High performance distributed simulation for interactive simulated vascular reconstruction

Robert G. Belleman and Roman Shulakov

Section Computational Science, Faculty of Science, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, the Netherlands.
(robbel|rshulako)@science.uva.nl

Abstract. Interactive distributed simulation environments consist of interconnected communicating components. The performance of such a system is determined by the execution time of the executing components and the amount of data that is exchanged between components. We describe an interactive distributed simulation system in the scope of a medical test case (simulated vascular reconstruction) and present a number of techniques to improve performance.

1 Introduction

Interactive simulation environments are dynamic systems that combine simulation, data presentation and interaction capabilities that together allow users to explore the results of computer simulation processes and influence the course of these simulations at run-time [4] (see also Fig. 1). The goal of these interactive environments is to shorten experimental cycles, decrease the cost of system resources and enhance the researcher's abilities for the exploration of data sets or problem spaces.

In a dynamic environment, the information presented to the user is regenerated periodically by the simulation process. The environment is expected to provide (1) a reliable and consistent representation of the results of the simulation at that moment and (2) mechanisms enabling the user to change parameters

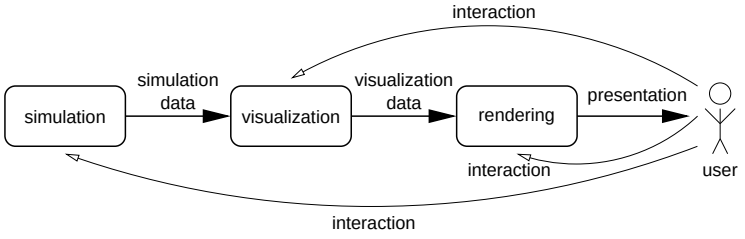

Fig. 1. Interactive simulation environments consist of a simulation, visualization and rendering component with which a user interacts to interactively explore data sets or problem spaces.

in the environment. An example of an application where this structure is used has previously been described in [5]. This test case environment allows vascular reconstruction procedures to be simulated using a fluid flow simulator that simulates the effect of a planned surgical procedure on a patient's blood circulation. An interactive immersive virtual reality environment allows a surgeon to study the effect of a reconstructive procedure and interactively explore alternatives for the best possible treatment for a specific patient.

1.1 Performance of interactive simulation environments

The most important factor in the performance of a dynamic simulation environment is *update time*; the delay between consecutive updates in the environment. Usability increases when update time is as short as possible, so for a highly responsive environment, delays should be minimized. Fig. 2 shows a schematic representation of the most typical delay imposing factors in a simple dynamic simulation environment.

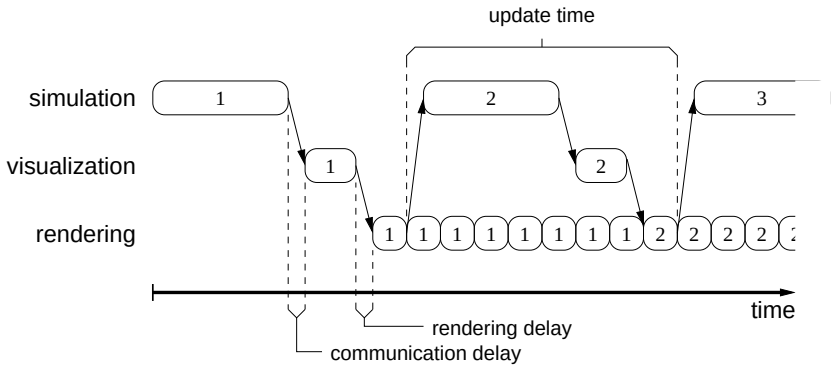

Fig. 2. Typical delays in a simple dynamic simulation environment.

The main factors on delay in interactive environments are the execution times of the components and the delay caused by the communication of data from one component to the next. This paper describes a number of methods to improve the performance of interactive simulation environments.

The ideas and methods in this paper have been applied to a test case, that of the simulated vascular reconstruction test case described in [5] which will be briefly described in section 2. Section 3 focuses on methods to maximize the performance of an interactive simulation environment. Section 4 presents some of the results obtained thus far. Finally, conclusions and future work are described in section 5.

2 Test case: interactive simulated vascular reconstruction in a virtual operating theater

The performance considerations described in the previous section are validated by analysis of a prototypical case study of *simulated vascular reconstruction*. This application combines simulation, visualization and interaction in an exemplary fashion. By a detailed analysis of the spatial and temporal characteristics of this test case we attempt to recognize generic elements for the design of an interactive simulation environment. Before we describe the methods that have been implemented to obtain a high performance interactive simulation environment, we begin with a description of the test case.

2.1 Introduction: interactive simulated vascular reconstruction

The purpose of vascular reconstruction is to redirect and augment blood flow or perhaps repair a weakened or aneurysmal vessel through a surgical procedure. The optimal procedure is often obvious but this is not always the case, for example, in a patient with complicated or multi-level disease. Pre-operative surgical planning will allow evaluation of different procedures *a priori*, under various physiological states such as rest and exercise, thereby increasing the chances of a positive outcome for the patient. The aim of this case study is to provide a surgeon with an environment in which he/she can explore the effect of a number of different vascular reconstruction procedures before it is put to practice. Our approach combines parallel flow simulation, interactive virtual reality and high performance computing and networking techniques into an interactive dynamic exploration environment that together allows human-in-the-loop types of experimentation [5].

2.2 Blood flow simulation: the lattice-Boltzmann method

The lattice-Boltzmann method (LBM) is a mesoscopic approach for simulating fluid flow based on the kinetic Boltzmann equation [7]. In this method fluid is modeled by particles moving on a regular lattice. At each time step, particles propagate to neighboring lattice points and re-distribute their velocities in a local collision phase. This inherent spatial and temporal locality of the update rules makes this method ideal for parallel computing [11]. During recent years, LBM has been successfully used for simulating many complex fluid-dynamical problems, such as suspension flows, multi-phase flows, and fluid flow in porous media [13]. All these problems are quite difficult to simulate by conventional methods [10, 12].

The data structures required by LBM (Cartesian grids) bare a great resemblance to the grids that come out of CT and MRI scanners. As a result, the amount of preprocessing can be kept to a minimum which reduces the risk of introducing errors due to data structure conversions. LBM has the benefit over other fluid flow simulation methods that flow around (or through) irregular geometries (like a vascular structure) can be simulated relatively easy. In addition,

velocity fields, pressure and shear stress on the arteries can be calculated directly from the densities of the particle distributions [2]. This may be beneficial in cases where we want to interfere with the simulation while the velocity and the stress field are still developing, thus supporting fast data-updating given a proposed change in simulation parameters as a result of user interaction.

2.3 High performance interactive visualization

The simulated vascular reconstruction operating theater provides visualization and interaction methods to simulate a vascular reconstruction procedure and visualize the effect of that procedure on a patient's blood circulation in real time. The environment can be used in a CAVE [8] but also on low cost commodity hardware in conjunction with a projection display and tracking hardware [6]. Multi-modal interaction methods such as speech recognition, hand gestures, direct manipulation of virtual 3D objects and measurement tools allow researchers to explore simulation results [3].

3 High performance interactive simulation

The responsiveness of an interactive system is directly related to the rate at which updates are generated by each of the components in the system. To increase responsiveness, the delays between the consecutive components in the interactive system should be minimized. The accumulation of all delays is referred to as "update time". In an interactive system there will always be some delay from the moment interaction takes place until the moment that the environment has reacted to this interaction. This delay is referred to as "response time".

3.1 Update and response time

In a non-interactive environment, the update time T_U is the sum over the execution time for the different components (T_{sim} for simulation, T_{vis} for visualization and T_{ren} for rendering) and the communication delay between components ($T_{sim \to vis}$ and $T_{vis \to ren}$):

$$T_U = T_{sim} + T_{sim \to vis} + T_{vis} + T_{vis \to ren} + T_{ren}. \quad (1)$$

Decreasing update time means that the delays imposed by the different components must be minimized. In the case of executing components this means that the time between the acceptance of input data and the production of results should be minimized. In the case of communication between components, the dominating factor on delay is the time that is required to transfer data from one component to the next.

In an interactive system, the response time T_R depends on which component the interaction is directed to since only this and subsequent components need to be updated. In case the user interacts with the simulation component, the response time will be $T_R = T_{i(sim)} + T_U$, where $T_{i(sim)}$ is the time required to "apply" the interaction to the simulation component.

3.2 Distributed pipelined execution

A dynamic system differs from a static system in that the simulation component is an iterative process that repeatedly produces (intermediate) results. Basically, the delay at which these intermediate results become available is given by equation (1). However, since the simulation component has no dependency on the execution of subsequent components, the update time of the whole environment can benefit from a pipelined execution model. In this model, a component resumes execution as soon as its output data has been accepted by the next (see Fig. 3). In this case simulation, visualization and rendering execute in parallel.

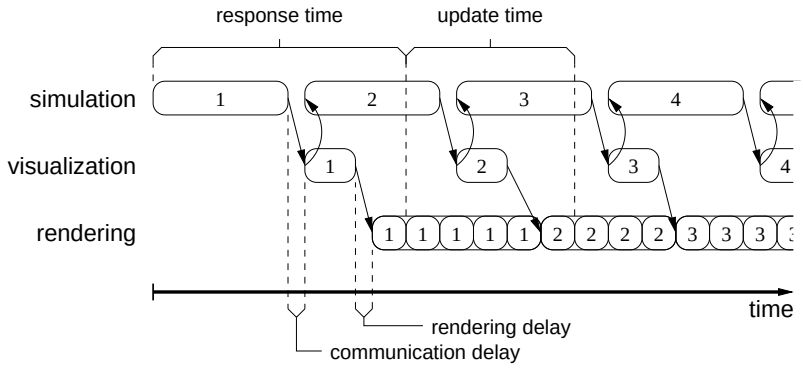

Fig. 3. Response time, update time and delays in a pipelined interactive simulation environment.

Once all components in the pipeline have executed at least once, and provided sufficient resources are available to execute all components simultaneously, the update time becomes

$$T_U = max(T_{sim} + T_{sim \rightarrow vis}, T_{vis} + T_{vis \rightarrow ren}, T_{ren}). \quad (2)$$

Although the components in Fig. 1 show a close coupling between them, it is not necessary, or even beneficial, to execute all on the same computing system. It is often more efficient to execute the components on systems that have optimized resources available for the most time consuming type of operations. The flow simulation component described in section 2, for example, runs more efficient on a parallel system, while the visualization component performs better on a system with optimized visualization libraries and the rendering component performs better on a system with specialized graphics hardware on board.

3.3 High performance network communication

Distributing components over different systems means that some form of communication must be established to allow the output of one component to be

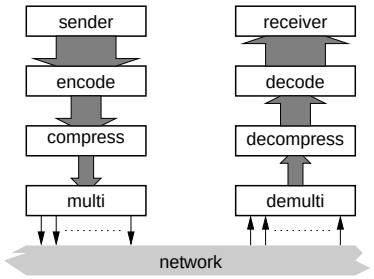

Fig. 4. The stages in the communication pipeline.

transferred to the next. It could be that the overhead generated by this communication mechanism annuls the benefits obtained by the distribution (i.e. although T_{c1} decreases through the use of optimized resources, $T_{c1 \rightarrow c2}$ increases because of the extra communication overhead between two components $c1$ and $c2$). To reduce the delays caused by communication it may be beneficial to reduce the amount of transferred data as much as possible. This reduction itself, however, also takes time so careful consideration is often necessary.

We have implemented three mechanisms to increase the throughput of data transfers over a network. These mechanisms are cascaded into a pipeline to decrease the amount of transferred data and spread the remaining data over multiple network connections, in an effort to maximize throughput (see Fig. 4).

Hiding latency by using multiple connections. This method increases communication throughput by using multiple network connections between peers at the same time [9]. There are two reasons why this increases throughput (see also Fig. 5). First; in the case of reliable network connections (such as connections using the TCP protocol), delays caused by waiting for acknowledgment packets can be "hidden" by serving other connections that are ready. Second; the technique can exploit "intelligent" network devices that spread communication over different routes. This technique therefore performs best when there are many such devices between peers (which is often the case when peers are geographically dispersed). In principle, data can travel along different routes from peer to peer, thereby circumventing congestion caused by other traffic on the network.

Note that the total volume of data that is transferred between peers is not decreased by this method. Instead, it increases throughput by exploiting as much available bandwidth as possible. The techniques described next aim to increase throughput by decreasing the volume of communicated data.

Data encoding. Data encoding is a data specific type of data reduction technique that aims to reduce the volume that is needed to represent the data to a minimum. This technique relies on the fact that the receiving side may not always be interested in the most accurate representation of the data that was

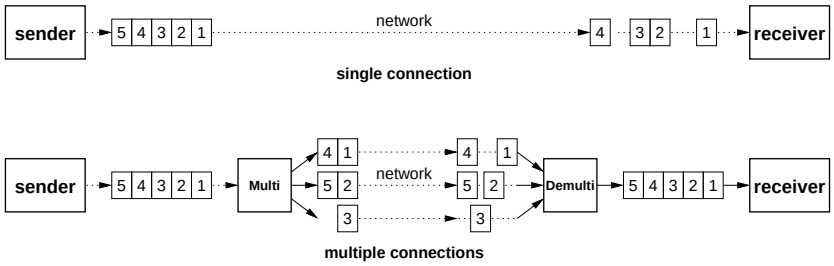

Fig. 5. Increasing communication throughput using multiple network connections.

calculated by the sender. Because this type of data reduction throws away unnecessary information, this method is frequently referred to as *lossy compression*.

Although a significant reduction in volume can be achieved using this technique, the receiver should be conscious of the fact that the data it has received is not of the same accuracy as was originally produced. Although this reduction may be acceptable under some circumstances, unexpected side effects may occur when the errors that are present in the data are accumulated due to an integrating method of analysis on the data. For example, an often used technique in vector field visualization is to represent the path of the flow using streamlines. These streamlines are created by integrating over individual vectors. Due to the accumulation of (small) errors in each individual vector, a streamline may follow a radically different path.

Data compression. Freely available compression libraries (such as *zlib* [1]) provide means of reducing data volumes very effectively. This data reduction does not make any assumption about the data contents and is therefore without any loss of information. Most compression libraries can be parameterized to indicate the level of compression that should be achieved at the expense of higher execution time. This type of data reduction is commonly referred to as *lossless compression* as the data is unchanged after decompression. The amount of compression depends on (1) the type of data and (2) the amount of data. In general, the compression ratio decreases when the amount of data decreases. Note that this is important in parallelized applications in which the original data volume is often decomposed into smaller subvolumes.

4 Results

The simulated vascular reconstruction environment described in section 2 has been implemented using the methods described in section 3. In this section we show some preliminary results on the performance of the environment.

4.1 Performance of the distributed simulation pipeline

Fig. 6 illustrates the performance increase that is achieved by using a parallelized implementation of the flow simulation kernel. Because the test case en-

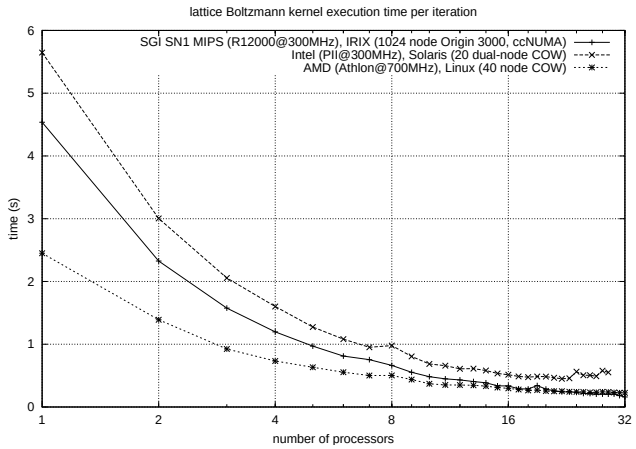

Fig. 6. Iteration time of the parallel lattice Boltzmann kernel on different multiprocessor systems.

vironment's architecture uses the distributed pipeline architecture described in section 3.2 and because the simulation component is, in general, the slowest executing component, the performance of the simulation environment in total is greatly increased (as shown by equation 2).

4.2 Performance of the network communication pipeline

Fig. 7 shows the mean throughput over 200 measurements of the multiple connection stage in the communication pipeline. As can be seen from this figure, the average throughput increases as more connections are used, but up to a maximum. Using more connections congests the network and no further increase in throughput can be obtained.

Fig. 8 shows the mean performance over 5 measurements of the complete network communication pipeline on a typical medical data set; using compression, multiple network connections, both with and without encoding. This figure illustrates that, on average, encoding doubles throughput. Although this figure shows the typical throughput that can be achieved, in some situations the total performance of the network communication pipeline resulted in a throughput of 62 Mbyte/s, which is over 5 times the bandwidth of the slowest network link (100 Mbit/s).

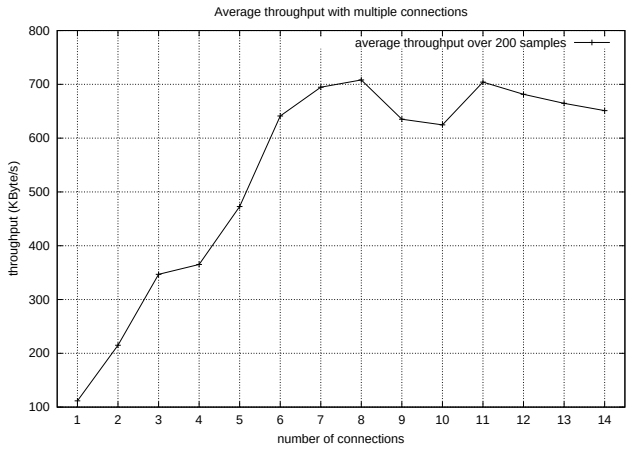

Fig. 7. Average network throughput when using multiple network connections on a 100 MBit/s link.

5 Conclusions and future work

Preliminary tests show a great performance increase over earlier versions of the interactive simulated vascular reconstruction environment. The results presented here were obtained using unoptimized algorithms; we expect future versions to increase performance even more.

The different stages in the communication pipeline require a certain amount of time to execute which adds delay to communication time. By tuning the parameters that influence this execution time, throughput can (in principle) be automatically optimized. For example; networks are generally shared by many institutes so that available bandwidth changes over time. The multiple connection technique can sense this change by measuring the effect of employing more or fewer connections during communication. An optimal number of connections can thus be determined dynamically (and transparently), while peers communicate. However, a different parameterization at one stage influences the execution time of subsequent stages which implies that parameter optimization is not a trivial task.

References

1. zlib homepage, 2001. On the web: `http://www.gzip.org/zlib/`.
2. A.M. Artoli, D. Kandhai, A.G. Hoekstra, and P.M.A. Sloot. Accuracy of shear stress calculations in the lattice Boltzmann method. Accepted for the 9th International Conference on Discrete Simulation of Fluid Dynamics.
3. R.G. Belleman, J.A. Kaandorp, D. Dijkman, and P.M.A. Sloot. GEOPROVE: Geometric probes for virtual environments. In P.M.A. Sloot et al., editors, *High Performance Computing and Networking (HPCN'99)*, pages 817–827, Amsterdam, 1999. Springer-Verlag.

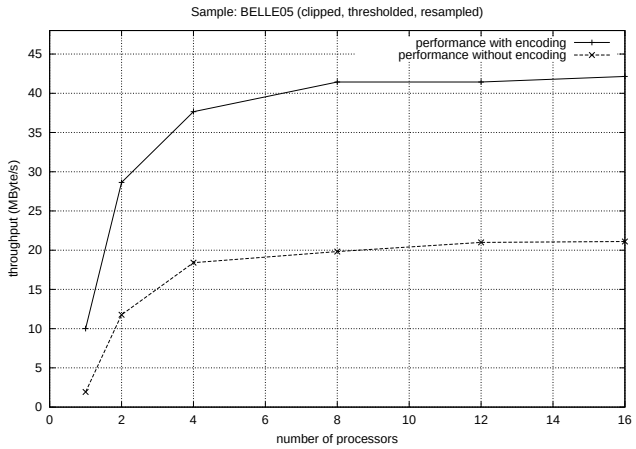

Fig. 8. Mean throughput of the network communication pipeline when used with the parallel LBM simulation kernel (with compression, shown with encoding and without encoding.

4. R.G. Belleman and P.M.A. Sloot. The design of dynamic exploration environments for computational steering simulations. In Marian Bubak et al., editors, *SGI Users' Conference*, pages 57–74, Kraków, 2000. CYFRONET AGH.
5. R.G. Belleman and P.M.A. Sloot. Simulated vascular reconstruction in a virtual operating theatre. In H.U. Lemke et al., editors, *CARS (Excerpta Medica, ICS-1230)*, pages 938–944, Berlin, 2001. Elsevier Science B.V.
6. R.G. Belleman, B. Stolk, and R. de Vries. Immersive virtual reality on commodity hardware. In R.L. Lagendijk et al., editors, *Seventh annual ASCI conference*, pages 297–304, Heijen, Netherlands, 2001. Advanced School for Computing and Imaging.
7. S. Chen and G.D. Doolen. Lattice Boltzmann method for fluid flows. *Annu. Rev. Fluid Mech.*, 30:329, 1998.
8. C. Cruz-Neira, D.J. Sandin, and T.A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *SIGGRAPH '93 Computer Graphics Conference*, pages 135–142. ACM SIGGRAPH, August 1993.
9. Open Channel Foundation. CAVERNsoft G2, A toolkit for high performance tele-immersive collaboration, 2001. On the web: `http://www.openchannelsoftware.org/projects/CAVERNsoft_G2/`.
10. D. Kandhai. *Large Scale Lattice-Boltzmann Simulations (Computational Methods and Applications)*. PhD thesis, Universiteit van Amsterdam, Amsterdam, the Netherlands, 1999.
11. D. Kandhai, A. Koponen, A.G. Hoekstra, M. Kataja, J. Timonen, and P.M.A. Sloot. Lattice Boltzmann hydrodynamics on parallel systems. *Computer Physics Communications*, 1998.
12. D. Kandhai, D. Vidal, A. Hoekstra, H. Hoefsloot, P. Iedema, and P.M.A. Sloot. Lattice-Boltzmann and finite element simulations of fluid flow in a SMRX mixer. *Int. J. Numer. Meth. Fluids*, 31:1019–1033, 1999.
13. A. Koponen, D. Kandhai, E. Hellen, M. Alava, A. Hoekstra, M. Kataja, K. Niskanen, P. Sloot, and J. Timonen. Permeability of three-dimensional random fiber webs. *Physical Review Letters*, 80(4):716–719, January 26 1998.