Environment. Technology. Resources. Rezekne, Latvia
Proceedings of the 13th International Scientific and Practical Conference. Volume 2, 210-213

Implementation of the Difference Scheme for Absorption Equation Type Problems Applying Parallel Computing Technologies

Maksims Zigunovs
Turība University
Riga, Latvia
maksims.zigunovs@inbox.lv

***Abstract* - This paper describes a way of parallel algorithm technology usage for analyzing physical processes parabolic differential problems on the surface. This analysis determine the temperature distribution on the surface. Such analysis can fit calculation of Maxwell and Maxwell-Stokes equations and can be focused on mathematical models that can be reduced to the absorption or diffusion-convection-reaction equations with the initial and boundary conditions of different order (1st, 2nd, 3rd order of boundary conditions). Parallel computing technologies usage provides an acceleration possibilities of mentioned calculations in different way and effect depending of parallel technology type and method combinations used during the calculations.**

Keywords* - *Parabolic equation, difference scheme, boundary conditions, parallel calculation.

I. Introduction

In the modern technology the large computing power is available.

The current CPU (central processor unit) processor frequency reaches up to 4 GHz. GPU (graphical processor unit) and graphics processor cards have reached up to 1,000 units in a single map, and each processor frequency reaches up to 1GHz.

This means that it is possible to create a computer system that could be designed for solving non-stationary physical phenomena modelling problem of the three-dimensional space.

II. Parallel computing technologies

The essence of parallel computing is to split the calculation procedure into several calculation nodes. Based on which technology will be used, a computer program must be created that will use the most optimal aspects of its technology to realize the best possible after use time and resources. There is the fact that parallelization of calculations takes place within a single processor (multi-core processor), then it must be ensured that each core OS-based process will use the same RAM memory area to read matrix elements.

A. Shared Memory

Shared Memory [3] is a memory that shared between processor core processes as a "shared access memory," which includes both hard disk space (non-removable) and RAM memory. A block diagram of this kind of memory type can be represented as follows (Fig. 1.):

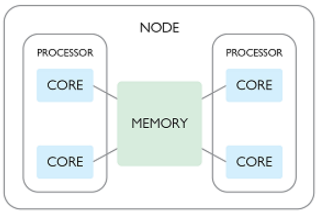

Fig. 1. Shared Memory Systems.

Since processor / processor kernels of one or more processor / processor cores typically use equivalent processors / processor cores, it may be considered that in the algorithm's parallelization process it would be appropriate for each processor / processor core to execute approximately the same number of operations by executing identical code snippets. Based on modern CPU architecture (fast execution and time delays for switching between processes), it is objectively necessary to split the parallel-generated code into large parallel code fragments to reduce

Online ISSN 2256-070X
https://doi.org/10.17770/etr2021vol2.6633

inefficient CPU usage for processor / processor kernel switching between OS processes.

B. *Distributed Memory*

The Distributed Memory [3] means that the "relative total" memory, which is distributed over computers on the network and is connected to one indivisible computing system. This system can represent computers-curators and computer-based data processing computers. Such computers can be both stationary and portable, both virtual and processor tiles. An intermediate communication tool is text messaging. Each computer has its own processor, RAM memory and hard disk (not always). Each computer receives signals from the host computer. These signals are code fragments of the computer program or their execution parameters. Moreover, it is possible to set up a dataset for sending and receiving data between computers. The logical structure of this kind of memory allocation technology can be represented as follows (Fig. 2.):

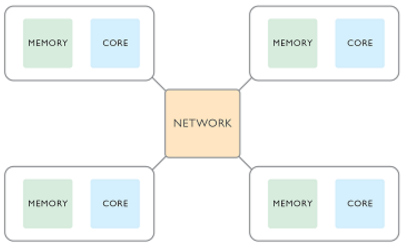

Fig. 2. Distributed Memory.

The method of parallelization of this algorithm would be useful only for the parallelism of the significant fragments of non-interlinked code. Otherwise, sending / receiving data between nodes can undermine the utility of the technology. Therefore, it is advisable to include in the calculation process the computers that are connected with the Internet connection within one router in order to take computer data interchange in relatively short time.

C. *Shared Memory and Distributed Memory symbiosis*

This type of memory combines the two-way division of executable instructions. In practice, here is a combination of two pre-reviewed technologies [3]. The logical structure of such memory allocation as follows (Fig. 3.):

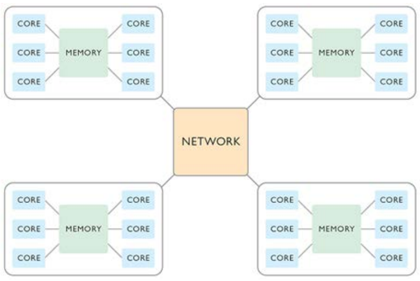

Fig. 3. Shared Memory and Distributed Memory symbiosis.

In using this method, the recommendations of the previous sections should be taken into account when performing the parallelization of the algorithm.

III. Parallel Calculations

Mathematical modeling is based on the application of numerical methods. The paper was written by overlooking the finite difference method. The essence of this method is the unbroken division of a space into a number of nodal points, where the central point of each node is the node average value representation. As more points in the node network, as the node is less sized, so the depth of the calculation or efficiency is increased. The final difference method means that before the calculation process is started, at least the first approximation values must be known in the entire discrete area as consideration (boundary conditions must be known as well). One of the most popular point stencils is the 5-point stencil that combines calculable nodes as follows (Fig. 4):

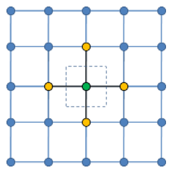

Fig. 4. 5 point stencil.

which means that the value of each point is equivalent to the sum of the values of 4 surrounding points (up, bottom, right and left)

An example is the absorption or Helmholtz 2D equations (Eq. 1):

$$G_x D_x \frac{\partial^2}{\partial x^2}\varphi(x,y) + G_y D_y \frac{\partial^2}{\partial y^2}\varphi(x,y)$$

$$\pm\sigma\varphi(x,y) = 0 \qquad (1)$$

The difference equation matrix method allows to build a solving equation for each unknown point of the grid. These equations contain the current for the certain equation surrounding point coefficients in a way shown in Eq 2:

$$k_{ij-1}u_{ij-1} + k_{i-1j}u_{i-1j} + k_{ij}u_{ij} + k_{ij+1}u_{ij+1} + k_{i+1j}u_{i+1j} = 0 \qquad (2)$$

It is possible to find out that for each point of the node, using the i and j indexes, there exist the coefficients of the existing point and four surrounding each P_{ij} points (multipliers) (Eq 3):

$$k_{ij-1} = \frac{G_y D_y}{{h_y}^2};\; k_{i-1j} = \frac{G_x D_x}{{h_x}^2};$$

$$k_{ij1} = -\left(2\frac{G_x D_x}{{h_x}^2} + \frac{G_y D_y}{{h_y}^2} \mp \sigma\right); \qquad (3)$$

$$k_{ij+1} = \frac{G_y D_y}{{h_y}^2}; k_{i+1j} = \frac{G_x D_x}{{h_x}^2};$$

Using the above, it is possible to draw up a matrix equation (a system of absolute sets of i and j index equations for each node) that looks as follows (Fig. 5.):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & . & 0 & = & \psi 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & . & 0 & = & \psi 2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & . & 0 & = & \psi 3 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & . & 0 & = & \psi 4 \\ k1\,x1 & 0 & 0 & k4\,x4 & k5\,x5 & k6\,x6 & 0 & 0 & 0 & k10\,x10 & . & 0 & = & 0 \\ 0 & k2\,x2 & 0 & 0 & k5\,x5 & k6\,x6 & k7\,x7 & 0 & 0 & 0 & . & 0 & = & 0 \\ 0 & 0 & k3\,x3 & 0 & 0 & k6\,x6 & k7\,x7 & k8\,x8 & 0 & 0 & . & 0 & = & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & . & 0 & = & \psi 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & . & 0 & = & \psi 9 \\ . & 0 & 0 & 0 & k5\,x5 & 0 & 0 & 0 & k9\,x9 & k10\,x10 & . & 0 & = & 0 \\ . & . & . & . & . & . & . & . & . & . & . & 0 & = & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & . & 1 & = & \psi n \end{bmatrix}$$

Fig. 5. Linear equation system for difference equation solving process using matrix method.

In order to make calculation time enhancement the modified time step was created by using absolute and relative errors of calculated temperature for l+1 time moment in order to speed up calculations by managing simulation time step based on absolute and relative errors of the calculations.

For the "x" unknown set classical calculation method the Ax=B matrix equation has be to used and then "$A^{-1}Ax=A^{-1}B$ => $x=A^{-1}B$". Matrix inversion calculation process is very "expensive" in terms of necessary HPC cluster needs in order to significantly reduce the time spent on parallel computing. If the matrix has Dim points in one direction so Dim^4 operations should be performed to find out the matrix inversion.

The disadvantages of this method are mainly related to computerized rounding and long calculation time and required computer resources. Special "expensive" by the necessary computing time and computer resources, it is a matrix inversion operation (A^{-1}) (matrix inversion method). An example is a 2D matrix, which has 10000 elements in the each direction. This means that this matrix has 10^8 elements. Each element has a data type. If each element occupies 8 bytes, then the matrix takes a minimum computer memory $8x10^8$B = 8x105KB = 8x102MB = 800MB, which is required to store the matrix. "Good" matrix inversion algorithm requires the number of operations, which is equal to the number of matrix elements squared. In 10000x10000 matrix case, it would be 10^{16}, or 10^{10} million operations. Considering that today's non-threading CPU can calculate 10^9 operations per second, the total calculation time is 10^7 seconds = 2778 hours = 115 days. Parabolic differential equation problems need at least 10000 time step calculation. This means that the total time consumed one problem needs 115x10000 days equal to 3151 years.

Looking at the [2] reference, the solution uses a locked-circuit calculation based on the ADI method. It means that the time step is divided into 2 stages, in which the calculations take one of the directions as an open scheme and 2 directions as an enclosed scheme, where at the top the index "l" means the time slice index (Fig. 6.). In order to elaborate the parallel calculation method, the difference scheme was decomposed by space and time by applying ADI [1] method assumes calculation half steps using two calculation directions with initial and border conditions:

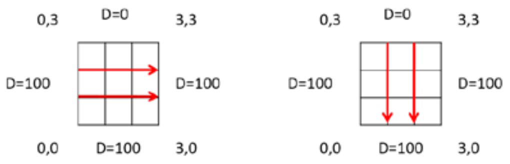

Fig. 6. ADI calculation split in directions.

$$u_{i,j}^{l+\frac{1}{2}} - u_{i,j}^{l} = \frac{1}{2}\tau\alpha\left(\frac{u_{i-1,j}^{l+\frac{1}{2}} - 2u_{i,j}^{l+\frac{1}{2}} + u_{i+1,j}^{l+\frac{1}{2}}}{(\Delta x)^2} + \frac{u_{i,j-1}^{l} - 2u_{i,j}^{l} + u_{i,j+1}^{l}}{(\Delta y)^2}\right) \quad (4)$$

$$u_{i,j}^{l+1} - u_{i,j}^{l+\frac{1}{2}} = \frac{1}{2}\tau\alpha\left(\frac{u_{i-1,j}^{l+\frac{1}{2}} - 2u_{i,j}^{l+\frac{1}{2}} + u_{i+1,j}^{l+\frac{1}{2}}}{(\Delta x)^2} + \frac{u_{i,j-1}^{l+1} - 2u_{i,j}^{l+1} + u_{i,j+1}^{l+1}}{(\Delta y)^2}\right) \quad (5)$$

Each partstep equation contains 3 unknowns in only one direction for a 5 point stencil (examples: 5 point stencil for 2D space and 7 point stencil for 3D space) and all other direction unknowns' values are taken from the previous partstep. Since there are only 3 unknowns and they are located in one direction it is possible to construct a 3 diagonal matrix. It means that there are totally separated 3 diagonal matrix calculations for each directions' index values that do not have unknowns series within current partstep. For example, if direction X is the direction for current partstep, that means that it is possible to solve a 3 diagonal matrix with unknowns in direction X for each Y index separately.

The calculations of the ADI method include non-interconnected calculations, which can be separated between several unit nodes. This means that ADI calculations make it possible to use a modern processor with 100% power per computer, using a common memory area. This can significantly reduce the time of one-step calculations. For ADI method implementation in 3D space, there have to be 3 equations instead of just 2.

IV. Enhancements

In order to make calculation time enhancement the modified time step was created by using absolute and relative errors of calculated temperature for l+1 time moment in order to speed up calculations by managing simulation time step based on absolute and relative errors of the calculations.

Absolute and relative error calculations for X direction (same for Y and Z) as follows in the beginning of each next iteration starting from second one in order to:

```
For[x=1,x<=DimX,x++,
ErrorAbsoluteX[[x]]=0.00;
ErrorRelativeX[[x]]=0.00;
For[y=1,y<=DimY,y++,
For[z=1,z<=DimZ,z++,
ErrorAbsoluteX[[x]]=ErrorAbsoluteX[[x]]+(Prognos
eX[[x]][[y]][[z]]-
CurrentStepResult[[x]][[y]][[z]])^2;
ErrorRelativeX[[x]]=ErrorRelativeX[[x]]+CurrentS
tepResult[[x]][[y]][[z]]^2;
];
];
ErrorRelativeX[[x]]=ErrorAbsoluteX[[x]]/ErrorRel
ativeX[[x]];
ErrorX[[x]]=ErrorRelativeX[[x]];
];
```

Finding the maximal error for X direction (same for Y and Z) is implemented as follows:

```
ErrorAbsoluteXYZ=0;
ErrorRelativeXYZ=0;
ErrorXYZ=0;
For[x=1,x<=DimX,x++,
If[
ErrorAbsoluteXYZ<MatricaErrorAbsoluteX[[x]],
ErrorAbsoluteXYZ=MatricaErrorAbsoluteX[[x]];
];
If[
ErrorRelativeXYZ<MatricaErrorRelativeX[[x]],
ErrorRelativeXYZ=MatricaErrorRelativeX[[x]];
];
If[
ErrorXYZ<MatricaErrorX[[x]],
ErrorXYZ=MatricaErrorX[[x]];
];
];
```

It is possible to calculate next iteration time step length based on previously found out ErrorXYZ value. As follows:

```
If[
(ErrorXYZ³toltol/2),t1=t;t=t/tkoef;If[t<tmin,t=t
min];];
If[
(ErrorXYZ£toltol/2)ß(ErrorXYZ³toltol/5),t1=t;t=t
];
If[
(ErrorXYZ£toltol/5),t1=t;t=t*tkoef;If[t>tmax,t=t
max];];
```

, where ErrorXYZ = maximal error between predicted and calculated results, toltol = error threshold, tau1 = temporary variable, tau = time step, taukoef = coefficient of time step changing, taumin = minimal time step, taumax = maximal time step.

When the next iteration time step is found out it is possible to calculate prediction for the next iteration calculation results as follows:

```
For[x=1,x<=DimX,x++,
For[y=1,y<=DimY,y++,
For[z=1,z<=DimZ,z++,
PrognoseY[[x]][[y]][[z]]=CurrentStepResult[[x]][
[y]][[z]]+t/t1*(CurrentStepResult[[x]][[y]][[z]]
-PrevY[[x]][[y]][[z]]);
PrognoseX[[x]][[y]][[z]]=CurrentStepResult[[x]][
[y]][[z]]+t/t1*(CurrentStepResult[[x]][[y]][[z]]
-PrevX[[x]][[y]][[z]]);
PrognoseZ[[x]][[y]][[z]]=CurrentStepResult[[x]][
[y]][[z]]+t/t1*(CurrentStepResult[[x]][[y]][[z]]
-PrevZ[[x]][[y]][[z]]);
PrevY[[x]][[y]][[z]]=CurrentStepResult[[x]][[y]]
[[z]];
PrevX[[x]][[y]][[z]]=CurrentStepResult[[x]][[y]]
[[z]];
PrevZ[[x]][[y]][[z]]=CurrentStepResult[[x]][[y]]
[[z]];
];
];
];
```

V. Aprobation

The author aprobated the described approach and received same results applying different methods of calculations for 3rd order boundary conditions shows in figure 7:

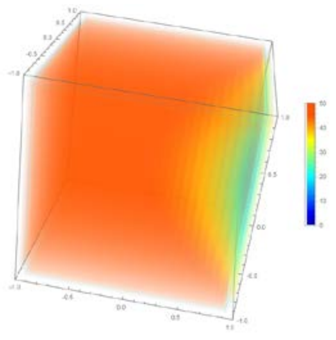

Fig. 7. 3D calculated temperature disctibution using 3rd order boundary conditions.

REFERENCES

[1] I.J.D. Graig, A.D.Sneyd (1988), AN ALTERNATING-DIRECTION IMPLICIT SCHEME FOR PARABOLIC EQUATIONS WITH MIXED DERIVATIVES, Comput. Math. Applic. Vol. 16, No. 4, pp. 341-350.

[2] F. MAMPAEY (1989), A numerical technique to increase the stability of the ADI method in solidification simulation, Journal of Computational and Applied Mathematics 28 297-308.

[3] Siegfried Benkner, Viera Sipkova (2003), Exploiting Distributed-Memory and Shared-Memory Parallelism on Clusters of SMPs with Data Parallel Programs, International Journal of Parallel Programming,Vol.31, No.1, February.