# LOAD BALANCING IN PARALLEL IMPLEMENTATION OF VASCULAR NETWORK MODELING

Krzysztof Jurczuk[1], Marek Kręţowski[1], Johanne Bézy-Wendling[2,3]

[1]Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland
[2]INSERM, U642, Rennes, F-35000, France
[3]University of Rennes 1, LTSI, Rennes, F-35000, France

**Abstract:** In this paper, load balancing mechanisms in a parallel algorithm of vascular network development are investigated. The main attention is focused on the perfusion process (connection of new cells to vascular trees) as it is the most time demanding part of the vascular algorithm. We propose several techniques that aim at balancing load among processors, decreasing their idle time and reducing the communication overhead. The core solution is based on the centralized dynamic load balancing approach. The model behaviors are analyzed and a tradeoff between the different mechanisms is found. The proposed mechanisms are implemented on a computing cluster with the use of the message passing interface (MPI) standard. The experimental results show that the introduced improvements provide a more efficient solution and consequently further accelerate the simulation process.

**Keywords:** parallel algorithms, load balancing, cluster computing, computational modeling, vascular network

## 1. Introduction

The last decade has seen a revolution in high performance scientific computing [1]. This is mainly due to a tremendous development of parallel computers. Because of physical and economic limitations of processor frequency scaling (e.g. power consumption and consequently heat generation) both industry and science prefer to use many moderately fast processors, rather than a single high speed processing unit. Nowadays, computing clusters and multi-core/multi-processor computers are becoming widespread platforms [2]. As a results, many scientists have gained an easy access to parallel machines able to support an ever-rising demand for high-speed processing.

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 6, pp. 41-61, 2010.

In this paper, we focus on applying parallel computing to modeling and simulation in biomedical research on vascular networks. Vascular networks play a very important role in the detection process of various pathological anomalies since changes in their structure and function can be directly caused by diseases [3]. Moreover, when a contrast agent is administrated, these anatomical or functional modifications can appear in medical images. Therefore, the modeling of vascular systems can help to understand the mechanisms of dynamic image formation and support the development of methods to detect early disease indicators.

Nevertheless, one of the most important and simultaneously the most difficult challenges in model designing is to choose the level of details to include in the model [4]. A high quality vascular model has to take into account the most essential physiological and anatomical properties and to disregard those elements whose role is insignificant. Such a model should also be effective in practical cases, i.e. computational simulations must be performed in a reasonable time. Therefore, it seems to be very useful and desirable to take advantage of parallel computing in modeling of living organisms and particulary in the case of the vascular system modeling. Firstly, we are able to provide a significant increase in computational performance by splitting problem into parts that are performed by separate processors in parallel [5]. Secondly, using multiple processing units often allows us to provide a more precise solution or to solve a larger problem in a reasonable amount of time. Moreover, parallel computers are very useful when the same problem has to be evaluated multiple times, with different parameters for instance.

In parallel systems, computations are decomposed into tasks. In order to achieve an efficient solution, overheads of the parallel tasks have to be minimized [6]. One ought to strive to reduce the total amount of time some processors are idle while the others are still busy. Secondly, the amount of time spent for communication between processors has to be also minimized. These two objectives are often in conflict with each other, therefore one should find an optimal tradeoff between them and propose load balancing mechanisms able to spread the tasks evenly across the processors.

Load balancing techniques used in parallel algorithms can be broadly classified into two major categories: static and dynamic. In the former type, usually referred to as the mapping problem [7] or scheduling problem, tasks are distributed among processors before the execution of the algorithm based on a priori knowledge. Several techniques for static load balancing have been developed, e.g. round robin algorithm [1], simulated annealing (stochastic optimization algorithm) [8], [9] or real-coded genetic algorithms [10]. However, there exists a large class of applications that workloads of tasks are uneven and unpredictable and may change during the computation. Therefore, for these applications we are not able to spread the tasks

evenly across processors beforehand. In this case, dynamic load balancing (DLB) schemes are needed. In DLB, the decision on task arrangement is made during the execution of the program based on the current load status. Moreover, such an approach can be more appropriate in the case of heterogeneous parallel machines with additional sources of an external load. Due to a big and fast growing number of different dynamic load balancing techniques, we refer the reader to [11] for a detailed survey of DLB algorithms.

In our previous studies, we developed a two-level physiological model of vascularization [12], [13]. It consists of a macroscopic model able to simulate growth and pathological structural modifications of vascular network, and a microvascular model responsible for simulation of blood and contrast agent transport through capillary walls [14]. Initially, we made use of a sequential algorithm of vascular development to obtain the structure of the vascular network. The vascular development results from a progressive increasing number of cells and consequently a progressive increasing number of vessels that support blood supply for these cells. Subsequently, we introduced the basic [15] and improved [16] parallel solutions of vascular growth algorithm. These two parallel solutions were implemented on a computing cluster with the use of the message passing interface (MPI) standard [17].

In this paper, we propose mechanisms that try to achieve balanced load among processors and reduce the communication overhead in the parallel modeling of the vascular network growth. Both static and dynamic algorithms are used. We consider a centralized model, in which tasks are generated at the central scheduler (master processor) and are allocated to slave processors. Workloads of the tasks are uneven and it is impossible to estimate their execution times because each particular job has an indeterminate number of steps to reach its solution. In addition, we have to deal with a dynamically changing structure of vascular trees. We analyze various model behaviors in a parallel environment and propose a tradeoff between the different load balancing strategies in order to provide a more efficient solution and consequently to further accelerate the simulation process.

The rest of the paper is organized as follows. In the next section, the vascular model is described and sequential and both parallel algorithms of vascular network development are recalled. In section 3 the load balancing mechanisms are presented. An experimental validation of the proposed mechanisms is performed in section 4. The last section contains the conclusion and future works.

## 2. Model Description

In the macroscopic part of the model we can distinguish two main elements: the tissue and the vascular network. The tissue is represented by a set of Macroscopic Functional Units (MFU) that are regularly (but randomly) distributed inside the specified, three-dimensional organ shape. The vascular network is composed of vessels supplying the MFUs. The microvascular part of the model is hidden in MFUs and is responsible for the propagation of an MRI contrast agents in the tissue. The five-compartments [18] and axially distributed Blood Tissue EXchange (BTEX) [14] contrast propagation approaches were proposed.

The most important and original part of the work presented here concerns the algorithms of vascular development on macroscopic level. Therefore, in the next part of this section, the macroscopic part of the model is described in more details followed by the presentation of sequential and parallel algorithms of vascular development.

### 2.1 Macroscopic model

**Tissue modeling** A MFU is a small, fixed size part of tissue to which a class is assigned that determines most of functional/structural (rhythm of mistosis/necrosis) and physiological features (e.g. blood flow rate, blood pressure). Several classes of MFUs can be defined to differentiate functional or pathological regions of tissue (e.g. tumoral, normal). Moreover, the MFU class can be changed over time, which makes it possible to simulate the evolution of a disease (e.g. from HepatoCellular Carcinoma to necrotic tissue or from benign nodule to malignant tumor). In order to introduce more natural variability, certain parameters (such as blood flow rate) are described by defined distributions.

**Vascular Network Modeling** Most of model features are not linked with any specific organ. However, it is very hard to model a vascular network without any kind of specialization. In our work, the model expresses the specificity of the liver. The liver plays a major role in the metabolism and has a number of functions in the body including protein synthesis, detoxification, glycogen storage, etc. [19]. Moreover, it stands out from other vital organs by its unique organization of vascular network that consists of three vessel trees. Hepatic arteries and portal veins deliver blood to cells, 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 (see Fig. 1a). A vessel segment (part of vessel between two consecutive

bifurcations) is represented by an ideal, rigid tube with fixed radius, wall thickness, length and position. The geometry of capillaries is not considered in the model. These smallest vessels are hidden in the MFUs (microvascular model). According to the morphometrical investigation dealing with bigger vessels, e.g. conducted by Zamir [20], it is assumed that a single vascular structure has a form of a binary tree. In effect, anastomoses (e.g. mutual vessel intersections) that may occur particulary in pathological situations or among vessels with very small radii are not taken into account.

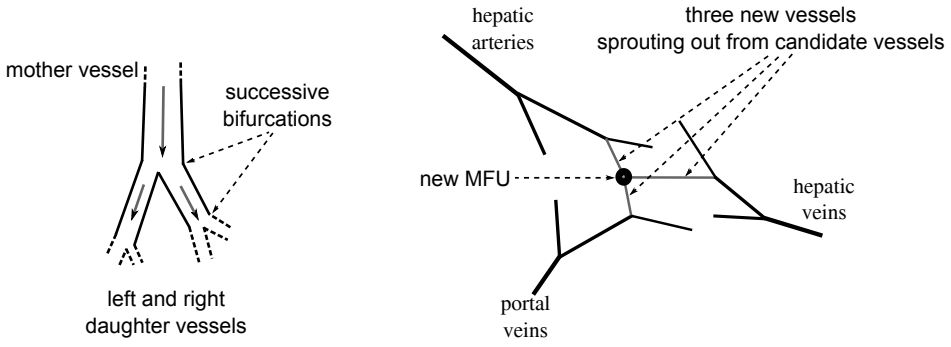


**Fig. 1.** Part of binary vascular trees: a) mother vessel and its two daughter vessels connected by a bifurcation, b) new MFU perfusion by three new vessels sprouting out from candidate vessels each from different vascular tree.

In the model, the blood is treated as a Newtonian fluid that is transferred from hepatic arteries and portal veins to the hepatic veins through MFUs. Its flow is modeled as a non-turbulent streamline flow in parallel layers (laminar flow) and governed by Poiseuille's law:

$$\Delta P = Q\frac{8\mu l}{\pi r^4}\ , \tag{1}$$

where $l$ is the vessel length, $r$ is its radius, $Q$ is the blood flow and $\Delta P$ is the pressure difference between the two vessel extremities. Moreover, at each bifurcation the law of matter conservation has to be observed:

$$Q = Q_r + Q_l\ . \tag{2}$$

It says that the quantities of blood entering a bifurcation (blood flow in parent vessel $Q$) and leaving the bifurcation (blood flows in the right and left daughter branches

$Q_r + Q_l$) are equal. Another constraint deals with the decreasing vessel radii in the vascular trees when we move from proximal to distal segments of vascular network, creating/describing the relation between the mother vessel radius ($r$) and the radii of its two daughters (right $r_r$ and left $r_l$):

$$r^\gamma = r_r^\gamma + r_l^\gamma \,, \tag{3}$$

where $\gamma$ varies between 2 and 3 [21].

### 2.2 Sequential Vascular Network Growth Algorithm

An adult organ is obtained in a vascular development process that is modeled as a analogy to a hyperplasia process (progressive increasing number of cells [22]). The simulation starts with an organ whose size is a fraction of a mature one. After parameters' initialization, in discrete time moments (called cycles), the organ enlarges its size (growth phases). The relative positions of MFUs remain unchanged but distances between them are increased, leading to appearance of empty spaces. Subsequently, these spaces are filled by new MFUs in consecutive subcycles. In each subcycle, each MFU can divide and give birth to a new MFU of the same class (mitosis process) or die (necrosis process). Probabilities of mitosis and necrosis are sensitive to the time and they decrease exponentially with the age of the MFU. New cycle starts only when the current organ shape is totally filled by MFUs. The increasing number of MFUs induces the development of a vascular network which is responsible for the blood delivery.

New MFUs that appear during the mitosis process are initially ischemic, i.e. they are not perfused by the existing vascular network. Therefore, for each new macroscopic functional unit a fixed number of the nearest/candidate vessels is found. Then, each candidate vessel temporarily creates a bifurcation perfusing the MFU (one vessel is replaced by three vessels connected by a bifurcation point). The spatial position of the bifurcation is controlled by local minimization of the additional blood volume necessary to the MFU perfusion (Downhill Simplex algorithm [23]).

The above process can be regarded as a kind of competition because only one vessel in each tree can be finally designated to permanently perfuse the new macroscopic functional unit. Additionally, the problem of avoiding possible collisions between perfusing vessels is taken into account. The algorithms detects intersections between vessels coming from the same tree or from two different trees and rejects the related candidate vessels. Finally, from among the remaining candidate vessels, the combination (a single combination consists on one vessel from each tree) with the lowest sum of volumes is chosen to permanently perfuse the MFU

(see Fig. 1b). Afterwards, a recalculation of vessels' characteristics (i.e. pressure, radius etc.) in the vascular trees is performed. This step ensures the consistency of the characteristics according to the assumed physical and physiological laws.

After the reproduction process (i.e. mitosis and perfusion processes), comes the degeneration phase. At this step of the algorithm, few MFUs can die (necrosis process) and then all the vessels supplying these MFUs retract and disappear (retraction process). Next, the algorithm goes back to the reproduction process.

### 2.3 Parallel Vascular Network Growth Algorithm

In the presented sequential algorithm of vascular growth, all MFUs are connected to the vascular network one by one. Each MFU perfusion involves the necessity of creating and testing a number of temporary bifurcations. It requires a great number of calculations to face the imposed constraints to assure the consistency of vascular trees. A vascular tree is consistent if: i) it has the same blood pressure and fixed blood flow in all terminal vessels attached to MFUs and ii) the Poiseuille's law in each its vessel and the matter conservation and bifurcation laws in each its bifurcation are fulfilled. As a result, the perfusion process is the time dominant operation in the organ growth simulation. Profiling results (e.g. execution times of specific methods) showed us that it can generally consume around 70-90% of the total CPU time needed to develop an adult organ. Therefore, in order to accelerate the simulation process we proposed two parallel vascular growth algorithms [15], [16] that spread the most time consuming computations between processors and consequently are able to decrease the simulation time. Moreover, these implementations in a parallel environment can bring the model closer to reality where perfusion processes are inherently parallel.

The two previously proposed parallel algorithms are based on message passing paradigm [17] and therefore are perfectly suited for distributed memory architectures. Both algorithms use the master-slave model [5], it means that master/managing processor/node generates tasks and distributes them among slave/calculating processors.

The general scheme of the first algorithm [15] is presented in Fig. 2. It parallelizes the perfusion process. The remaining processes (i.e. necrosis, retraction and shape growth) are performed sequentially at the master node. In that case, before each perfusion phase, slave nodes do not possess the most current vascular system and tissue. Thus, at the beginning of each subcycle the master node has to broadcast the latest MFUs and vascular trees. Subsequently, after the sequential mitosis, the parallel perfusion is carried out. In comparison to its sequential version, here the managing node does not make any attempt to find candidate vessels and bifurcation points but instead it spreads these tasks over calculating nodes.

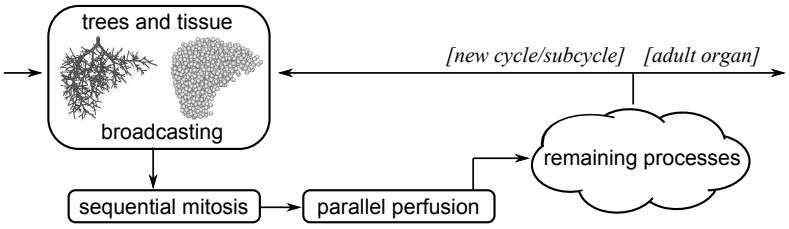


**Fig. 2.** The outline of the first parallel algorithm of vascular growth. At the beginning of each cycle/subcycle, the trees and tissue broadcasting is performed. Next, the sequential mitosis, parallel perfusion and remaining vascular processes (e.g. necrosis, retraction) are carried out in turn. The algorithm ends when the organ reaches its adult form.

When a computational node receives the message with several MFUs, it attempts to find the closest vessels and finally the optimal bifurcation points to perfuse these new tissue elements. Each time, when the search ends with success, the parameters of the optimal bifurcation are sent to the master node. Next, if there are any queued messages with permanent changes in vascular network sent by master node, the slave node applies these changes and continues to perform its remaining tasks.

The master node manages the perfusion process. It is responsible for gathering messages coming from the slave nodes and making decisions about the permanent perfusions. When it receives a message with optimal bifurcation parameters of one of the new MFUs, it has to check if this MFU can be connected to the current vascular network. A rejection is possible because vascular networks at individuals nodes (both at computational ones and managing one) can be slightly different (trees' nonuniformity) as a result of communication latency and independent work of slave nodes. Therefore, the master processor tries to find in its vasculature the vessels related with the proposed optimal bifurcation. If the processor cannot find at least one of these vessels, then the MFU is rejected. But in the other case, the new MFU is permanently connected to the vascular network and all organ changes related with the new tissue element are broadcasted to the slave processors.

However, we found that the efficiency of this algorithm can decrease in the case of a huge number of vessels (i.e. tens of thousands). The reason is related to the periodical broadcasting of the whole organ. We minimized the message size and only the parameters that cannot be reconstructed by slave nodes are sent. Moreover, many initial parameters are read from input files. As a result, the time to send the packed messages is insignificant. But, unfortunately, it turned out that the time needed to reconstruct vascular trees from the received packed messages by slave processors can be responsible for slowing down the algorithm.

Therefore, we also proposed an improved parallel algorithm [16]. Its general diagram is presented in Fig. 3. Each node during the whole simulation has its own copy of vascular trees and tissue. Thus, only at the beginning, the master node broadcasts the whole initial organ to ensure that all the nodes possess the same starting information. Each new subcycle starts with the sequential mitosis. Next, the perfusion process is carried out in parallel. Slave nodes attempt to find optimal bifurcations points, while the master node is responsible for managing the process of permanent perfusion and broadcasting changes.

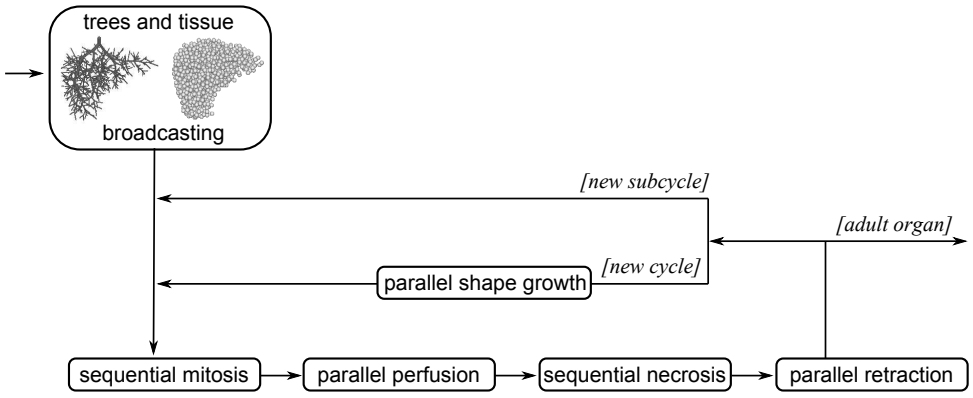


**Fig. 3.** The outline of the improved parallel algorithm of vascular growth. Only at the beginning, the trees and tissue broadcasting is performed. Next, the sequential mitosis, parallel perfusion, sequential necrosis and parallel retraction are carried out in turn. Then, in the case of new cycle, the parallel shape growth phase comes and algorithm returns to the sequential mitosis. In the case of new subcycle, the algorithm returns directly to the sequential mitosis. The algorithm ends when the organ reaches its adult form.

After the reproduction process, the degeneration phase follows. At the master node, the sequential necrosis is performed. Due to giving up broadcasting the whole organ in each subcycle, all the slave nodes have to be informed about possible necrosis changes. Therefore, the master node broadcasts to all other nodes information about the MFUs that have to be removed. The entire algorithm of retraction is performed at each node simultaneously. If the shape growth phase is needed, it is also carried out simultaneously at each node. The performance analysis showed that the time needed for these parts of the algorithm can be neglected, as it is very short in comparison to the perfusion time.

## 3. Load Balancing Mechanisms in Parallel Algorithm of Vascular Network Development

One of the most important issues in parallel computing is load balancing. It aims at roughly equal workload arrangement across processors and minimization of their idle time. Such an arrangement typically improves the performance and increases the efficiency of parallel applications, which reduces the run time of computations. Obviously, we can also improve the performance by increasing power of processors or by delivering more processors. Nevertheless, this expensive way of achieving the goal is not often able to increase the efficiency and usually should be used in the cases in which all processing units are overloaded or there is no possibility of an equal load distribution.

However, in many studies it has been shown that, even when the tasks are strongly linked with each other and their workloads are totally unpredictable, load balancing algorithms can be very useful [6]. On the other hand, one has to be careful to avoid that the cost of load balancing exceeds its possible benefits, which would decrease the overall performance.

In the next part of this section, we propose several load balancing mechanisms in the parallel algorithm of vascular network development. Firstly, the load balancing mechanisms across slave processors are presented and then we also describe how to efficiently load a master processor.

### 3.1 Load Balancing Across Slave Processors

In the proposed parallel algorithms we focus mainly on the perfusion process as it is the most time demanding phase of the vascular growth simulation. This process is decomposed into a set of tasks that solve the problem in parallel. A single task consists in finding a fixed number of candidate/nearest vessels and then optimal bifurcation points for a single MFU. In order to find the nearest vessels, the whole vascular network has to be searched. The time needed to perform this operation can differ for successive MFUs because of changes in vascular tree structures: new branches (i.e. vessel segments) can appear and old ones can disappear. Thus, it is very hard to estimate the time of this operation before the work distribution because one does not know a priori how the vascular system will look after each next permanent perfusion.

Subsequently, for all candidate vessels optimal bifurcation points are calculated. This operation is the most time consuming part of the perfusion process since it takes approximately 60-90% of the time needed to this process. In order to find

the position of bifurcation that minimizes a local volume of blood the Downhill Simplex algorithm is used [23], [24]. Again, we do not know how much time this searching can take because number of steps necessary to find the local minimum is hard to precise, even in the case of invariable structures of the vascular system. Moreover, for different MFUs the number of candidate vessels to be processed can be various since some of these vessels may be rejected due to their neighboring vascular structures preventing the creation of any bifurcation (e.g. lack of free space in the neighborhood).

The last stage of searching the optimal bifurcation points is the selection of one of the candidate vessels from each vascular tree in such a way that the global blood volume (i.e. for the whole vascular network) is minimal. Moreover, the algorithm detects all possible intersections between the perfusing vessels (vessels constituting the new bifurcation) in the same tree and two different trees (e.g. between arteries and veins) and rejects the related candidate vessels. Also in this case, it is impossible to predict the number of steps. This phase can end after checking the first combination of candidate vessels as well as after checking the last one.

To sum up the above general description of operations that have to be done in each task, we can state that the work required to find the optimal bifurcation points can vary for different MFUs. Moreover, it is impossible to approximate the time needed to perform each task before the work distribution as well as immediately before its execution. Thus, it is very hard to find any algorithm able to precisely decide when, where and how much work has to be assigned. In addition, we have to deal with the small grain parallelism (in one subcycle, the number of tasks can come to several thousands) and use of any sophisticated (i.e. computationally extensive) load balance algorithm can introduce an overhead that may exceed possible benefits. Therefore, we decided to propose a mechanism that is based on the basic centralized dynamic load balance algorithm [5].

After the sequential mitosis, the master node holds the collection of tasks, i.e. new MFUs to perfuse. At the beginning, a fixed part of these tasks is spread between processors (part A in Fig. 4 - 1st load balancing mechanism). Each slave processor receives approximately the same number of jobs. The master processor keeps the rest of new MFUs that will be assigned to slave nodes only on demand. When a slave node finishes its jobs, it sends a request to the master node in order to get more work (unbalance load detection). If the master node still has MFUs to be checked, it sends part of these MFUs to the under-loaded node (operation 5, part B in Fig. 4 - 1st load balancing mechanism). The number of MFUs to send is calculated according to the

following formula:

$$\frac{number\_of\_the\_remaining\_MFUs}{number\_of\_processors} + 1 \,. \tag{4}$$

This mechanism detects unbalance load dynamically and transfers tasks to idle processors. The decision on how many new MFUs are distributed immediately after the mitosis is made once at the beginning of the simulation. The great advantage of this mechanism is that it is simple for the master node to know when to terminate. In our case, the perfusion process ends when: the task queue is empty, all permanent perfusions are broadcasted across slaves nodes and all slave nodes have finished their jobs.

When a slave node finishes a single task, it sends the parameters of optimal bifurcation points to the master node (operation 2, part B in Fig. 4). Next, if there are any queued messages with permanent vascular tree changes broadcasted by the master node, the slave node applies these changes and continues to perform its remaining tasks. Such a solution decreases idle time of slave nodes because they can perform calculations without any break to wait for a response from the master node. On the other hand, due to continuous work of slave nodes (i.e. without waiting for a response whether the proposed optimal bifurcation points can be used for permanent perfusion) the trees' nonuniformity can increase. If the trees' nonuniformity increases then the possibility of MFU rejection by the master node also rises, which can cause that the time of simulation is longer. As a result, a tradeoff between these two approaches has to be found. We decided that if the number of computational nodes is quite small, the nodes work continuously. Otherwise, the nodes try to minimize the trees' nonuniformity and wait for a response (part B in Fig. 4 - 2nd load balancing mechanism).

Another crucial point in the algorithm is sending/broadcasting permanent vascular changes by the master node (operation 3, part B in Fig. 4). If one wants to minimize the time of communication between nodes, then the best solution is to send a set of changes (not a single change each time that this change appears). However, such an approach can increase the trees' nonuniformity. Thus, we made a decision to synchronize this mechanism with the mechanism described above. When computational nodes work continuously between successive MFUs (non-blocking receive of changes) then the changes are collected by the master node and sent as a set of changes (part B in Fig. 4 - 3rd load balancing mechanism). On the other hand, when the computational nodes wait for a response after each MFU (after each task) then the master node broadcasts changes as quick as possible (i.e. when they appear).

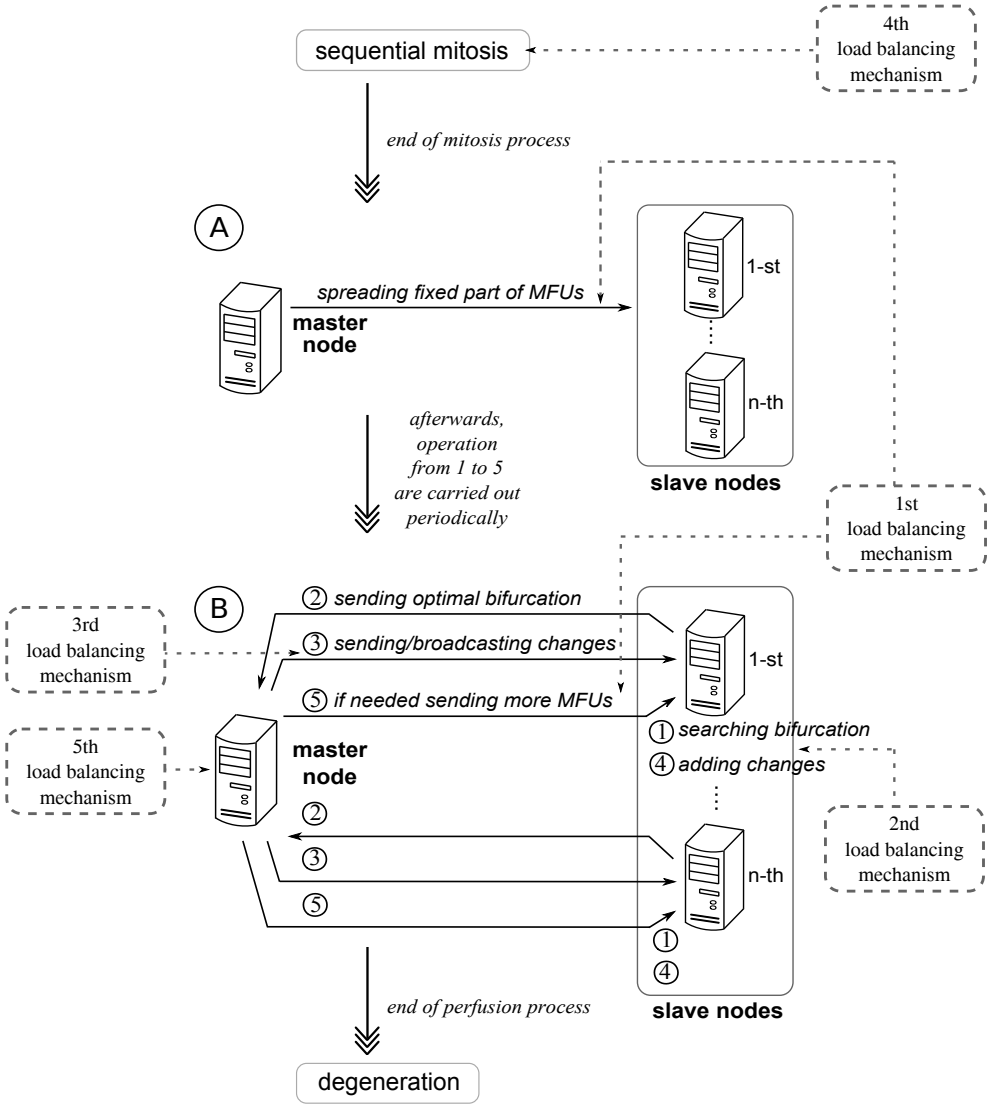


**Fig. 4.** Load balancing mechanisms in the parallel algorithm of vascular growth. The first load balancing mechanism is related with spreading only a fixed part of new MFUs and sending the remaining MFUs on demand. The second and third load balancing mechanisms are responsible for an appropriate organization of sending and receiving permanent vascular changes. The fourth one enables us to start the perfusion process even before the end of mitosis process. The fifth one tries to efficiently load the master processor in the case of a small number of slave nodes. Part A concerns the work distribution before the perfusion process, while part B illustrates the perfusion process during which slave nodes search optimal bifurcations and the master node manages permanent perfusions and broadcasts changes.

The next load balancing mechanism that we want to bring in concerns the mitosis process (see Fig. 4 - 4th load balancing mechanism). When the master node performs the algorithm of new MFU creation all slave nodes are idle. Therefore, we introduced the possibility that some MFUs can be spread even before the end of the mitosis process. In this case, slave nodes can start their work quicker.

### 3.2 Efficient Load of Master Processor

In the standard centralized dynamic load balancing algorithm [1], the master processor is responsible only for the managing of task distribution. However, in order to provide still more efficient solution, in the presented algorithm, the master processor can also perform calculations related to finding parameters of optimal bifurcations, i.e. the same as slave processors (see Fig. 4 - 5th load balancing mechanism). This mechanism can be particulary useful in the case of a small number of slave nodes since the master node can also have time to do additional calculations besides managing the perfusion process.

## 4. Experimental Validation

This section contains an experimental verification of the proposed load balancing mechanisms. The presented results were obtained in many experiments. We tested the behavior of the vascular model starting from small size configurations (about 1000 MFUs) and ending with large size configurations (about 50000 MFUs and consequently about 300000 vessel segments). In Fig. 5 a visualization of one of the obtained vascular network of a liver is presented. Typical physiological parameters of the hepatic vascular network were used [12]. At the beginning, the efficiency of proposed load balancing mechanisms is evaluated using the speedup and next the detailed results of the particular mechanisms are presented.

In the experiments a cluster of sixteen SMP servers running Linux 2.6 and connected by an Infiniband network was used. Each server was equipped with two 64-bit Xeon 3.2GHz CPUs with 2MB L2 cache, 2GB of RAM and an Infiniband 10GB/s HCA connected to a PCI-Express port. We used the MVAPICH version 0.9.5 [25] as the MPI standard implementation [17]. Moreover, we carried out the experiments on a similar cluster of sixteen SMP servers but each server was equipped with eight processing units and the MVAPICH version 1.0.3. In order to execute the performance analysis we used the Multi-Processing Environment (MPE) library with the graphical visualization tool Jumpshot-4 [17] and Tuning and Analysis Utilities (TAU) Performance System [26].

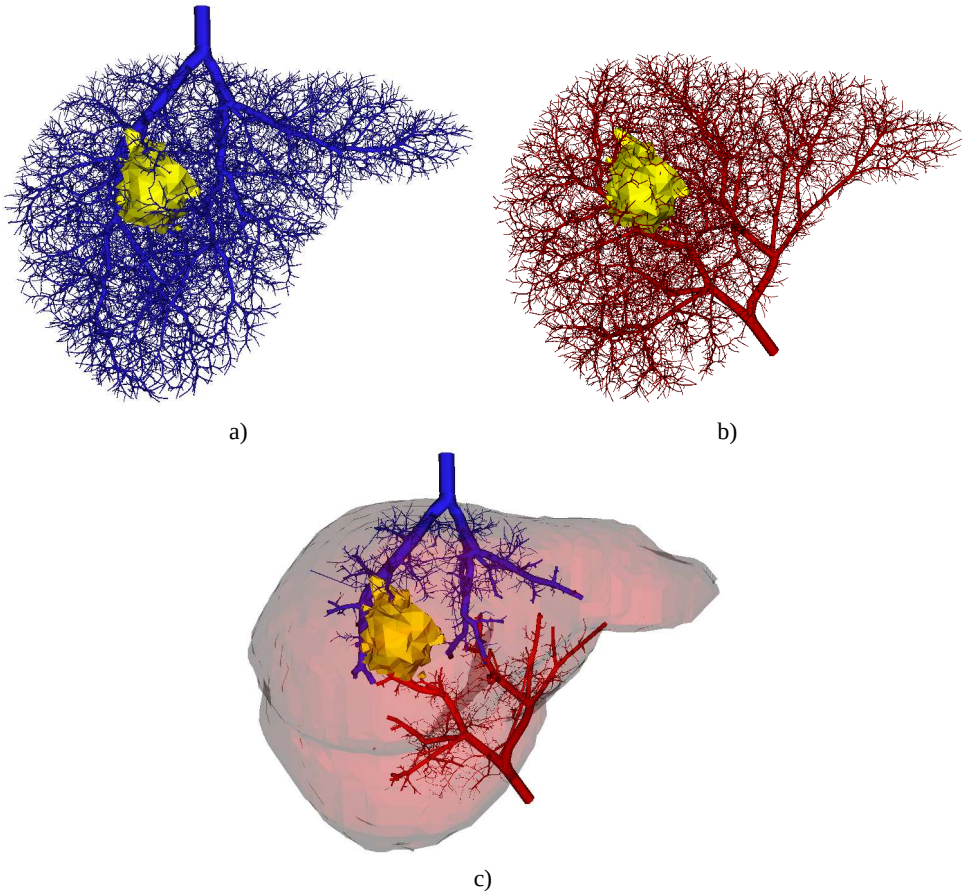


**Fig. 5.** Visualization of an adult liver (about 49000 MFUs and 300000 vessels): a) hepatic veins with a tumor shape, b) hepatic arteries with a tumor shape, c) main hepatic arteries, portal veins and hepatic veins with liver and tumor shapes.

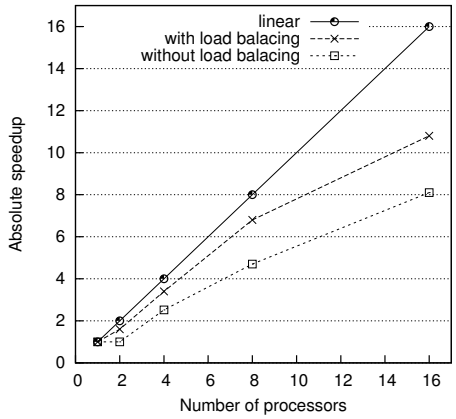


**Fig. 6.** Mean speedups of the parallel algorithm with and without load balancing mechanisms from experiments for different vascular network size (from several thousands to several hundreds thousands of vessel segments).

In Fig. 6 one can see the obtained average speedups of algorithms with and without load balancing mechanisms. It is clearly visible that the solution with load balancing mechanisms is from 20% to 25% quicker than the solution without this mechanisms. As a result, the simulation time on sixteen CPUs needed to receive the adult organ consisting of about 50000 MFUs and 300000 vessel segments equals approximately 2 hours with load balancing, instead of 3 hours without load balancing or 23 hours on a single processor machine (64-bit Xeon 3.2GHz with 2MB L2 cache, 2GB of RAM).

The following figures present in more detailed the influence of the load balancing algorithms. The mean time results from the simulation on eight CPUs (seven slave processors and one master processor) with large size vascular network (about 48000 MFUs) are shown in Fig. 7a . One can see the idle+communication time for particular slave processors with and without load balancing mechanisms. It is clearly visible that without load balancing mechanisms the processors waste more time for waiting and communication, which can be caused by a higher number of rejected MFUs by the master processor and consequently the necessity to test more new MFUs. Moreover, particular slave processors are unevenly loaded. On the other hand, we observe that with load balancing mechanisms the slave nodes waste approximately the same amount of time (i.e. are evenly loaded). In addition, the total idle time is shorter.

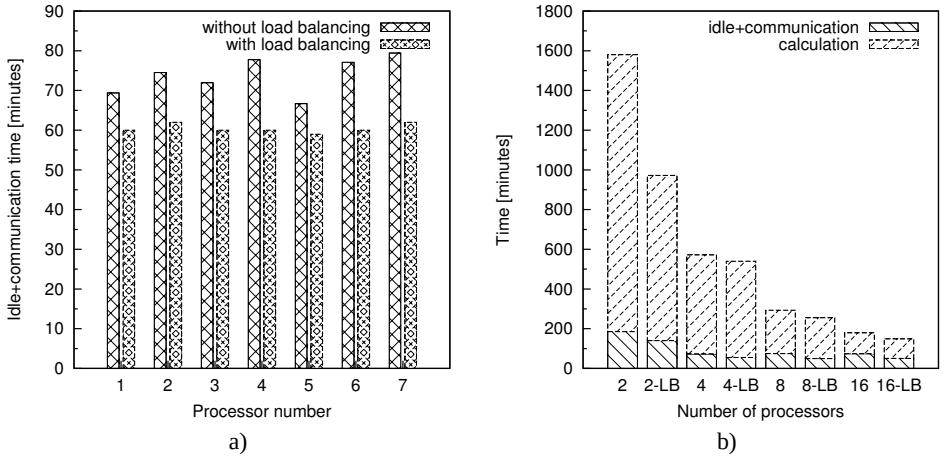


**Fig. 7.** The influence of load balancing mechanisms: a) mean idle+communication time for particular slave processors from experiments on eight CPUs with large size vascular network with and without load balancing mechanisms, b) mean idle+communication and calculation times for a different number of processors with large size vascular network with and without load balancing mechanisms.

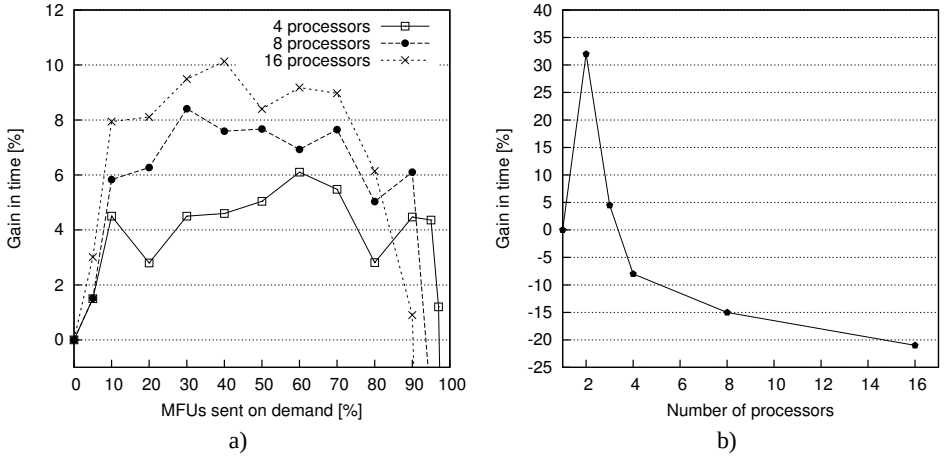


**Fig. 8.** The influence of load balancing mechanisms: a) related to a fixed part of MFUs that are kept by the master processor and sent only on demand to under-loaded processors, b) related to additional tasks that are assigned to the master processor.

Fig. 7b presents the mean idle+communication and calculation times obtained in experiments with large size vascular network (about 48000 MFUs) for a different number of processors without and with load balancing (LB) mechanisms. We can see that each time when the load balancing mechanisms are used both calculation and idle+communication times are shorter. The idle+communication time is shorter mainly because of keeping by the master node the fixed part of new MFUs that are sent on demand to under-loaded processors. While, the calculation time is shorter as a result of an appropriate organization of sending and receiving changes, which decreases the trees' nonuniformity. Based on many experiments, we suggest that if the number of processors is smaller than eight, the slave processors should work continuously between successive MFUs and the master node collects changes and sends them in groups. Otherwise, the slave processors wait for a response after each MFU and the master node broadcasts changes as quick as possible. Obviously, the found limit is suitable for the used clusters and may vary for other hardware.

Moreover, we thoroughly investigated the load balancing mechanism consisting in keeping by the master node a fixed part of new MFUs which are sent on demand to under-loaded processors (see Fig. 8a). From one point of view, we can see that it is very hard to choose one common value (number of MFUs) that gives the best gain in time for a different number of processors. On the other hand, it is visible that values higher than 90% can increase the simulation time. Finally, we suggest that any value within the range from 30% to 70% is acceptable.

Furthermore, we tested in which cases it is worth to involve the master processor also in calculations connected with finding optimal bifurcations points (see Fig. 8b). It is clearly visible that in the case of small number of processors (i.e. smaller than four) if the master node, besides managing, performs the same calculations as slave processors the simulation time can be reduced. On the other hand, i.e. the number of processor is bigger than three, we should not arrange any additional job to the master node.

## 5. Conclusion and Future Works

In this paper we propose several mechanisms with the aim to balance workload across processors and to reduce the communication overhead in the parallel algorithm of vascular network development. We consider a master-slave model in which tasks appear at the central scheduler (master processor) and are distributed between slave processors. Thus, the core mechanism is based on the centralized dynamic load balancing algorithm that detects an unbalance load dynamically and tries to send more job to under-loaded processors. Moreover, we investigate the influence of trees'

nonuniformity between particular calculation nodes and managing node on simulation time. In consequence, we found the tradeoff between communication overhead and processors' idle time. The proposed mechanisms were thoroughly validated in many experiments. The results have shown that the introduced improvements are able to further accelerate the process of vascular growth simulation. As a result, the simulation time even when we introduce more physiological details to the model or increase the number of MFUs can be done still in a reasonable period of time. In addition, it is easier to perform multiple experiments in order to calibrate model parameters.

In the future, we plan to pay more attention to parallel computing in shared-memory environments. We want to implement the vascular growth process in the framework of multi-platform shared-memory parallel programming (OpenMP) using a fine-grained parallelism. Moreover, our goal is to develop an hybrid solution able to take advantage of machines with both shared and distributed memory architectures (MPI+OpenMP implementation).

## References

[1] Wilkinson, B., Allen, M.: Parallel Programming, Techniques and Applications Using Networked Workstation and Parallel Computers, Second Edition, Prentice Hall, 2005.

[2] Scott, L.R., Clark, T., Bagheri, B.: Scientific Parallel Computing, Princeton University Press, 2005.

[3] Maton, A.: Human Biology and Health, Third Edition, Pearson Prentice Hall, 1997.

[4] Zeigler, B.P., Praehofer, H., Kim, T.G.: Theory of Modeling and Simulation, Academic Press, 2000.

[5] Grama, A., Karypis, G., Kumar, V., Gupta, A.: Introduction to Parallel Computing, Addison-Wesley, 2003.

[6] Shirazi, B.A., Kavi, K.M., Hurson, A.R.: Scheduling and Load Balancing in Parallel and Distributed Systems, IEEE Computer Society Press, 1995.

[7] Bokhari, S. H.: On the mapping problem, IEEE Trans. Comput. 30(3), 1981, pp. 207-214.

[8] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing, Science 220, 1983, pp. 671-680.

[9] Lee, S.Y., Lee, K.G.: Synchronous and asynchronous parallel simulated annealing with multiple markov chains, IEEE Trans. Parallel and Distributed Systems 7, 1996, pp. 993-1008.

[10] Mani, V., Suresh, S., Kim, H.J.: Real-coded genetic algorithms for optimal static load balancing in distributed computing system with communication delays, Lecture Notes in Computer Science 3483, 2005, pp. 269-279.

[11] Osman, A., Ammar, H.: Dynamic load balancing strategies for parallel computers, Scientific Annals of Computer Science Journal of Cuza University 11, 2002, pp. 110-120.

[12] Krętowski, 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. on Medical Imaging 22(2), 2003, pp. 248-257.

[13] Kretowski, M., Bezy-Wendling, J., Coupe, P.: Simulation of biphasic CT findings in hepatic cellular carcinoma by a two-level physiological model, IEEE Trans. Biomed. Eng. 54(3), 2007, pp. 538-542.

[14] Mescam, M., Kretowski, M., Bezy-Wendling, J.: Multiscale model of liver DCE-MRI towards a better understanding of tumor complexity, IEEE Trans. on Medical Imaging 29(3), 2010, pp. 699-707.

[15] Jurczuk, K., Kretowski M.: Parallel implementation of vascular network modeling, Lecture Notes in Computer Science 5101, 2008, pp. 679-688.

[16] Jurczuk, K., Kretowski M., Bezy-Wendling J.: Vascular network modeling – improved parallel implementation on computing cluster, Lecture Notes in Computer Science 6067, 2010, pp. 289-298.

[17] Pacheco, P.: Parallel Programming with MPI, Morgan Kaufmann Publishers, 1997.

[18] Mescam, M., Eliat, P.A., Fauvel, C., De Certaines, J.D., Bezy-Wendling, J.: A physiologically-based pharmacokinetic model of vascular-extravascular exchanges during liver carcinogenesis: Application to MRI contrast agents, Contrast Media Molecular Imag. 2(5), 2007, pp. 215–228.

[19] Sherlock, S., Dooley, J.: Diseases of the Liver and Biliary System, Blackwell Science, 2002.

[20] Zamir, M., Chee, H.: Branching characteristics of human coronary arteries, Can. J. Physiol. Pharmacol. 64, 1986, pp. 661-668.

[21] Kamiya, A., Togawa, T.: Optimal branching structure of the vascular trees, Bulletin of Mathematical Biophysics 34, 1972, pp. 431-438.

[22] Goss, R.J.: The strategy of growth in Control of Cellular Growth in Adult Organisms, Teir, H., Tapio R., Eds. Academic Press, 1967, pp. 3-27.

[23] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes in C. The art of scientific computing, Cambridge University Press, 1992.

[24] Kretowski, M., Rolland, Y., Bezy-Wendling, J., Coatrieux, J.-L.: Fast 3D modeling of vascular trees, Computer Methods and Programs in Biomedicine 70(2), 2003, pp. 129-136.
[25] MVAPICH: MPI over InfiniBand and iWARP, `http://mvapich.cse.ohio-state.edu/`
[26] Shende, S., Malony, A.D.: The TAU parallel performance system, International Journal of High Performance Computing Applications 20(2), 2006, pp. 287-311.

# MECHANIZM ZRÓWNOWAŻENIA OBCIĄŻENIA W RÓWNOLEGŁEJ IMPLEMENTACJI ROZWOJU SIECI NACZYŃ KRWIONOŚNYCH

**Streszczenie** W artykule rozważane są mechanizmy zrównoważające obciążenie w równoległym algorytmie rozwoju sieci naczyń krwionośnych. Główną uwagę zwrócono na proces perfuzji (podłączanie nowych komórek do drzew krwionośnych) jako, że proces ten jest najbardziej czasochłonnym fragmentem rozpatrywanego algorytmu. Zaproponowane przez autorów rozwiązania mają na celu zrównoważenie obciążenia pomiędzy procesorami, skrócenie ich czasu bezczynności oraz zredukowanie narzutu komunikacyjnego. Jądro rozwiązania jest oparte na scentralizowanym dynamicznym podejściu równoważenia obciążenia. Zachowania modelu zostały przeanalizowane i kompromis pomiędzy różnymi technikami został zaproponowany. Przedstawione mechanizmy zostały zaimplementowane na klastrze obliczeniowym przy wykorzystaniu standardu MPI. Otrzymane rezultaty jednoznacznie pokazują iż wprowadzone usprawnienia zapewniają bardziej efektywne rozwiązanie co w konsekwencji pozwala na jeszcze większe przyśpieszenie procesu symulacji.

**Słowa kluczowe:** algorytmy równoległe, mechanizmy równoważenia obciążenia, klastry obliczeniowe, modelowanie komputerowe, system krwionośny

Artykuł zrealizowano w ramach pracy badawczej W/WI/3/2010.