards a Better Understanding of Tumor Complexity. IEEE Trans. Med. Imaging 29(3), 699–707 (2010)
5. Jurczuk, K., Kręt̨owski, M., Bézy-Wendling, J.: Vascular Network Modeling - Improved Parallel Implementation on Computing Cluster. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2009, Part I. LNCS, vol. 6067, pp. 289–298. Springer, Heidelberg (2010)
6. Pacheco, P.: Parallel Programming with MPI. Morgan Kaufmann Publishers, San Francisco (1997)
7. Jurczuk, K., Kretowski, M., Bezy-Wendling, J.: Vascular System Modeling in Parallel Environment - Distributed and Shared Memory Approaches. IEEE Trans. Inf. Technol. Biomed. 15(4), 668–672 (2011)
8. Chapman, B., Jost, B.G., van der Pas, R., Kuck, D.J.: Using OpenMP: Portable Shared Memory Parallel Programming. MIT Press, Cambridge (2007)
9. Rabenseifner, R., Hager, G., Jost, G.: Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes. In: 17th Euromicro Int. Conf. on Parallel, Distributed & Network-based Processing, pp. 427–436. IEEE Press, Weimar (2009)
10. Shima, D.T., Ruhrberg, C.: Angiogenesis. In: Pelengaris, S., Khan, M. (eds.) The Molecular Biology of Cancer, pp. 411–423. Blackwell, Oxford (2006)
11. Zamir, M.: Arterial Branching Within the Confines of Fractal L-system Formalism. Journal of General Physiology 118, 267–275 (2001)
12. Schreiner, W., et al.: Optimized Arterial Trees Supplying Hollow Organs. Medical Engineering & Physics 28(5), 416–429 (2006)
13. Jones, M.D., Yao, R., Bhole, C.P.: Hybrid MPI-OpenMP Programming for Parallel OSEM PET Reconstruction. IEEE Trans. Nucl. Sci. 53(5), 2752–2758 (2006)
14. Pope, B., et al.: Performance of Hybrid Programming Models for Multiscale Cardiac Simulations: Preparing for Petascale Computation. IEEE Trans. on Biomed. Eng. 58(10), 2965–2969 (2011)
15. Amdahl, G.M.: Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In: Proc. AFIPS, Atlantic City, vol. 30, pp. 483–485 (1967)
16. Curtis-Maury, M., et al.: Prediction Models for Multi-dimensional Power-Performance Optimization on Many Cores. In: Proc. 17th Int. Conf. on Parallel Architectures & Compilation Techniques, pp. 250–259. IEEE Press, Toronto (2008)