Optimization Strategies for Distributed Volume Ray Casting in Heterogeneous Networks

M. A. Zhernovkov
Department of computational mathematics and cybernetics
Ufa state aviation technical university
Ufa, Russia
e-mail: max.zhernovkov@googlemail.com

W. Schotte
Institute for information management in engineering
Karlsruhe institute of technology
Karlsruhe, Germany
e-mail: wolfgang.schotte@kit.edu

S. Bevier
Institute for information management in engineering
Karlsruhe institute of technology
Karlsruhe, Germany
e-mail: soeren@bevier.de

A. R. Gafarov
Department of computational mathematics and cybernetics
Ufa state aviation technical university
Ufa, Russia
e-mail: arthur.gafarov@gmail.com

Abstract[1]

In this paper the scheme of distributed calculations system in the heterogeneous network for visualization of voxel models is observed. Voxel models are generated with usage of DICOM – files. The volume ray casting algorithm is used as 3D – visualization algorithm. Optimization strategies and methods for visualization acceleration and the quality of resulting picture are observed.

1. Introduction

Nowadays in medicine a wide variety of scanning devices are used (such as scanners of computer tomography) for the diagnosis of patients. These devices scan certain parts of the patient's body and produce slices of images as standardized DICOM-files. Specialists can use this data for the diagnosis and analysis of patient's state. Or, for research, modern computer technologies allow visualizing this data in a 3D-space as voxel models. This approach allows the specialist an opportunity not only to study flat pictures, but interact with the whole model in a real-time: to see the model from the different angles, to do slicing, to manipulate individual parts of the model and much more. The process of analysis and research becomes much more convenient, affordable, complete and effective through this approach and the functional features of the software. The main problem of voxel models visualization is the speed of visualization. The main approach to speed-up the visualization is the creation of the distributed calculations system. Nowadays most distributed systems are based on the cluster model. The disadvantages of cluster models are the following: high cost, relatively narrow focus in the use, difficulty in scaling, difficulty in maintenance. Instead of the cluster a local network of computers can be used in the medical facility to do distributed calculations. The local network can be scaled more easily (computers can be just added or deleted from the network), it is easier to maintain it (only the proper configuration of the network and computers is needed to keep the system operable), it is more wide in use than cluster (computers of local network can be used not only to do distributed calculations but also for the needs of the stuff), it is not so expensive as cluster.

Proceedings of the 12th international workshop on computer science and information technologies CSIT'2010, Moscow – Saint-Petersburg, Russia, 2010

2. Problem statement

The goal is to create the distributed calculations system using local heterogeneous network of computers and the 3d-visualization algorithm to reach the speed of visualization in a real-time. As the 3D-visualization algorithm the modified volume ray casting algorithm is used. This algorithm is based on the algorithm described in [1] and implemented in the LESC (Lifecycle Engineering Solutions Center) at the KIT (Karlsruhe Institute of Technology). The goal is divided into the following tasks:

- Development of the distributed calculations system;
- Integration of the volume ray casting algorithm into the developed system;

- Development of the optimization strategies and algorithms for the effective operation of the system.

The problems are the following:

- Task generation algorithms;
- The algorithms of distributing the amounts of calculations between computers according to their calculating power;
- Mechanisms and algorithms for tracking the computer's available memory and the memory needed to do calculations;
- Tracking of bandwidth of the network parts;
- Latency tracking in the network.

Mathematically the task can be represented as a distribution problem of linear programming [5]. It can be formulated as follows.

The visualization area has M points (pixels). Consider the visualization area as set H. N workers are available. Each worker has its own limited amount of memory v_j (j = 1, …, N). Consider p_i – the memory, that is needed to store the i-th point (i = 1, …, M). Consider that $\sum_{j=1}^{N} v_j \geq \sum_{i=1}^{|M} p_j \cdot c_{i_j}$ – the time of visualization of the i-th point by j-th worker. Each worker j can have a set of points L_j (L_j C H) of the initial area. Then the time of visualization T_j of the set L_j by the worker j will be the sum of values $c_{kj} \in L_j$, and the used memory K_j of the worker j will be the sum of values $p_k \in L_j$. Then the goal of optimization task will be to fill the sets L_1, …, L_N by the points of set H in such a way to fulfill the following conditions:

$$1)\quad L_1 \cup L_2 \cup \ldots \cup L_N = H;$$

$$2)\quad \forall\, i, j\ (i = 1,\ldots,N;\ j = 1,\ldots,N):\ L_i \cap L_j = \mathbb{C};$$

$$3)\quad \sum_{j=1}^{N} T_j \rightarrow \min;$$

$$4)\quad \sum_{j=1}^{N} K_j \rightarrow \max.$$

However, the task cannot be solved by accurate methods because of incomplete of information on each iteration step. The system cannot know all values c_{ij} because of changing the scene parameters. Because of that the heuristic methods are used to solve the problems and to reach the goal.

3. Implemented Algorithms

3.1. The Distributed Calculations System Scheme

The distributed calculations scheme is based on the URay Framework approach described in [2] (pic.1).

When user changes the scene parameters (camera position or rotation, threshold density, etc.), the GUI – events are generated. After this the requests for visualization of the next picture are sent to the Task Manager, which generates the tasks for the Calculating nodes. Each Calculating node calculates the task and sends the results of the calculations to the Assembly node. Assembly node generates the resulting output picture and sends it to the Display

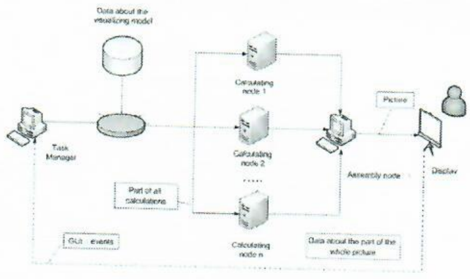

Fig. 1. The distributed calculations system scheme

3.2. Calculating nodes memory state tables generation

Each Calculating node contains an image of the whole visualized model. The image is loaded into the memory of the Calculating node in the following cases: Calculating node was added into the network or the user has changed the visualized model. After each calculation process the node generates a table about its memory state based on the visualized model image and voxels that were needed to make calculations. Each voxel has three states:

- Voxel was in the memory of the Calculating node, but wasn't used during the calculations;
- Voxel was needed to do calculations, but wasn't in the memory of the Calculating node;
- Voxel was needed to do calculations and it was in the memory of the Calculating node (in this case the amount of accesses to the voxel is recorded).

3.3. Dividing the calculating nodes into the classes

All Calculating nodes are divided into two categories: active and non-active. Active nodes are used to make calculations for the generation of the resulting picture. Non-active nodes are nodes with bad characteristics (big latency, etc.) and they are not used to make calculations until their characteristics become better. But if during the process of distributed calculations the characteristics of the non-active Calculation node will become better, then it can be set as an active node. It is also possible to move nodes from one class to another.

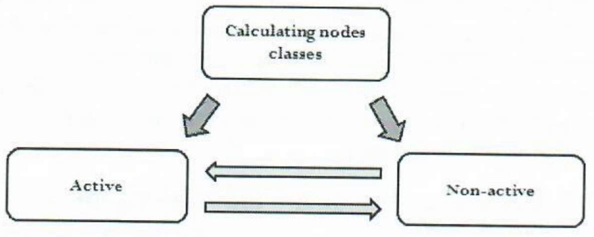

Fig. 2. The Calculating nodes classes

3.4. Tasks generation for the calculating nodes

On each iteration step it is necessary to visualize a certain number of lines of the visualization area.

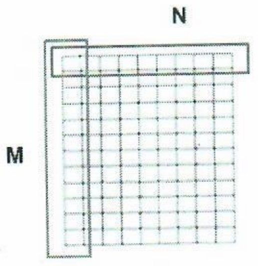

Fig. 3. The visualization area with M lines N pixels each

The task for each Calculating node has the following structure: the number of the first line to make calculations and the number of the total lines to make calculations.

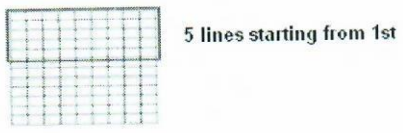

Fig. 4. An example of the task for the Calculating node

3.5. Latency tracking in the network

During the operation of the system the latency of each Calculating node V_i is measured. After this the average latency time K in the network is measured and for each Calculating node its latency is compared with the value K. If $V_i > K * 1.5$, then the Calculating node's class will be set as non-active. Otherwise, the node's class will be set as an active, if it was set as non-active earlier, or its class doesn't change. The factor 1.5 is chosen empirically.

3.6. Distributing the amounts of calculations between calculating nodes according to their power

Each Calculating node measures its calculating time T_i after each iteration step. After the generation of the resulting picture the average calculations time is calculated. On the next tasks generation process, performed by the Task Manager, the tasks are changed in the following way: if the time of calculations of the Calculating node was less than the average time, then the amount of lines for the calculations increases by M. Otherwise, it decreases by M. M – is the constant, chosen empirically and is equal 5% of the total amount of lines of the visualization area.

3.7. Preprocessing of the calculating nodes

When the new Calculating node is added into the network its class is set as non-active. After this its preprocessing is initiated. During that process a part of the visualized model is "downloaded" into the memory of the Calculating node. This allows the amount of artifacts in the resulting picture to be reduced when the node's class will be set as an active. After this the node's class is set as an active and the node is involved into the distributed calculations process. The "downloading" is based on the memory state tables of the node and test tasks.

3.8. Active calculating nodes memory management

When the calculations are done by the Calculating node the following procedures based on its memory table are done:

- The amount of memory needed to store missing voxels is calculated;
- If the amount of memory needed to store missing voxels exceeds the amount of free memory of the Calculating node then the deletion of unused data is initiated;
- The missing data "downloading" to the memory of the Calculating node is initiated.

4. Results

The GUI (Display) and the Assembly node are combined in one application at the moment.

The GUI of the system is shown on the next picture:

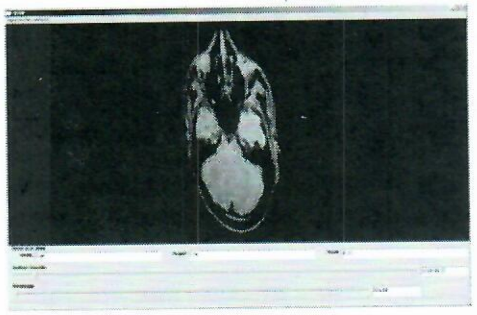

Fig. 5. The system's GUI (Display)

The Task Manager and the Calculating node are console applications.

The system was tested on the real data (human skull) with the following parameters of the model:

- Model's size – 86 Mb (46151728 voxels);
- Visualization area – 800*600 pixels.

The Calculating node (each) and network parameters are the following:

- Network bandwidth – 1 Gbps;
- 16 Gb of RAM;
- 4 CPU, 2.2 GHz each.

The graph acceleration (times)/number of Calculating nodes is shown on the next picture:

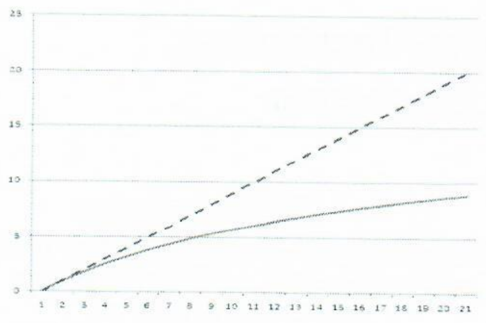

Fig. 6. The graph acceleration (times)/number of Calculating nodes

The horizontal axis represents the number of active Calculating nodes (all the same), vertical axis – the acceleration (in times). The red dotted line represents perfect case when the acceleration is linear dependent on the number of Calculating nodes. The blue line represents dependency in our system. The acceleration practically reaches its maximum when the number of Calculation nodes equals 20. There's no acceleration after this, because the performance of the system depends not on the calculating power, but on the network properties and the speed of transferring the data. The preprocessing time of the first Calculating node equals approximately 300 seconds and the amount of transferred voxels during the first test iteration equals 2828483 (6% of 46151728 voxels). It is necessary to say, that the time of preprocessing and the amount of transferred voxels decreases when the number of Calculating nodes increases. However, the amount of artifacts in the resulting picture increases when the number of the Calculating nodes increases. The next picture shows the system's GUI and the resulting picture of the visualized model after the preprocessing of the Calculating nodes and changing camera rotation. The amount of the Calculating nodes equals 3.

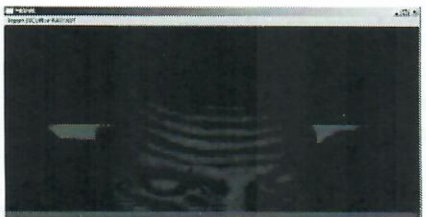

Fig. 7. The system's GUI. Three active Calculating nodes are in the network

5. Conclusions and future work

In this paper we have described the principles and algorithms that we used to build the prototype of distributed calculations system in the local heterogeneous network of computers. The system scheme is based on the URay Framework scheme [2] and the optimization of the systems operation is done using the heuristic algorithms. The system was tested on the real data. The results of the system performance are shown in chapter 4 (Results). This work is a part of the project of the LESC (Lifecycle Engineering Solutions Center) in the KIT (Karlsruhe Institute of Technology) and it is still under development. The most important areas for the future work are these:

- Improving the algorithms of distributing the memory data about the whole visualized model between Calculating nodes based on their memory tables;
- Implementing more effective and flexible algorithms of task distribution between Calculating nodes;
- Implementing the mechanism of rendering in a low-resolution and optimization strategies based on this mechanism;
- Transferred data compression algorithms;
- Implementing the algorithms of network's parts bandwidth evaluation;
- Implementing the algorithms of preliminary analysis of the scene before tasks generation;
- Improving and implementing new algorithms of analysis the Calculation's node possibility to be set as an active.

References

1. Marmitt G., Kleer A., Wald I., Friedrich H., Slusallek P. "Fast and Accurate Ray-Voxel Intersection Techniques for Iso-Surface Ray Tracing". Stanford, USA. November 16-18, 2004.
2. Repplinger M., Offler A.L., Rubinstein D., Slusallek P. "URay: A Flexible Framework for Distributed Rendering and Display". Computer Graphics Group, Saarland University, Germany, Technical Report TR-2008-01. December 19, 2008.
3. Wald I., Benthin C., Dietrich A., Slusallek P. "Interactive Ray Tracing on Commodity PC Clusters - State of the Art and Practical Applications". Saarland University, Germany, 2003, pp 499-508. (Lecture Notes on Computer Science 2790, Proceedings of EuroPar 2003).
4. Cedilnik A., Geveci B., Moreland K., Ahrens J., Favre J. "Remote Large Data Visualization in the ParaView Framework". Heirich A., Raffin B., Santos L.P. (eds.) editors. Eurographics Parallel Graphics and Visualization 2006, pp 162-170, May 2006.
5. Sircova E.D. "Mathematical methods in the planning and management of construction production". Tutorial, M., «High school», 1972, pp 265-293.
6. The DICOM standard (http://medical.nema.org/)
7. The BOINC system (http://boinc.berkeley.edu/)