

# Parallel Simulation of 3D Nonlinear Acoustic Fields on a Linux-cluster

Xing Cai      Åsmund Ødegård  
*Department of Informatics*  
*University of Oslo*  
*P.O. Box 1080, Blindern*  
*N-0316 Oslo, Norway*  
{xingca,aasmundo}@ifi.uio.no

## Abstract

*Simulating the propagation of 3D ultrasonic waves in a nonlinear medium is a demanding task. It requires the solution of time-dependent and nonlinear partial differential equations (PDEs). For such nonlinear PDEs, we have to use an implicit numerical method that is very CPU-intensive. Parallel simulation is therefore essential for studying those ultrasonic waves with satisfactory accuracy. We present in this paper the parallelization of an ultrasonic wave simulator and report some numerical experiments that have been run on a 24-node Linux cluster. Our CPU-measurements indicate that Linux clusters can deliver satisfactory parallel computing power for the numerical solution of PDEs. For simulating 3D ultrasonic waves in particular, we have found that our Linux cluster, which consists of 48 Pentium-III 500MHz processors inter-connected with a 100Mbit/s ethernet network, is fully comparable with an SGI Origin 2000 machine.*

## 1. Introduction

In order to mathematically model the propagation of 3D ultrasonic waves in a nonlinear medium, we have to resort to a system of nonlinear PDEs. Solution of these time-dependent and nonlinear PDEs requires an implicit numerical method. The resulting numerical simulations are very CPU-intensive, thus can only be run on parallel computers when a large scale discretization is employed. Traditionally, such huge computing power is acquired on multi-million-dollar supercomputers. However, the emerging research field of scientific computing on clusters of low-cost PCs provides a more cost-effective alternative; see [3, 4, 16]. Although the inter-processor communication of those PC clusters, compared with that of high-end supercomputers, is considerably slow against the processor speed, numerical solution of PDEs is well suited on them, see [1, 2, 11].

This is because the involved computational kernel is almost exclusively an iterative solution process for systems of linear equations, where the encountered inter-processor communication consists of two simple types. The first type is all-to-all broadcast of very short messages and the second type is that a processor exchanges long messages with *only* its immediate neighbors. When sequential PDE simulators are parallelized carefully using a message-passing programming model, those PC clusters may thus produce computing power comparable with that of medium scale supercomputers. In particular, our numerical experiments of simulating 3D ultrasonic waves achieve better performance on a 48-CPU linux-cluster, compared with a SGI Origin 2000 machine when using the same number of processors.

The rest of the paper is organized as follows. Section 2 gives a brief introduction to the mathematical model of propagating acoustic waves in a nonlinear medium. Section 3 then follows with a presentation of the numerical method to be used. Afterwards, Section 4 is concerned with a numerical ultrasonic wave simulator and its parallelization. Thereafter, Section 5 contains the specifications of a 24-node Linux cluster and the results of two benchmark tests. Numerical experiments and measurements of some 3D ultrasonic wave simulations are reported in Section 6, before Section 7 summarizes with some discussions.

## 2. The mathematical model

We start with the nonlinear model for acoustic waves as derived by Makarov and Ochmann in [15]:

$$\nabla^2 \varphi - \frac{1}{c^2} \frac{\partial^2 \varphi}{\partial t^2} + \frac{1}{c^2} \frac{\partial}{\partial t} \left[ (\nabla \varphi)^2 + \frac{B/A}{2c^2} \left( \frac{\partial \varphi}{\partial t} \right)^2 + b \nabla^2 \varphi \right] = 0, \quad (1)$$

$$p - p_0 = \rho_0 \frac{\partial \varphi}{\partial t} + \frac{\rho_0}{2c^2} \left( \frac{\partial \varphi}{\partial t} \right)^2 - \frac{\rho_0}{2} (\nabla \varphi)^2 + \rho_0 b \nabla^2 \varphi. \quad (2)$$

The above model (1-2) is derived from the equations of hydrodynamics for compressible fluids under two assumptions, where the first one says that the acoustic wave amplitude is small such that only second-order nonlinear terms are kept. The second assumption is that we only consider low-absorbing fluid, so only linear thermoviscous terms are taken into account. Equation (1) governs the velocity potential  $\varphi$ , and the relation between pressure  $p$  and  $\varphi$  is implicitly given by Equation (2). Moreover,  $c$  is the speed of sound,  $\rho_0$  is the density,  $p_0$  is the initial pressure,  $b$  is the absorption parameter and  $B/A$  is the nonlinearity parameter. The problem is to be solved in some domain  $\Omega \in \mathbb{R}^3$  from  $t = 0$  and for a time period forward.

In the present paper, a simpler relation between  $\varphi$  and  $p$  is used. That is, we assume that the nonlinear and lossy terms in (2) are small, thus get the following simpler relation

$$p - p_0 = \rho_0 \frac{\partial \varphi}{\partial t}. \quad (3)$$

Therefore, equations (1) and (3) constitute the nonlinear model to be studied by the present paper. Furthermore, we assume that the system is at rest initially, with some constant velocity potential. This gives the following initial conditions

$$\varphi(x, 0) = \varphi_0, \quad x \in \Omega, \quad (4)$$

$$\frac{\partial \varphi}{\partial t}(x, 0) = 0, \quad x \in \Omega. \quad (5)$$

The boundary  $\partial\Omega$  consists of two disjointed parts:  $\partial\Omega_T$  and  $\partial\Omega_{T^c}$ , where  $\partial\Omega_T$  is the part of the boundary on which the ultrasound transducer is located. On  $\partial\Omega_T$  the pressure waves we want to study are implemented. The present work is part of a larger project whose main objective is to study limited diffraction beams. Two variants of such beams are the Bessel beams of order zero,

$$p(r, z, t) = J_0(\alpha r) e^{j(\beta z - \omega t)}, \quad (6)$$

and the X-wave of order zero,

$$p(r, z, t) = \frac{a_0}{\sqrt{r^2 \sin^2 \zeta + [a_0 - j(z \cos \zeta - ct)]^2}}. \quad (7)$$

Here  $a_0$  is a constant,  $\omega$  is the frequency,  $r$  is the distance from the center line of the transducer,  $z$  is the propagation distance from the transducer surface, and  $\zeta$ ,  $\alpha$  are design parameters with  $\beta = \sqrt{(\omega/c)^2 - \alpha^2}$ . See [14, 10, 9] for the details of design and implementation.

On  $\partial\Omega_{T^c}$  we adopt a first order non-reflecting boundary derived with methods from [8]

$$\frac{\partial \varphi}{\partial n} = -\frac{1}{c} \frac{\partial \varphi}{\partial t}, \quad x \in \partial\Omega_{T^c}. \quad (8)$$

In comparison with the nonlinear model (1,3), we also consider the following linear model

$$\nabla^2 p - \frac{1}{c^2} \frac{\partial^2 p}{\partial t^2} = 0, \quad (x, t) \in \Omega \times \mathbb{R}^+, \quad (9)$$

which is a linearized wave equation in pressure, arising from the the equations of hydrodynamics by assuming an incompressible fluid. The initial and boundary conditions will be the same as those of the nonlinear model, while the non-reflecting boundary condition for this linear model is

$$\frac{\partial p}{\partial n} = -\frac{1}{c} \frac{\partial p}{\partial t}, \quad x \in \partial\Omega_{T^c}. \quad (10)$$

### 3. The numerical method

Our approach to solving the nonlinear equation (1) is to use Galerkin finite element method in the spatial domain, combined with finite difference approximations of the temporal derivatives. Using centered differences for time derivatives, we get a semi-discrete scheme as:

$$\begin{aligned} & \nabla^2 \varphi^n - \frac{1}{c^2 \Delta t^2} (\varphi^{n+1} - 2\varphi^n + \varphi^{n-1}) \\ & + \frac{1}{c^2 \Delta t} \left\{ [(\nabla \varphi)^2 + \frac{B/A}{2c^2} \left( \frac{\partial \varphi}{\partial t} \right)^2 + b \nabla^2 \varphi]^{n+\frac{1}{2}} \right. \\ & \left. - [(\nabla \varphi)^2 + \frac{B/A}{2c^2} \left( \frac{\partial \varphi}{\partial t} \right)^2 + b \nabla^2 \varphi]^{n-\frac{1}{2}} \right\} = 0. \end{aligned} \quad (11)$$

Assume that  $\{N_i\}_{i=1}^m$  is a finite element basis for the solution domain  $\Omega$ , such that the approximation of  $\varphi$  at time level  $n$  is  $\sum_i \varphi_i^n N_i$ . Then using the Galerkin finite element method for the spatial derivatives in (1) will yield an  $m \times m$  system of nonlinear equations involving approximations from three time levels:  $\varphi^{n+1}$ ,  $\varphi^n$  and  $\varphi^{n-1}$ . At time level  $n+1$ , when  $\varphi^n$  and  $\varphi^{n-1}$  are already computed, we can solve the nonlinear problem (11) by using Newton iterations, which will use the solution of the linearized model (9) as the starting guess. More specifically, at each time level we first solve (9) and then (11). A Krylov subspace method will be used to iteratively solve the linear system of equations encountered in each Newton iteration. More details on the numerical method may be found in [17].

A simplified approach is used for the linear model (9). Using centered differences for the time derivatives, we get

the semi-discrete scheme

$$\nabla^2 p^n - \frac{1}{c^2 \Delta t^2} (p^{n+1} - 2p^n + p^{n-1}) = 0, \quad x \in \Omega. \quad (12)$$

Then the Galerkin finite element method will yield a set of  $m$  equations for the  $m$  unknowns  $p_i^{n+1}$ ,

$$\begin{aligned} \sum_{i=1}^m \left[ \frac{1}{c \Delta t} \int_{\partial \Omega_{T^c}} N_i N_j d\Gamma + \frac{1}{c^2 \Delta t^2} (N_i, N_j) \right] p_i^{n+1} \\ = \frac{1}{c \Delta t} \int_{\partial \Omega_{T^c}} p^n N_j d\Gamma - (\nabla p^n, \nabla N_j) \\ + \frac{1}{c^2 \Delta t^2} (2p^n - p^{n-1}, N_j), \quad j = 1, \dots, m, \end{aligned} \quad (13)$$

which can be solved *explicitly* at each time level. In (13) above, the form  $(u, v)$  means  $\int_{\Omega} uv dx$ .

#### 4. A parallel ultrasonic wave simulator

We started the creation of a parallel ultrasonic wave simulator by building a sequential simulator in the framework of Diffpack, which is an object-oriented programming environment for developing PDE software; see [6, 13]. More specifically, Diffpack consists of a large number of “building blocks” that are organized in form of hierarchies of C++ classes and cover all the sub-tasks in the numerical solution of PDEs. Examples of such building blocks are mesh generation, finite element and difference discretization functionality, time stepping control, different iterative strategies for solving systems of nonlinear equations, as well as an extensive collection of solution methods, preconditioners and convergence monitors for systems of linear equations.

The construction of a sequential Diffpack simulator for ultrasonic waves is therefore realized by deriving a new C++ class from a generic Diffpack base class containing functionality for finite element discretization. The most important task is to re-implement a virtual C++ member function that translates the integrals needed in writing out (11) of the Galerkin finite element method. Another main task is to implement the strategy of first solving the linearized model (9) to obtain a good starting guess and then submitting it to the subsequent Newton iterations at each time level. Thanks to Diffpack’s rich collection of PDE class hierarchies, the development of such a sequential ultrasonic wave simulator that incorporates a quite sophisticated numerical method was relatively straightforward.

Discretization of PDEs by finite elements, as well as by finite differences or finite volumes, almost exclusively gives rise to sparse matrices, because only mesh nodes lying in the same element have nonzero couplings between each other. This property suits very well for a domain decomposition parallelization approach, in which the original global

computational mesh is partitioned into a set of local sub-meshes. Each processor is then responsible for one or several sub-meshes. In such a parallelization approach, global matrices and vectors are replaced by local sub-matrices and sub-vectors. The global linear algebra operations needed in the numerical solution of PDEs can now be achieved by operating on *only* those sub-matrices and sub-vectors, plus inter-processor communication. Therefore, message passing is an appropriate programming model because the inter-processor communication is mostly in form of neighboring processors exchanging messages that contain mesh nodal values shared between them. Message passing also works well for the other form of frequently encountered inter-processor communication, which is all-to-all broadcast of one or several scalar values. This form of inter-processor communication is e.g. used during the calculation of the inner-product between two vectors that are distributed on multiple processors.

Sequential Diffpack simulators can be parallelized by two approaches; see e.g. [5]. The first so-called *linear-algebra-level* (LAL) approach aims to provide automatic parallelization of the CPU-intensive linear algebra operations that are needed in solving PDEs. The code related to this parallelization approach is organized into a small *add-on* library to the existing sequential Diffpack libraries. This add-on LAL library contains, among other things, different mesh partition strategies and high-level routines for inter-processor communication. An existing sequential Diffpack simulator can be straightforwardly parallelized using the LAL approach by inserting a few function calls to the LAL library.

The second parallelization approach, termed as *simulator parallel* (SP), is based on the LAL approach and aims to let Diffpack users easily develop parallel multilevel methods for solving PDEs, such as multigrid and domain decomposition methods. In this approach the existing sequential simulator is meant to be reused as a whole on each processor. The code related to the SP parallelization approach is also organized as an add-on library containing generic C++ class hierarchies, from which users can pick out ready-made classes and/or derive new C++ subclasses. In this way, the SP approach provides a flexible and structural way of building parallel PDE solvers with exceptionally good numerical efficiency, which is due to the incorporation of a multilevel method.

The LAL parallelization approach has been applied to the sequential ultrasonic wave simulator. As only 10 new lines of function calls to the LAL library were needed to be inserted into the over-1000-line original sequential code, the whole parallelization process was done under half an hour. The performance of this parallel ultrasonic wave simulator on a Linux-cluster will be reported in Section 6. We remark that the use of MPI functions inside the add-on LAL library

makes the parallel simulator completely portable, i.e., the source code used on the Linux-cluster is *identical* with that used on other parallel Unix platforms.

## 5. A Linux-cluster

### 5.1. Specification and configuration

We have recently built a beowulf-class Linux-cluster, using only off-the-shelf components. The cluster consists of 24 dual Pentium-III computing nodes, where all the 48 processors run at 500MHz and are equipped with 512KB level two cache. In addition, a separate single-CPU computer is used as the front-end and file server. All the computers have a 3com905B network interface card and are inter-connected with a 100Mbit/s ethernet network, through a 26-port Cisco Catalyst 2926 switch. Each computing node is equipped with 512MB of memory, amounting to a total of 12GB for the entire cluster. The whole system was purchased in the beginning of 2000 for a total cost of NOK 500,000, approximately US\$ 60,000. (For more information please refer to the web page [7].)

The Debian GNU/Linux operating system runs on the cluster, with a Linux-kernel of version 2.2.14 customised for the hardware. We have installed the MPI library Mpich, version 1.1.2 [18]. Moreover, all parallel jobs on the cluster are run through the PBS batch queue system [19]. We are currently running PBS version 2.2, patch level 7.

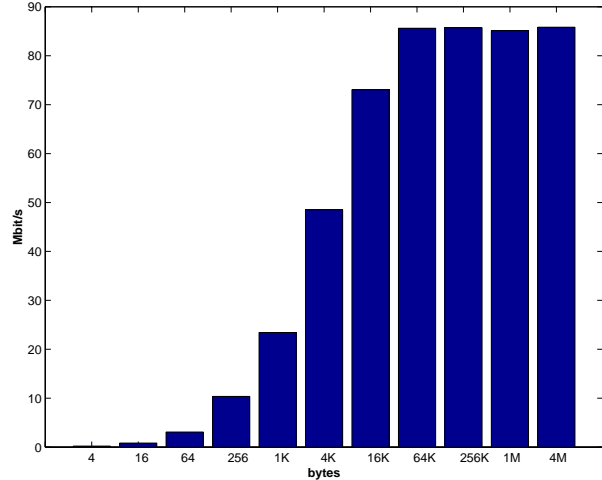
### 5.2. Benchmarks

Two standard benchmark tests are used to measure the performance of the Linux-cluster. The first one is the Top500-test from netlib, which is based on LU-factorization of a dense matrix. Using a matrix of dimension  $32000 \times 32000$  and a division into  $6 \times 8 = 48$  subparts, i.e. one per CPU, we achieved 9274.77 Mflops. The required block-size in this test was set to  $58 \times 58$ .

The second test is a simple ping-pong MPI-test, which sends a number of messages of increasing sizes, back and forth between two processors. This test measures the round-trip average time and the memory bandwidth of the network. When the test is run on two processors located on two different nodes, we have found out that the latency of the network on our cluster is approximately  $150\mu s$ , while the bandwidth is over 85Mbit/s for large enough messages; see Table 1 and Figure 1. Meanwhile, if the two processors chosen by the ping-pong test are located on the same node, the measurements for both latency and bandwidth are greatly improved; see Table 2 and Figure 2. In both Table 1 and Table 2, we denote the message size (in bytes) by  $S$ , round-trip time (in  $ms$ ) by  $T$  and memory bandwidth (in Mbit/s) by  $\beta$ .

**Table 1. Measurements of the ping-pong test running on 2 CPUs on different nodes**

$S$	4	64	1K	16K	256K	1M	4M
$T$	0.32	0.33	0.70	3.59	48.9	197	782
$\beta$	0.20	3.09	23.4	73.1	85.7	85.1	85.8



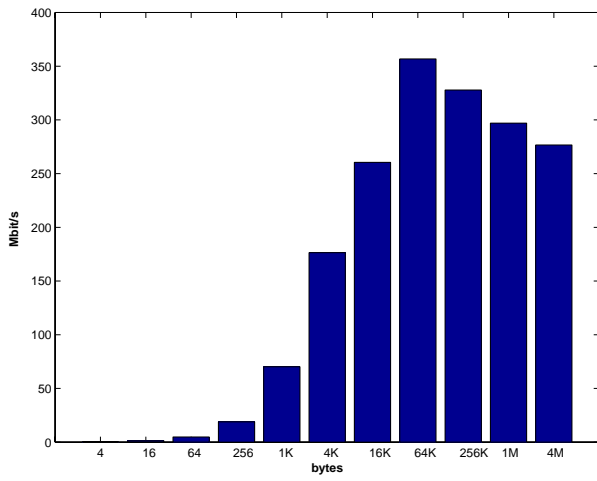
**Figure 1. Memory bandwidth for different message sizes, measured by the ping-pong test running on 2 CPUs on different nodes**

## 6. Numerical experiments and measurements

All simulations are done in a 3D spatial domain  $\Omega = [-0.004, 0.004]^2 \times [0, 0.008]$ , see Figure 3, which has a circular ultrasound transducer located on the face at  $z = 0$ , with the center of the transducer in origin. The transducer has a radius  $r = 0.002$  and  $\alpha$  in (6) is chosen such that the fourth zero of the Bessel function of order zero coincides with the edge of the transducer (i.e. at  $r = 0.002$ ). Moreover, we use  $\omega = 2.5\text{MHz}$ , speed of sound  $c = 1500\text{m/s}$

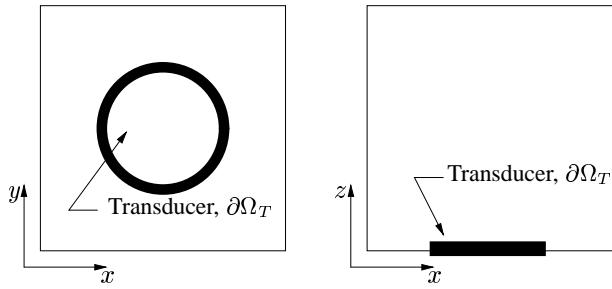
**Table 2. Measurements of the ping-pong test running on 2 CPUs on the same node**

$S$	4	64	1K	16K	256K	1M	4M
$T$	0.208	0.21	0.23	1.01	12.8	56.5	241
$\beta$	0.31	4.85	70.1	261	328	296	276



**Figure 2. Memory bandwidth for different message sizes, measured by the ping-pong test running on 2 CPUs on the same node**

and initial pressure  $p_0 = 1.01 \cdot 10^5$ . For the nonlinear model, we set the absorption parameter  $b = 6.8 \cdot 10^{-6}$  and the non-linearity parameter  $B/A = 2.5$ . These are the parameters experimentally derived for water. Finally, we use the density  $\rho_0 = 1000.0$ .



**Figure 3. Two perspectives of the 3D solution domain  $\Omega$  used for all the simulations**

In the following, we report some CPU measurements obtained on both our Linux cluster and an SGI Origin 2000 machine with 128 R10000 195MHz MIPS processors. We remark that the Origin 2000 machine scored 40.25GFlops in the Top500-test. All the CPU-times are given in seconds, and the C++ compiler and command-line options are

`g++ -O`  
and

`CC -64 -G 0 -O -mips4 -OPT:Olimit=4000`

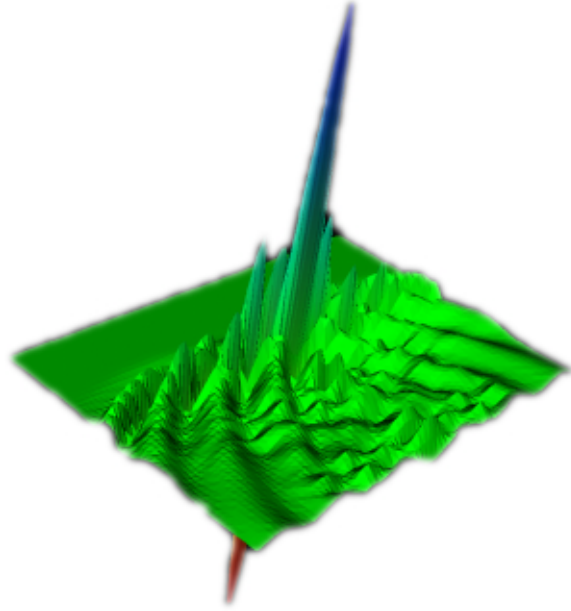
for the Linux-cluster and the Origin 2000 machine, respec-

tively. In Tables 3-5  $P$  denotes the number of processors and  $\eta$  denotes the speedup obtained by

$$\eta = \frac{T_1}{T_P}.$$

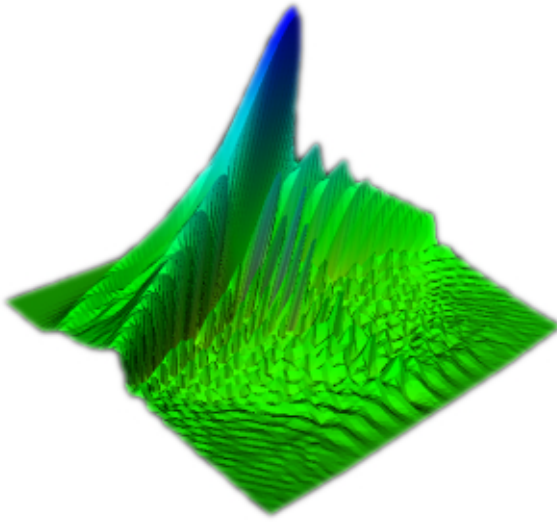
## 6.1. Measurements for the linear model

For the linear model (9), we discretize the spatial domain by  $96^3 = 884,736$  tri-linear elements. Each element has 8 nodes, amounting to a total of 912,673 degrees of freedom. The problem is solved from  $t = 0$  until  $t = 5.0 \cdot 10^{-6}$ . Two snapshots of such simulations are shown in Figures 4-5. When multiple processors are used, the spatial domain is partitioned by planes perpendicular to the axes. The sub-meshes are non-overlapping and of strictly the same size.



**Figure 4. A snapshot of a Bessel beam travelling in a linear medium**

Table 3 and Figure 6 show the CPU-times and the associated speedup results for different numbers of processors in use. We note that the numerical method for the linear model is an explicit scheme, where the computational kernel at each time step is matrix-vector multiplication. The inner-processor communication is of the form that one processor only exchanges information with its immediate neighbors, no all-to-all communication occurs. Together with perfect load balance, this explains why the Linux cluster performs very well even at its full capacity, whereas, the measurements of the Origin2000 machine are more or less affected by the high word load.



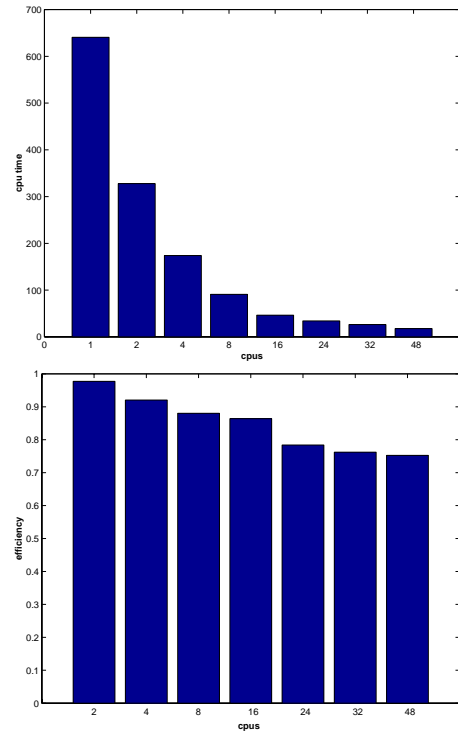
**Figure 5. A snapshot of a X-wave travelling in a linear medium**

## 6.2. Measurements for the nonlinear model

For the nonlinear model (1,3), we use tri-quadratic elements where the total number of degrees of freedom is 1,030,301. Here we adopt a general mesh partition approach that is much more flexible than that being used for the parallel linear simulations. More specifically, we use the METIS [12] package to first get an unstructured non-overlapping partition. Then a member function of the add-on LAL library enlarges each sub-mesh, such that a layer of element overlap arises between neighboring sub-meshes. The overlap is necessary for the parallel nonlinear simulations, because systems of linear equations need to be solved

**Table 3. CPU-measurements of simulations for the linear model**

$P$	Sub-mesh	Origin2000		Linux-cluster	
		CPU	$\eta$	CPU	$\eta$
1	$96 \times 96 \times 96$	944.83	N/A	640.7	N/A
2	$48 \times 96 \times 96$	549.21	1.72	327.8	1.95
4	$48 \times 48 \times 96$	282.75	3.34	174.0	3.68
8	$24 \times 48 \times 96$	155.01	6.10	90.98	7.04
16	$24 \times 48 \times 48$	80.41	11.8	46.35	13.8
24	$32 \times 24 \times 48$	65.63	14.4	34.05	18.8
32	$24 \times 24 \times 48$	49.97	18.9	26.27	24.4
48	$24 \times 24 \times 32$	35.23	26.8	17.74	36.1



**Figure 6. CPU-times and efficiency measurements obtained on the Linux-cluster associated with simulations for the linear model**

by the processors collaboratively.

In each Newton iteration, the linear system of equations, which is non-symmetric, is solved iteratively by the stabilized Bi-Conjugate-Gradient method. Convergence of this method is considered to be achieved if the discrete  $L_2$ -norm of the residual is below  $10^{-8}$ . For the Newton iterations, convergence is reached if the difference between the two latest solutions, divided by the latest solution, is smaller than  $10^{-8}$ .

Table 4 and Figure 7 show the measurements for running one time step of the nonlinear simulation. Since the discretization of the entire spatial domain requires more than 512MB, measurement associated with a single processor of the Linux cluster is not available. Although this is not a problem for the Origin 2000 machine, which has a shared memory of 24GB, we have used the CPU-time obtained on two processors as the reference for calculating speedup results for both the machine types. In this way Table 4 brings a fair comparison.

We remark that the relatively poorer speedup results of Table 4, when compared with those of the linear simulations, are primarily due to the general mesh partition approach and the element overlap layer between neighbor-

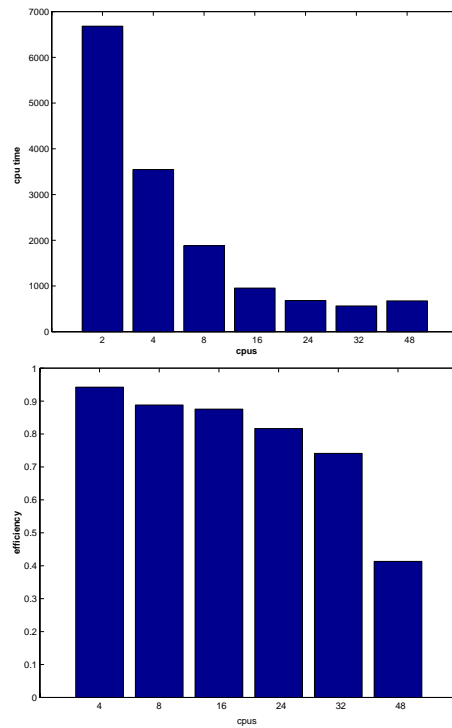
**Table 4. CPU-measurements of simulations for the nonlinear model**

$P$	Origin2000		Linux-cluster	
	CPU	$\eta$	CPU	$\eta$
2	8670.8	N/A	6681.5	N/A
4	4628.5	3.75	3545.9	3.77
8	2404.2	7.21	1881.1	7.10
16	1325.6	13.0	953.89	14.0
24	1043.7	16.6	681.77	19.6
32	725.23	23.9	563.54	23.7
48	557.61	31.1	673.77	19.8

ing sub-meshes. Secondly, the stabilized Bi-Conjugate-Gradient method involves inner-product calculation between two vectors, requiring all-to-all reduction operations of inter-processor communication. Such all-to-all communication becomes very costly on the Linux cluster, when being used close to its full capacity. That is why measurements of the Origin 2000 machine become better than those of the Linux cluster for large numbers of processors. We also recall that for the current configuration of our cluster, the two processors on the same node have to share the *same* ethernet cable connecting towards the switch, thus making the inter-processor communication extraordinarily expensive when more than 24 processors are in use. The situation may be improved if each node is equipped with a second network card and ethernet cable towards the switch. To see when the communication overhead makes the parallel nonlinear simulations unprofitable for the current configuration, we also list in Table 5 more detailed measurements for the Linux cluster obtained on between 32 and 48 processors. The speedup results are scaled against the CPU-time obtained on two processors. In addition to the maximum CPU-time consumption among all the processors, which we have consistently used in previous tables, we also listed the average and minimum CPU-time consumption among the processors. This is to show the disparity in the measurements due to load-imbalance.

## 7. Conclusion and discussion

In this paper, we have included different aspects of the numerical simulation of 3D ultrasonic waves. A parallel simulator has arisen easily from its sequential counterpart in the framework of Diffpack, and the source code is completely transparent for the underlying parallel platform. For such parallel 3D ultrasonic wave simulations, we have made an interesting observation that a low-cost Linux-cluster is able to deliver fully comparable computing power as an



**Figure 7. CPU-times and efficiency measurements obtained on the Linux-cluster associated with simulations for the nonlinear model**

SGI Origin 2000 machine. It is reflected by not only the absolute CPU-time measurements, but also the speedup capability. This is especially true for simulations where no all-to-all inter-processor communication occurs. Undoubtedly, clusters fabricated with high-end myrinet-based networks are capable of delivering better parallel performance, as pointed in e.g. [11]. We believe, however, that low-end Linux-clusters will remain a usable and cost-effective option for solving certain types of PDEs in parallel.

**Acknowledgements.** The authors thank the anonymous referees for their valuable comments. This work has received support from The Research Council of Norway (Programme for Supercomputing) through a grant of computing time.

## References

- [1] J. Abate, P. Wang and K. Sepehrnoori, *Parallel compositional reservoir simulation on a cluster of PCs*. <http://topeka.cpge.utexas.edu/papers/Cluster/Cluster.html>.

**Table 5. Detailed simulation measurements for the nonlinear model, using from 32 to 48 processors of the linux cluster**

$P$	CPU-time	$\eta$	Avg. CPU	Min. CPU
32	563.5	23.71	546.4	519.8
34	541.0	24.70	523.7	491.1
36	518.0	25.80	495.4	444.0
38	555.3	24.06	497.5	440.4
40	625.4	21.37	438.0	334.8
42	734.2	18.20	598.2	434.4
44	601.2	22.23	483.1	389.7
46	966.2	13.83	667.5	373.7
48	673.8	19.83	522.9	389.9

- [2] B.G. Allan, *Performance of OVERFLOW on Coral*. <http://www.icase.edu/~allan/coral/>.
- [3] D.J. Becker, T.L. Sterling, D. Savarese, J.E. Dorband, U.A. Ranawak, and C.V. Packer, *Beowulf: a parallel workstation for scientific computation*. Proceedings of the International Conference on Parallel Processing 1995.
- [4] R. Buyya (Ed.), *High Performance Cluster Computing, Vol 1*. Prentice Hall 1999.
- [5] X. Cai, *Two object-oriented approaches to the parallelization of Diffpack*. Proceedings of the HiPer'99 Conference (1999), pp. 405–418.
- [6] Diffpack Home Page: <http://www.diffpack.com>.
- [7] Diplopodus Home Page: <http://communalis.ifi.uio.no/diplopodus>.
- [8] B. Engquist and A. Majda, *Absorbing boundary conditions for the numerical simulation of waves*. Mathematics of computation (1977), vol. 31(139).
- [9] P. D. Fox and S. Holm, *Effects of parameter mismatch in ultrasonic Bessel transducers*. Proc. IEEE Norwegian Signal Processing Symposium NORSIG'99 (1999).
- [10] S. Holm, *Bessel and conical beams and approximation in annular arrays*. IEEE Transaction on Ultrasonics, Ferroelectrics and Frequency control (1998), vol. 45(3).
- [11] G. S. Karamanos, C. Evangelinos, R. C. Boes, R. M. Kirby and G. E. Karniadakis, *Direct numerical simulation of turbulence with a PC/Linux cluster: fact or fiction?* Proceedings of the Super Computing '99 conference.
- [12] G. Karypis, V. Kumar, *METIS: Unstructured graph partitioning and sparse matrix ordering system*. Department of Computer Science, University of Minnesota, Minneapolis/St. Paul, MN, 1995.
- [13] H.P. Langtangen, *Computational Partial Differential Equations – Numerical Methods and Diffpack Programming*. Springer-Verlag 1999.
- [14] J.Y. Lu, *Designing limited diffraction beams*. IEEE Transaction on Ultrasonic, Ferroelectrics and Frequency control (1997), vol. 44(1).
- [15] S. Makarov and M. Ochmann, *Nonlinear and thermo-viscous phenomena in Acoustics, Part II*. ACUSTICA – acta acustica (1997), vol. 83, pp. 197–222.
- [16] T.L. Sterling, J. Salmon, D.J. Becker and D.F. Savarese *How to build a Beowulf*. MIT Press, 1999.
- [17] Å. Ødegård, P. Fox, S. Holm, and A. Tveito, *Finite Element Modelling of Pulsed Bessel Beams and X-Waves using Diffpack*. To appear in Proceedings of 25th International Acoustical Imaging Symposium, Bristol, United Kingdom, 2000.
- [18] Mpich Home Page: <http://www-unix.mcs.anl.gov/mpi/mpich>.
- [19] PBS Home Page: <http://pbs.mrj.com>.