The Architecture of Specialized GPU Clusters Used for Solving the Inverse Problems of 3D Low-Frequency Ultrasonic Tomography

Alexander Goncharsky and Sergey Seryozhnikov (✉)

Lomonosov Moscow State University, Moscow, Russia.

gonchar@srcc.msu.ru
s2110sj@gmail.com

Abstract. This paper is dedicated to the development of the architecture of specialized GPU clusters that can be used as computing systems in medical ultrasound tomographic facilities that are currently being developed. The inverse problem of ultrasonic tomography is formulated as a coefficient inverse problem for a hyperbolic equation. An approximate solution is constructed using an iterative process of minimizing the residual functional between the measured and simulated wave fields. The algorithms used to solve the inverse problem are optimized for a GPU. The requirements for the architecture of a GPU cluster are formulated. The proposed architecture accelerates the reconstruction of ultrasonic tomographic images by 1000 times compared to what is achieved by a personal computer.

Keywords: Ultrasonic tomography · Coefficient inverse problems · Finite-difference time-domain (FDTD) method · GPU clusters · Medical imaging.

1 Introduction

This paper focuses on using specialized supercomputers for medical ultrasonic tomography imaging. The primary application is the differential diagnosis of breast cancer. The development of ultrasonic tomography devices is currently at the prototype stage [1–3]. One of the most difficult problems in designing ultrasonic tomographic scanners is that the inverse problems of high-resolution wave tomography are nonlinear and have a very large number of unknowns — up to 10^8. The experimental data gathered in one examination amounts to approximately 5 GB. Solving such problems using precise mathematical models that take into account the diffraction, refraction and absorption of ultrasonic waves can be carried out only with the help of powerful modern supercomputers.

However, a general-purpose supercomputer cannot be included as a part of a tomographic setup. The aim of this study is to develop the architecture of specialized supercomputers for medical ultrasonic tomography. A specialized supercomputer should have an energy consumption not exceeding 10–20 kW and should fit into a

single rack. This specialized supercomputer should be optimized for the most effective implementation of the iterative gradient method developed in the authors' previous works [4–8]. Preliminary studies have shown that the optimal choice for this task is a GPU cluster. Using modern hardware, it is possible to design a specialized GPU cluster, which can be included in an ultrasonic tomography facility.

2 Formulation of the Inverse Problem of Ultrasonic Tomography

A simple mathematical model that takes into account the ultrasound diffraction and absorption effects is a scalar wave model based on a second-order hyperbolic equation. In this model, the acoustic pressure $u(\boldsymbol{r}, t)$ satisfies the equation:

$$c(\boldsymbol{r})u_{tt}(\boldsymbol{r},t) + a(\boldsymbol{r})u_t(\boldsymbol{r},t) - \Delta u(\boldsymbol{r},t) = \delta(\boldsymbol{r}-\boldsymbol{q})\cdot f(t)\,, \quad (1)$$

$$u(\boldsymbol{r},t=0) = 0,\;\; u_t(\boldsymbol{r},t=0) = 0\,,\; \partial_n u(\boldsymbol{r},t)\,|_{\mathrm{ST}} = p(\boldsymbol{r},t)\,. \quad (2)$$

Here, $c(\boldsymbol{r}) = 1/v^2(\boldsymbol{r})$, where $v(\boldsymbol{r})$ is the speed of sound in the medium, $\boldsymbol{r} \in \mathbf{R}^3$ is the point in space, $a(\boldsymbol{r})$ is the absorption coefficient, and Δ is the Laplace operator with respect to $\boldsymbol{r}$. The sounding pulse generated by the point source at $\boldsymbol{q}$ is described by the function $f(t)$; $\partial_n u(\boldsymbol{r}, t)|_{\mathrm{ST}}$ is the derivative along the normal to the surface S of the domain Ω, where $(\boldsymbol{r}, t) \in \mathrm{S}\times(0, T)$; the function $p(\boldsymbol{r}, t)$ is known. The conditions (2) represent the boundary and initial conditions. It is assumed that $v(\boldsymbol{r}) = v_0 = const$, $a(\boldsymbol{r}) = 0$ outside of the studied object. This simple model of wave propagation (1) can be used to describe ultrasonic waves in soft tissues.

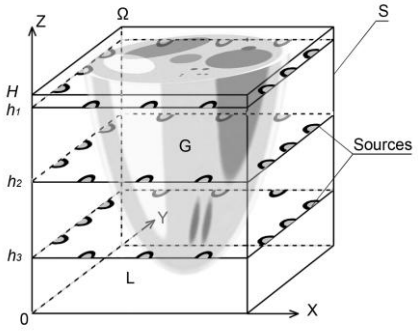


Fig. 1. The scheme of the experiment.

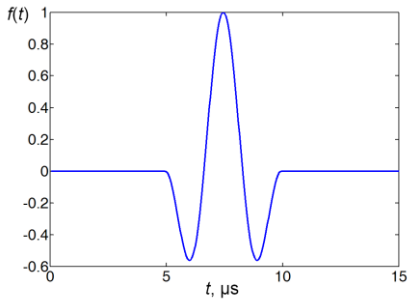


Fig. 2. Waveform of the sounding pulse.

Fig. 1 shows a typical scheme of the tomographic experiment. The studied object G is located inside the region Ω. For simplicity, we assume that the domain Ω is a cube of height H. The free space L is filled with water with a known sound speed v_0. The sources are located on the boundary S of the domain Ω in several planes: h_1, h_2, h_3. The detectors can be located on the side and bottom faces of the cube Ω. In ultra-

sound mammography applications, sources and detectors cannot be located on the upper side; thus, this is an incomplete-data tomography problem [6].

Let us consider the inverse problem of reconstructing the unknown coefficients $c(\boldsymbol{r})$ and $a(\boldsymbol{r})$ in equation (1), given that the acoustic pressure $U(\boldsymbol{s}, t)$ is measured at the points $\boldsymbol{s}$ of the boundary S for the time interval (0; T). The value of T is chosen to be large enough (~250 μs) so that all the waves passing through and reflected from the object are registered by the detectors. The measurements are performed for source positions $\boldsymbol{q}$. Fig. 2 shows a typical waveform of sounding pulses $f(t)$ emitted by the sources. For low-frequency acoustic tomography in the 500 kHz band, the duration of the pulses is 3–10 μs. The low-frequency approach allows to use a much smaller number of sources but requires precise measurements and a precise mathematical model [5].

The exact solution of the inverse problem includes the coefficients $c(\boldsymbol{r})$, $a(\boldsymbol{r})$, which, when substituted into equations (1)–(2), produce the wave field $u(\boldsymbol{r}, t)$ equal to the measured wave field $U(\boldsymbol{s}, t)$ at the detector points $\boldsymbol{s}$. Because the inverse problem is ill-posed, we formulate it as a problem of minimizing the residual functional with respect to its argument (c, a):

$$\Phi(u(c,a)) = \frac{1}{2}\int_0^T \int_S (u(\boldsymbol{s},t) - U(\boldsymbol{s},t))^2 d\boldsymbol{s} dt\,. \tag{3}$$

Here, $U(\boldsymbol{s}, t)$ is the acoustic pressure measured on the boundary S for the time interval (0, T); $u(\boldsymbol{r}, t)$ is the solution of the direct problem (1)–(2) for the given $c(\boldsymbol{r})$ and $a(\boldsymbol{r})$.

We use the gradient method to minimize the residual functional (3). Representations of the gradient $\Phi'(u(c,a))$ in various formulations were obtained in [7,8]. In [9] and [10], expressions for the gradient in the time-domain formulation were derived. The gradient $\Phi'(u(c,a)) = \{\Phi'_c(u), \Phi'_a(u)\}$, representing the linear part of the increment of the functional $\Phi(u(c,a))$ (3) with respect to the variation of the sound speed and the absorption coefficient $\{dc, da\}$, has the form:

$$\Phi'_c\ (u(c)) = \int_0^T w_t(\boldsymbol{r},t) u_t(\boldsymbol{r},t)\, d\,t\,, \quad \Phi'_a\ (u(a)) = \int_0^T w_t(\boldsymbol{r},t) u(\boldsymbol{r},t)\, d\,t\,. \tag{4}$$

Here, $u(\boldsymbol{r}, t)$ is the solution of the main problem (1)–(2), and $w(\boldsymbol{r}, t)$ is the solution of some "conjugate" problem for the given $c(\boldsymbol{r})$, $a(\boldsymbol{r})$ and $u(\boldsymbol{r}, t)$:

$$c(\boldsymbol{r}) w_{tt}(\boldsymbol{r},t) - a(\boldsymbol{r}) w_t(\boldsymbol{r},t) - \Delta w(\boldsymbol{r},t) = 0, \tag{5}$$

$$w(\boldsymbol{r}, t = T) = 0, \quad w_t(\boldsymbol{r}, t = T) = 0\,,\ \partial_n w|_{\mathrm{ST}} = u|_{\mathrm{ST}} - U\,. \tag{6}$$

At the points of the boundary S where no measured data are present, the boundary condition $\partial_n w|_{\mathrm{ST}} = 0$ is applied. To calculate the gradient $\Phi' = \{\Phi'_c(u), \Phi'_a(u)\}$ using formula (4), it is necessary to solve the direct problem (1)–(2) and the "conjugate" problem (5)–(6). With the calculated gradient, we can use various iterative algorithms to minimize the residual functional (4).

3 Numerical Algorithms for Solving Inverse Problems of Low-Frequency Ultrasonic Tomography

We use the finite-difference time-domain (FDTD) method to solve problems (1)–(2) and (5)–(6) numerically. Let us introduce a uniform discrete grid for the spatial coordinates (x, y, z) at the time t: $x_i = ih$, $y_j = jh$, $z_l = lh$, $t_k = k\tau$; $i, j, l = 1,...,N$, $k = 1,...,M$, where h is the grid step, and τ is the time step. To approximate equation (1), we use the following second-order finite-difference scheme:

$$c_{ijl}\frac{u_{ijl}^{k+1} - 2u_{ijl}^{k} + u_{ijl}^{k-1}}{\tau^2} + a_{ijl}\frac{u_{ijl}^{k+1} - u_{ijl}^{k-1}}{\tau} - \frac{\Delta u_{ijl}^{k}}{h^2} = 0\,. \tag{7}$$

Here, u_{ijl}^{k} are the values of $u(\boldsymbol{r}, t)$ at the point (i,j,l) at the time step k; c_{ijl} and a_{ijl} are the values of $c(\boldsymbol{r})$ and $a(\boldsymbol{r})$ at the point (i,j,l). The first term in (7) approximates $c(\boldsymbol{r})u_{tt}(\boldsymbol{r}, t)$, and the second term approximates $a(\boldsymbol{r})u_t(\boldsymbol{r}, t)$. The discrete Laplacian is denoted by Δu_{ijl}^{k}. It is computed using the formula:

$$\Delta u_{i_0 j_0 l_0}^{k} = \sum_{i=i_0-1}^{i_0+1}\sum_{j=j_0-1}^{j_0+1}\sum_{l=l_0-1}^{l_0+1} b_{ijl}u_{ijl}^{k}\,. \tag{8}$$

The b_{ijl} coefficients were presented, for example, in [11]. The parameters h and τ for the three-dimensional problem are connected by the Courant-Friedrichs-Lewy (CFL) stability condition: $\tau < h/\sqrt{3}c$. Collecting the terms with u_{ijl}^{k+1} in (7), we obtain an explicit finite-difference scheme for the wave equation (1). A similar scheme is used for equation (5). To solve the direct problem (1)–(2), non-reflecting boundary conditions [12] are applied at the boundary of the computational domain.

The components of the gradient of the residual functional are computed using the formulas:

$$(\Phi_c')_{ijl} = \sum_{k=2}^{M-2}\frac{u_{ijl}^{k+1} - u_{ijl}^{k}}{\tau}\frac{w_{ijl}^{k+1} - w_{ijl}^{k}}{\tau}\tau\,, \quad (\Phi_a')_{ijl} = \sum_{k=2}^{M-2}u_{ijl}^{k}\frac{w_{ijl}^{k+1} - w_{ijl}^{k-1}}{\tau}\tau\,, \tag{9}$$

where M is the number of time steps.

The iterative gradient descent algorithm is used to minimize the residual functional. As initial approximations for $c(\boldsymbol{r})$ and $a(\boldsymbol{r})$, we use $c^{(0)}=c_0=\text{const}$, $a^{(0)}=0$. These values correspond to the parameters of the environment. For water, $c_0=1500$ m·s^{-1}. The following actions are performed at each iteration (m):

1. The initial pulse is computed.
2. The direct problem (1)–(2) is solved, given that $c(\boldsymbol{r})=c^{(m)}$, $a(\boldsymbol{r})=a^{(m)}$. The acoustic pressure $u^{(m)}(\boldsymbol{r}, t)$ is calculated using formula (7). The values of $u(\boldsymbol{s}, t)$ at the points $\boldsymbol{s}$, where the detectors are located, are stored in memory.
3. The residual functional $\Phi^{(m)}=\Phi(u^{(m)}(\boldsymbol{r}))$ is computed using formula (3).

4. The "conjugate" problem (5)–(6) is solved to compute the wave field $w^{(m)}(\boldsymbol{r}, t)$.
5. The gradient $\Phi'(u^{(m)})$ is computed using formula (9). The stages 1–5 are repeated for all the sources, and the values of $\Phi^{(m)}$ и $\Phi'(u^{(m)})$ are summed for all the sources.
6. The current approximation is updated: $c^{(m+1)} = c^{(m)} + \lambda^{(m)} \Phi'_c(u^{(m)}(\boldsymbol{r}))$, $a^{(m+1)} = a^{(m)} + \lambda^{(m)} \Phi'_a(u^{(m)}(\boldsymbol{r}))$. The process returns to stage 2.

The step of the gradient descent $\lambda^{(m)}$ is chosen using a priori considerations. During the iterative process, the step is automatically corrected: $\lambda^{(m)}$ is decreased by 1.5 times if $\Phi^{(m)} > \Phi^{(m-1)}$; otherwise, it is increased by 25%. These rates are tuned for typical sound speed variations in soft tissues (1400–1600 $\text{m}\cdot\text{s}^{-1}$).

4 GPU Implementation of Computations for the Direct and Inverse Problems of 3D Ultrasound Tomography

4.1 Specific Features of Graphics Processors

Graphics processors have become the first widely available parallel architecture and are already used in ultrasonic tomography applications [3].

The specific feature of the GPU memory hierarchy is a very high memory performance combined with a slow communication channel. As in most modern systems, the performance of arithmetic units is much higher than the memory performance.

Graphics processors are designed for data-parallel tasks, where each of the thread blocks processes its own data area, for example, a part of the image that does not overlap with other parts. Thus, an algorithm optimized for a GPU cluster must first divide the problem into processes that require a relatively small amount of fast memory; second, it must subdivide the task for each GPU into completely independent thread blocks.

4.2 Specific Features of the Problem and Optimization of the Algorithm

The inverse problem of low-frequency ultrasound tomography has some specific features that allow optimizing the algorithm and reducing the computational complexity.

1. The main computational complexity of the algorithm lies in computing the gradient of the residual functional using the formulas presented in (9), which includes computing the wave fields $u(\boldsymbol{r}, t)$ and $w(\boldsymbol{r}, t)$. Functional (3) is the sum of the squared differences $\|U(\boldsymbol{s}, t)\text{-}u(\boldsymbol{s}, t)\|^2$ for all the sources and detectors. It can be computed for each source separately, and the results can be added together. The gradient (4) can also be computed as the sum of partial gradients for each source.

In the proposed architecture of the GPU cluster, each computing node calculates the gradient for one of the sources. A scheme for parallelizing the computations for sources S_1–S_6 is shown in Fig. 3. The input data for all computing nodes are the same: $c^{(m)}$, $a^{(m)}$ at the m-th iteration of the gradient descent method. The data transfers between the nodes occur only when an iteration of the gradient method is completed and when the current approximation is updated.

The total value of the gradient is computed and broadcast to all the nodes using the `MPI_Allreduce` operation. Then the new iterative approximation $c^{(m+1)}$, $a^{(m+1)}$ is computed. Various parameters of the algorithm are synchronzed via `MPI_Bcast` operations.

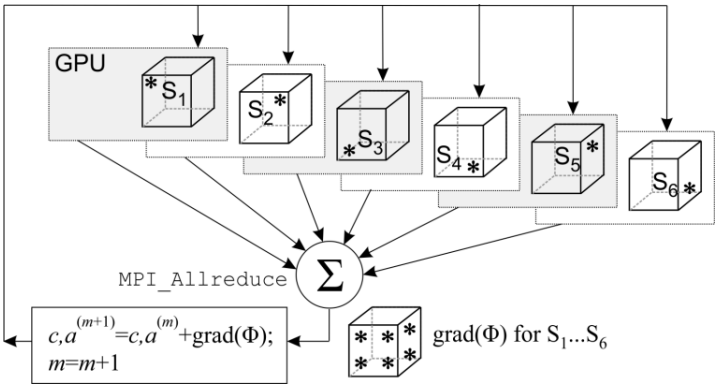


Fig. 3. Parallelizing the computations for multiple sources.

The amount of transferred data for a problem of size 400^3 is approximately 256 MB (400×400×400×4 bytes) for each node in each direction. Thus, the parallelization of computations by sources proves to be very effective. The total overhead is a few seconds per minute.

2. In medical diagnostic applications, the variance of the sound speed in soft tissues does not exceed 10–15%. This allows us to estimate the volume V, in which the wave propagates after being emitted from the source, in advance and to perform the computations within only this volume. The volume V is a sphere of radius $v_{\max}t$, where $v_{\max}$ is the maximum permissible sound speed in the model, and t is the current simulation time. This optimization is easily accomplished using the GPU by executing only the blocks for which $r \le v_{\max}t$. This optimization reduces the computation time by 25%.

3. The gradient of residual functional (4) is an integral over time. This means that to calculate the gradient, it is not necessary to store all the values of $u(\boldsymbol{r}, t)$ in the X×Y×Z×T region, which would require a huge amount of memory. It is sufficient to compute $u(\boldsymbol{r}, t_k)$ sequentially at time steps t_k. Thus, the required amount of memory is proportional to N^3, and the number of operations is proportional to N^4, where N is the number of grid points along one dimension.

Taking into account these features of the ultrasonic tomography problem, we propose a two-stage method according to the scheme presented in Fig. 4. In the first stage ("Forward-time computation"), the wave field $u(\boldsymbol{r}, t)$ generated by the source S in the volume V is computed sequentially in time. An absorbing layer of width d and simple non-reflecting boundary conditions [12] are used to cancel the reflected waves. A typical value of d is 32 grid points.

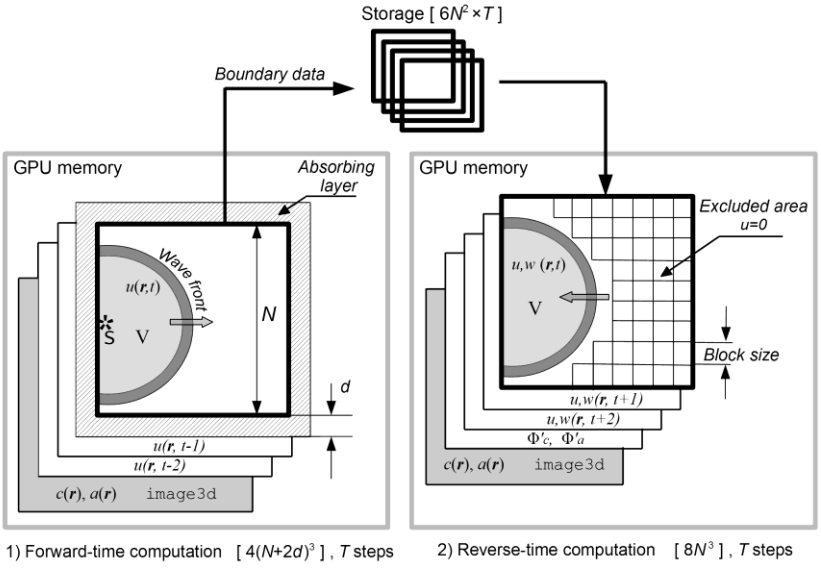


Fig. 4. The scheme of GPU computations performed in two stages.

To use the FDTD scheme (7), it is necessary to store $u(\boldsymbol{r}, t)$, $u(\boldsymbol{r}, t\text{-}1)$, $u(\boldsymbol{r}, t\text{-}2)$, which amounts to $3\cdot(N+2d)^3$ 32-bit words, and the coefficients $c(\boldsymbol{r})$ and $a(\boldsymbol{r})$ in the GPU memory. The coefficients do not change over time; thus, we can use the `image3d` structure, which enables texture caching, improving the performance by 10%.

Since the coefficients are unknown and determined only approximately, the 16-bit `half` data type is sufficient for their representation.The error between the exact and approximate solutions is ~2 $\text{m}\cdot\text{s}^{-1}$ for $c(\boldsymbol{r})$, which is ~2% of $|c(\boldsymbol{r})\text{-}c_0|$, and even worse for $a(\boldsymbol{r})$ [4]. This means that such variations of the coefficients do not noticeably affect the simulated ultrasound wave. Thus, a rounding error not exceeding 0.2% is acceptable.

The values of $u|_{\text{ST}}$ at the boundary of the computational domain are stored for use in the second stage. Fast access to these data is not needed; therefore, we place them in the system RAM. The required RAM capacity is $6N^2\cdot T$, where T is the number of time steps. It follows from the FDTD stability conditions that $T \approx 3N$, and the amount of memory needed for the boundary values is $18N^3$ 32-bit words.

To compute the gradient using formula (4), we need the values of $u(\boldsymbol{r}, t)$ and $w(\boldsymbol{r}, t)$ at the same points. To start computing $w(\boldsymbol{r}, t)$ from the last time step $t = T$, it is necessary to compute $u(\boldsymbol{r}, t)$ for all $t \le T$ first. In the second stage ("Reverse-time computation"), we solve equation (1) for $u(\boldsymbol{r}, t)$ and problem (5)–(6) for $w(\boldsymbol{r}, t)$ simultaneously in reverse time. The values of $u(\boldsymbol{r}, t)$ and $w(\boldsymbol{r}, t)$ are computed based on $u(\boldsymbol{r}, t+1)$, $w(\boldsymbol{r}, t+1)$, $u(\boldsymbol{r}, t+2)$ and $w(\boldsymbol{r}, t+2)$. The boundary conditions for $w(\boldsymbol{r}, t)$ are determined from the experimental data $U(\boldsymbol{s}, t)$ by using formula (6).

To fill in the missing values of $u(\boldsymbol{r}, t)$ at the boundary, we use the values of $u|_{ST}$ stored in the memory in the first stage. The wave field $u(\boldsymbol{r}, t)$ obtained in this way is equal to $u(\boldsymbol{r}, t)$ computed in the first stage. The recalculation of $u(\boldsymbol{r}, t)$ allows us to use a data array of size X×Y×Z, not X×Y×Z×T. The numerical error introduced by the recalculation does not exceed 10^{-5}, making it negligible for practical measurements.

The amount of data stored in the GPU memory in the second stage is $8N^3$: $u(\boldsymbol{r}, t)$ and $w(\boldsymbol{r}, t)$ for the three time steps (six 32-bit words), the gradient values Φ'_c and Φ'_a (one word), which are updated, and the read-only coefficients $c(\boldsymbol{r})$, $a(\boldsymbol{r})$ (one word).

Since the gradient value is small compared to the coefficients $c(\boldsymbol{r})$, $a(\boldsymbol{r})$, the `short` data type is sufficient for its representation. The use of a 16-bit integer type reduces memory usage and computation time, while providing an acceptable level of precision to accumulate the integrals (4). A scaling factor of 24000/max|Φ'| is applied to the gradient Φ' in order to limit the possible increase of the gradient at the next iteration and to prevent the coefficients from exceeding the physically realistic range.

4.3 The Finite Difference Method

The finite difference method has been proven to be efficient for numerical simulations of physical processes. The problem under consideration is no exception. Solving problems (1)–(2) and (5)–(6) requires numerical simulation of the wave field with given parameters. An explicit FDTD scheme is a naturally data-parallel algorithm because the values at all the grid points are computed in the same way and do not depend on each other. Such algorithms fit well in SIMD/SPMD-architectures.

To compute the 3D wave fields at each time step, the volume V is divided into blocks that are processed by each thread block of the GPU. The blocks that are far from the ultrasound source, where $u(\boldsymbol{r}, t) = 0$ for the current simulation time t, are excluded. The "Z-marching" method is used inside each block because the optimal number of parallel threads in a typical GPU is several hundreds per multiprocessor (MP), and the dimension of the problem is approximately 400^3. The typical number of 3D blocks to be processed is approximately 10000. The thread blocks are two-dimensional (x, y), and each thread computes the data sequentially along the Z-axis.

The discrete Laplacian (8) can be reduced to scalar products of three-component vectors by collecting the terms, because only four of the b_{ijl} coefficients have non-repeating values:

$$\Delta u(i, j, z_0) = \boldsymbol{b_0} \cdot \boldsymbol{u}(z_0) + \boldsymbol{b_1} \cdot (\boldsymbol{u}(z_0\text{-}1) + \boldsymbol{u}(z_0\text{+}1)),$$

$$\boldsymbol{b_0} = \{b_{000}, b_{100}, b_{110}\}, \boldsymbol{b_1} = \{b_{100}, b_{110}, b_{111}\},$$

$$\boldsymbol{u}(z) = \{u_{ij},\ u_{i\,j+1} + u_{i+1\,j} + u_{i\,j-1} + u_{i-1\,j},\ u_{i+1\,j+1} + u_{i+1\,j-1} + u_{i-1\,j+1} + u_{i-1\,j-1}\}.$$

To compute Δu sequentially along the Z-axis, we need to keep three vectors per thread in the registers: $\boldsymbol{u}(z_0)$, $\boldsymbol{u}(z_0\text{-}1)$, and $\boldsymbol{u}(z_0\text{+}1)$. At each step of the Z-marching method, $\boldsymbol{u}(z_0\text{+}1)$ becomes $\boldsymbol{u}(z_0)$, the new $\boldsymbol{u}(z_0\text{+}1)$ is computed, and the results for the $z = z_0$ plane are saved in the global memory. Small amount of data per thread and mostly sequential memory access pattern allow for an efficient GPU implementation.

4.4 Profiling the Algorithm

For profiling, a test run of 6 iterations of the gradient method is carried out for a 320×320×320 problem using the NVidia GeForce GTX Titan graphics card. The OpenCL interface was used for GPU programming. Program profiling statistics are shown in Table 1. The computational kernels that require less than 0.01% of the GPU time were left out. The start and end times of each kernel run were obtained using OpenCL profiling events, then the execution times and overlap times were computed. The total time of all GPU operations corresponds to 100%.

The test shows that, as expected, almost all the time is spent on the calculation of 3D problems (1)–(2) and (5)–(6) ("ForwardWave" and "BackwardWave" functions). The boundary conditions require approximately 6% of the time because the memory access pattern is mostly random. The performance impact decreases as the problem size N increases, because the boundary contains $\sim N^2$ elements, while the volume contains $\sim N^3$ elements.

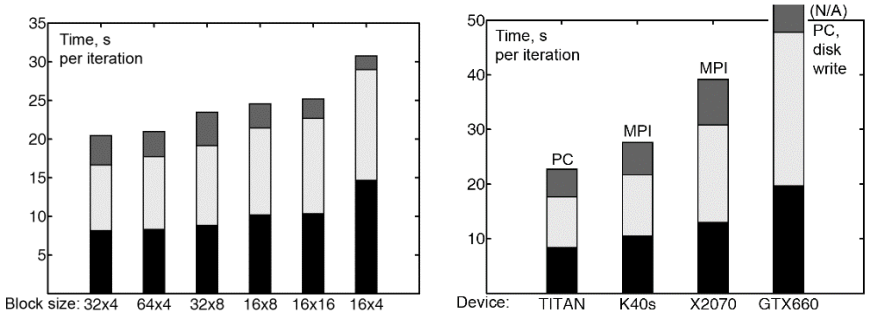
Table 1. Execution time for computational kernels and data transfers.

Runs	avg, μs	min, μs	max, μs	Over-lap, s	Total time s	% GPU	Function
293216	271	252	499	0	79.47	0.94	LoadBound
306172	11184	112	15842	3423.1	3424.39	0.02	SaveBound
1375	911	385	3049	0	1.25	0.01	Initialize
306172	1215	1139	2041	0	372.11	4.41	FwdBoundCond
306172	11645	275	16146	0	3565.55	42.21	ForwardWave
293216	14292	712	17981	0	4190.81	49.62	BackwardWave
12331	11076	38	16143	0	136.59	1.62	SaveExData
293216	28	23	251	0	8.37	0.10	LoadExData
14671	2017	1247	3615	0	29.60	0.35	DisplayGL
8512	1100	1063	1245	0	9.37	0.11	ScalarMax
							DATA TRANSFERS
306172	251	26	1506	76.19	77.08	0.01	_LoadFromGPU
293216	32	26	51	0	9.51	0.11	_SaveToGPU
293216	33	32	65	0	9.91	0.12	_SaveExData
39767	76520	0	47662	0	30.43	0.36	_BufferOp

The data transfers between the GPU and the system memory require less than 1% of the time. Some data transfers are performed in parallel with the calculations. The total program execution time exceeded the total GPU time by 7.5%. This value shows the overhead costs that are not parallel with the GPU computations, like summation and data distribution via the MPI interface.

Further optimization of the algorithm includes choosing the size of the thread blocks, which determines the optimal use of the GPU register files and memory access circuits. Fig. 5a shows the execution time for different block sizes on a GTX Titan device. The optimal block size choice can provide up to 15% performance boost, and the size of 32×4 was found to be optimal for this problem on all of the tested devices.

The algorithm was tested using a number of GPU devices: NVidia Tesla K40s on the "Lomonosov-2" supercomputer, NVidia Tesla X2070 on the "Lomonosov" supercomputer of the Moscow State University Supercomputer Center [13], NVidia GeForce GTX Titan and GTX 660 on personal computers.



a) **b)**

Fig. 5. Execution time: direct problem (bottom), "conjugate" problem (middle), other (top): **a)** for various block sizes, GTX Titan; **b)** for various devices, 32×4 block size.

Fig. 5b shows the execution time for various devices. On the supercomputers, the test run comprised executing 8 tasks in parallel on 8 devices and collecting the data using the MPI interface, as shown on Fig. 3; on PCs, a single task was executed and the data were saved to a disk. The `MPI_Allreduce` operation required less than 1% of the total time (280 ms for 8 parallel tasks, 340 ms for 48 parallel tasks on the "Lomonosov" supercomputer equipped with a 40 Gbit/s QDR Infiniband network).

The tests demonstrated a direct relationship between the performance and the memory bandwidth of the device. The more expensive Tesla devices showed lower performance, which means that the specific Tesla features are not relevant to this particular problem. The parameters of the algorithm, such as caching in local memory vs. automatic caching, were tuned for best performance on each device.

5 The Architecture of the Computing System for Solving Inverse Problems of 3D Ultrasonic Tomography

The main problem in 3D wave tomography imaging is that in a typical problem of size 400^3, the number of unknowns reaches 10^8. The number of ultrasound sources required to collect enough data is approximately 20–40, and the total number of computed data points reaches 10^{12}. These computations have to be performed within a reasonable time.

The FDTD method is a data-intensive task, for which the memory performance is of prime importance. Therefore, graphics processors are a natural choice. GPU computing performance remains high as long as the data fits into the on-board memory of the device.

Let us formulate the requirements for a computer system that can be used in a tomographic complex. To determine these requirements, we ran a series of tests. Ta-

ble 2 shows the execution times and memory requirements for different problem sizes. These tests were performed on an NVidia GeForce GTX Titan device.

The performance is a primary limiting factor here because the execution time is proportional to N^4, whereas the amount of memory is proportional to N^3. This means that there is an optimal amount of on-board GPU memory. Thus, using expensive devices with large amounts of memory is impractical because of the greatly increased time needed to process such amount of data.

Table 2. Memory requirements and execution times for various problem sizes.

3D problem size (N)	256	288	320	384	416	448	512
GPU memory used, GB	0.8	1.2	1.5	2.2	2.8	3.5	5.0
System RAM used, GB	1.5	2	3	5	6	8	12
Time per one iteration, s	11	16	23	48	63	92	160

Let us assume that a practically acceptable computation time is 1 hour for 100 gradient method iterations (36 s per iteration). Then, we can use one device with 3 GB of on-board memory for each ultrasound source when the problem size is limited to 360^3. To tackle problems of sizes up to 400^3, we can use two such devices per source, or a single higher-end device. This setup requires 6 GB of system RAM and 3.5 GB of GPU memory for each source. For example, the NVidia GeForce GTX 690 graphics card, which contains two 2-GB GPU devices on a single board, can be used to this end. The proposed algorithm theoretically allows splitting the processed volume between two GPUs across the Z-axis. To balance the load, the partitions should include the same number of blocks, which is known a priori. This is a standard approach to parallelizing 3D FDTD schemes.

Recently announced devices with High Bandwidth Memory architecture (HBM) have approximately 3 times the performance compared to devices with GDDR5 memory. Using one such device per source, the problem size can be increased to 480^3. In this setup, the device should have 6 GB of on-board memory, and 10 GB of system RAM per source is required. A problem size of 480^3 is close to the practical requirements for ultrasonic mammography applications. This grid size provides a resolution of 0.4 mm over a 20 cm range.

Let us formulate the essential characteristics of the specialized GPU computer.

Graphics processors and RAM storage. Each computing node must contain at least a sufficient number of GPU devices to compute the residual functional gradient for a single source in a reasonable time. For problem sizes up to 360^3 — one GPU device (~250 GB/s memory bandwidth, 3 GB of on-board memory) and 4 GB of RAM storage for each ultrasound source. For larger problems — two consumer-class GPU devices (4–6 GB of total GPU memory), or one higher-end or HBM-class device, and 10–12 GB of RAM storage per source. The total number of ultrasound sources is 20–30. The number of GPU devices per node should be maximized in order to reduce the total number of hardware components. Currently available mainboards can support up to 4 devices.

Central processors. The CPUs distribute data to the GPUs and between computing nodes. The CPUs must meet the minimal requirements.

Communication network. The network is used to combine the data from all the nodes to compute the gradient and to distribute the next iterative approximation to all the nodes (Fig. 3). These actions are carried out only once per iteration. Because only the all-reduce and broadcast operations are needed, the optimal network topology is a star or a tree topology. The minimal required bandwidth is ~200 MB/s.

Disk storage. The disk storage must meet the minimal requirements. The amount of data to be stored is under 50 GB for one experiment.

These requirements can be met using common solutions that fit in a single rack and have a power consumption of 10–20 kW. Modern graphics cards require 150–300 W per unit, and this value steadily decreases as the technology improves. A mainboard with CPU and RAM requires no more than 250 W; thus, a node containing a mainboard and four GPU devices requires at most 1.5 kW.

Using widely available hardware components, we can build computing systems that provide the medical image reconstruction using the 3D wave tomography technology. The performance gain relative to a single-CPU personal computer is on the order of 1000 times. This estimation is based on a typical 30-fold difference between CPU and GPU implementations of 3D FDTD methods [14], multiplied by an estimated number of devices in the cluster of 32.

Graphics cards and GPU supercomputers continue to improve. There is no doubt that in a few years, the performance of graphics processors will increase by several times, while the energy footprint will decrease. All this speaks in favour of using GPU clusters as specialized supercomputers for the new ultrasonic tomographic systems currently being developed.

6 Conclusions and Discussion

The requirements for a GPU cluster that provides an efficient implementation of the iterative gradient methods of the reconstruction of tomographic images are formulated. The specialized GPU cluster can achieve a 1000-fold performance increase compared to that of a single-CPU personal computer. The characteristics, such as the size, power consumption, and cost, of a GPU cluster allow it to be used as a computing system in the new medical ultrasonic tomographic complexes being developed.

There are other high-performance solutions that can be used for solving the inverse problems of ultrasound imaging. Modern multicore systems, such as the Intel Xeon Phi, have performances comparable to that of a GPU, but these systems are much more expensive because they are much more complex devices designed for a wide range of applications. CPU-based systems require a large number of memory access channels and a large cache to be efficient for 3D imaging. According to the authors, GPU clusters have the most promising architecture for high-performance 3D image reconstruction.

Acknowledgements

This research was supported by Russian Science Foundation (project No. 17-11-01065). The study was carried out at the Lomonosov Moscow State University.

References

1. Duric N., Littrup P., Li C., Roy O. et al. Breast ultrasound tomography: Bridging the gap to clinical practice. Proc. SPIE. Medical Imaging 8320, 83200O (2012).
2. Wiskin J., Borup D., Andre M., Klock J. et al. Three-dimensional nonlinear inverse scattering: Quantitative transmission algorithms, refraction corrected reflection, scanner design, and clinical results. J. Acoust. Soc. Am. 133, 3229 (2013).
3. Birk M., Dapp R., Ruiter N.V., Becker J. GPU-based iterative transmission reconstruction in 3D ultrasound computer tomography. J. Parallel Distrib. Comput. 74, 1730–1743 (2014).
4. Goncharsky A.V., Romanov S.Y. Inverse problems of ultrasound tomography in models with attenuation. Phys. Med. Biol. 59, 1979–2004 (2014).
5. Goncharsky A.V., Romanov S.Y., Seryozhnikov S.Y. A computer simulation study of soft tissue characterization using low-frequency ultrasonic tomography. Ultrasonics 67, 136–150 (2016).
6. Goncharsky A.V., Romanov S.Y., Seryozhnikov S.Y. Inverse problems of 3D ultrasonic tomography with complete and incomplete range data. Wave Motion 51, 389–404 (2014).
7. Goncharskii A.V., Romanov S.Y. Two approaches to the solution of coefficient inverse problems for wave equations. Comput. Math. Math. Phys. 52, 245–251 (2012).
8. Goncharsky A.V., Romanov S.Y. Iterative methods for solving coefficient inverse problems of wave tomography in models with attenuation. Inverse Problems 33(2), 025003 (2017).
9. Natterer F. Possibilities and limitations of time domain wave equation imaging. In: Contemporary Mathematics Vol. 559, pp. 151–162. American Mathematical Society, Providence (2011).
10. Beilina L., Klibanov M.V., Kokurin M.Y. Adaptivity with relaxation for ill-posed problems and global convergence for a coefficient inverse problem. J. Math. Sci. 167, 279–325 (2010).
11. Mu S.-Y., Chang H.-W. Dispersion and local-error analysis of compact LFE-27 formula for obtaining sixth-order accurate numerical solutions of 3D Helmholtz equation. Progress In Electromagnetics Research 143, 285–314 (2013).
12. Engquist B., Majda A. Absorbing boundary conditions for the numerical simulation of waves. Math.Comput. 31, 629–651 (1977).
13. Voevodin Vl.V., Zhumatiy S.A., Sobolev S.I., Antonov A.S., Bryzgalov P.A., Nikitenko D.A., Stefanov K.S., Voevodin Vad.V. Practice of "Lomonosov" Supercomputer. Open Systems J. (2012), No.7, 36–39 (In Russian).
14. Zhang L., Du Y. and Wu D. GPU-Accelerated FDTD simulation of human tissue using C++ AMP. In: 31st International Review of Progress in Applied Computational Electromagnetics (ACES), pp. 1–2. Williamsburg, VA (2015).