USING CUDA ARCHITECTURE FOR COMPUTER SIMULATIONS OF THERMOMECHANICAL PHENOMENA

***Grzegorz Michalski**[1]**, Norbert Sczygiol**[2]**, Siergiei Leonov**[3]*

[1,2] Institute of Computer and Information Sciences, Czestochowa University of Technology
Częstochowa, Poland
[3] I.I. Polzunov Altai State Technical University
Altai, Russia
[1] grzegorz.michalski@icis.pcz.pl, [2] norbert.szczygiol@icis.pcz.pl, [3] sergey_and_nady@mail.ru

Abstract. This paper presents a simulation of the casting solidification process performed on graphics processors compatible with nVidia CUDA architecture. Indispensable for the parallel implementation of a computer simulation of the solidification process, it was necessary to modify the numerical model. The new approach shown in this paper allows the process of matrix building to be divided into two independent phases. The first is independent from the nodal temperature values computed in successive time-steps. The second is performed on the basis of nodal temperature values, but does not require a description of the finite element mesh. This phase is performed in each time step of the simulation of the casting solidification process. The separation of these two phases permits an effective implementation of the simulation software of the casting solidification process on the nVidia CUDA architecture or any other multi-/manycore architecture. The use of GPUs nVidia for the implementation of a computer simulation of the solidification process significantly reduced the waiting time for results. In the course of computer simulations important speedup of the computations was observed.

***Keywords:** numerical modeling, solidification, distributed and parallel processing, CUDA*

1. General information

Problems faced by engineers and scientists often require carry out the complicated and time-consuming computer simulations, on the basis of which changes can be incorporated into the analyzed object (its geometric model). Such tasks require the most modern solutions in the field of computer science, the most popular solution is parallel processing. Over the last few years the dynamic development of multicore processors has had a direct impact on the availability of advanced high performance solutions for scientists and engineers. Graphic processors (GPUs - Graphics Processing Unit) are increasingly being used in high performance computations. A single GPU has a theoretical computational power several times higher than the fastest available CPU. Figure 1 shows the increase in the theoretical computational power of graphic processors and general purpose processors since 2003.

Computations using a GPU do not require any additional hardware equipment. A GPU has been designed to efficiently perform mathematical operations on two-dimensional matrices and should be capable of performing numerical simulations. The application of modern multi- and many-core architectures, such as graphic processors, for computational purposes allows for huge systems of equations, which may consist of many millions of variables, to be solved very fast. The algorithms used for solving such systems of equations are based primarily on standard arithmetic operations on matrices and vectors [1].

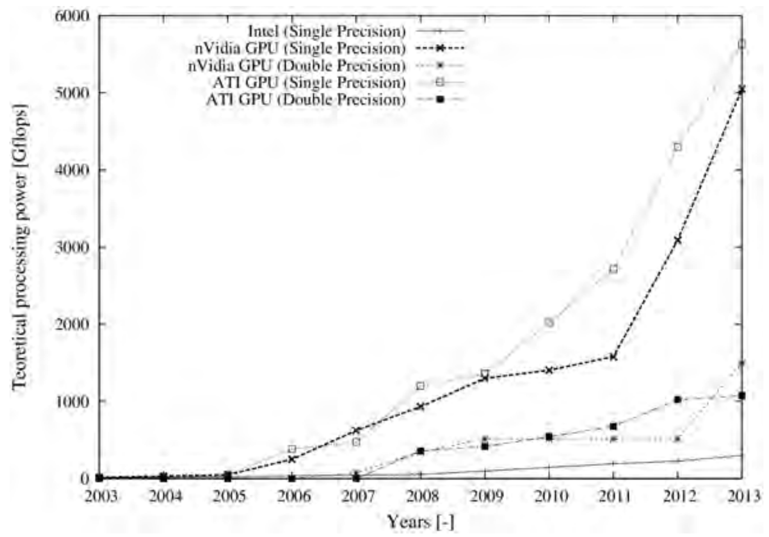

Fig. 1. An increase in the theoretical computational power of the GPU and CPU since 2003

The authors present a computer simulation of the casting solidification process (enthalpy formulation). For processes of this type, a system of equations is built from scratch in each time step. The matrix of the coefficients of this system of equations and a vector of right sides are built on the basis of several factors: a description of the finite elements mesh, the boundary conditions, the material properties and the results obtained from previous time steps. These operations require a large amount of data in each time step of the simulation to be processed. The specific character of this data (mainly the finite element mesh) makes it difficult to efficiently parallelize the matrix building process. This difficulty derives from the lack of regularity of the data that are to be processed. This problem can be clearly seen in the case of GPUs and SIMT architecture (Single Instruction Multiple Thread), where the conditional execution of certain parts of the program code has a strong negative impact on the efficiency of computations. An additional problem is the need to synchronize the write operations performed during the construction of the system of equations [2-4].

2. Building the system of equations on graphic processors

While performing simulations of solidification processes the system of equations has to be built many times (for each time step). Owing to the specific nature of the graphic processors architecture, this can constitute a real problem and can significantly decrease the efficiency of computations. This disadvantage relates to the slow data transfer from the global memory of the graphic devices (device memory) to the system memory (host memory). An additional problem is to use and synchronize mechanism and conditional instructions.

As the data transfer from the system memory to the global memory of the graphic devices should be reduced to a minimum, a good solution is to transfer the description of the finite element mesh to the graphic device global memory and build the system of equations using the graphic processor [4-6]. However, it should be remembered that the finite element meshes that are used nowadays often consist of a few million nodes. Transferring such a large amount of data is a very time-consuming process and this data can reduce the limited resources of the device.

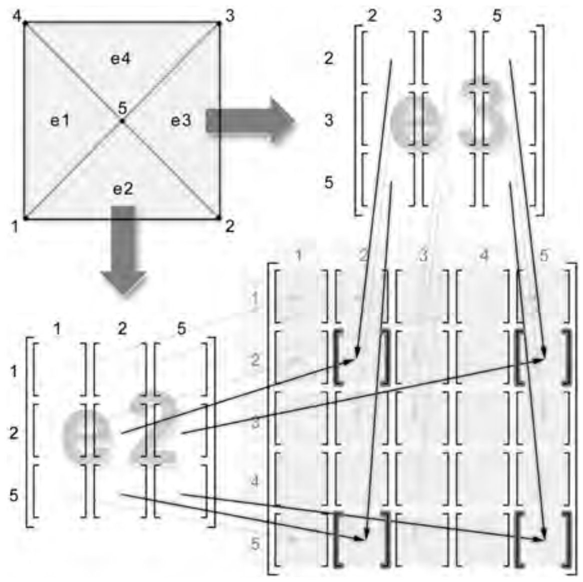

Fig. 2. Process of building of the global coefficient matrix

The parallelization operation of the matrix building process on the basis of the finite element mesh is very complicated. This process requires a lot of conditional instructions and synchronize blocks, which causes a significant decrease in the performance of computations made on the graphic processor.

The values of elements in *i*-row of the coefficient matrix depend on the finite elements, which include the node connected with the node with index *i*. A single node in the finite element mesh can belong to several finite elements. These indices can only be read from the finite element mesh. Figure 2 shows the process of building the global coefficient matrix. Non-zero elements of the global coefficients matrix are determined as the sum of several values which depend on those finite elements which include the pair of nodes with indexes corresponding to the indexes of the row and column of those elements.

The approach presented in this paper uses a modified way of building the system of equations which divides this process into two separate phases. The first requires only information from the finite element mesh and is performed only once at the beginning of the computer simulation. The second phase is performed in each time step and is dependent on the nodal temperature values and enthalpy from previous steps.

2.1. Two-step building of the system of linear equation

The system of equations resulting from the transformation of a differential equation, which describes the process of solidification into the numerical model using the Finite Element Method, can be finally written in matrix formulation (1) [7]

$$\mathbf{M}\mathbf{H}^{\mathbf{n}+1} + \Delta t \mathbf{K}^{n} \mathbf{T}^{n+1} = \mathbf{M}\mathbf{H}^{n} + \Delta t \mathbf{b}^{n+1} \tag{1}$$

The elements of (1), which are built on the basis of the finite element mesh description are: matrix **M** (2) and conductivity matrix **K** (3) [8]:

$$\mathbf{M}^{e} = \int_{\Omega^{e}} \mathbf{N}^{T}\mathbf{N} d\Omega \tag{2}$$

$$\mathbf{K}^{e}(T) = \int_{\Omega^{e}} \lambda(T) \nabla^{T}\mathbf{N} \cdot \nabla \mathbf{N} d\Omega \tag{3}$$

Matrix **M** is independent from the nodal temperature and enthalpy values in subsequent time steps. This matrix may be determined at the beginning of the simulation, before the first time step is executed and can be used in the same form to the end of computer simulation.

In contrast, conductivity matrix **K** is a temperature function. The element which causes this dependency is thermal conductivity coefficient λ. The value of the thermal conductivity coefficient is determined during the building of the matrix of coefficients for each finite element. The value of thermal conductivity coefficient λ for the finite element is calculated as the average value of the thermal conductivity coefficients determined for the nodes belonging to that finite element. After building the local coefficient matrices, they are assembled into a global matrix of coefficients **K**, which contains elements that are the sum of the products (their factors are different thermal conductivity coefficient λ). Such a situation makes it impossible to separate this part of the matrix, which is temperature-dependent, from that part which is built on the basis of information contained in the finite element mesh.

In order to find a solution to this problem the authors have developed an alternative approach to the building of the system of linear equations. If thermal conductivity coefficient λ is not dependent on the spatial coordinates, it can be taken out before the integral. Finally equation (3) assumes the form

$$\mathbf{K}^e(T) = \lambda(T) \int_{\Omega^e} \nabla^T \mathbf{N} \cdot \nabla \mathbf{N} d\Omega \tag{4}$$

After integration (4), conductivity matrix **K** takes the form (5), where $\mathbf{K}^e$ is a local coefficient matrix for the finite element, **A** the surface area of the finite element, λ_j are a thermal conductivity coefficient of the corresponding j-node of the finite element, C_{ij} are a coefficients which depend on the spatial coordinates of the nodes belonging to the finite element.

$$\mathbf{K}^e = \frac{1}{4A}\begin{bmatrix} \lambda_1(C_{21}^2 + C_{31}^2) & \lambda_1(C_{21}C_{22} + C_{31}C_{32}) & \lambda_1(C_{21}C_{23} + C_{31}C_{33}) \\ \lambda_2(C_{21}C_{22} + C_{31}C_{32}) & \lambda_2(C_{22}^2 + C_{32}^2) & \lambda_2(C_{22}C_{23} + C_{32}C_{33}) \\ \lambda_3(C_{21}C_{23} + C_{31}C_{33}) & \lambda_3(C_{22}C_{23} + C_{32}C_{33}) & \lambda_3(C_{23}^2 + C_{33}^2) \end{bmatrix} \tag{5}$$

This approach allows the two parts of conductivity matrix **K** to be separated. It involves the introduction into the local matrices, values of the thermal conductivity coefficient λ determined for the nodes and not for the finite elements as in the original approach. It permits the removal of thermal conductivity coefficient λ before the parenthesis in each element of global matrix **K**. After these steps the solidification equation in matrix formulation (1) takes the form described in equation

$$\mathbf{MH}^{\mathbf{n}+1} + \Delta t \lambda^n \mathbf{K}^* \mathbf{T}^{n+1} = \mathbf{MH}^n + \Delta t \mathbf{b}^{n+1} \tag{6}$$

where λ is the diagonal matrix of the thermal conductivity coefficient for each node of the finite element mesh, determined on the basis of nodal temperatures from the appropriate time-step, $\mathbf{K}^*$ is a matrix of coefficients built on the basis of the finite element mesh description.

This approach allows the process of building the global coefficient matrix to be divided into two phases. The first phase is independent from the nodal temperatures values, and thus simultaneously independent from the time steps. This phase is performed only at the beginning of the computer simulation. In this stage, matrix $\mathbf{K}^*$ is built on the basis of information from the finite element mesh. Additionally, matrix **B** (calculated on the basis of the boundary conditions), diagonal matrix **M** and the vector (calculated on the basis of the boundary conditions) with values necessary to build the vector right sides in each time step are created in this stage.

After this step, the information stored in the mesh of finite is no longer required for the process of computer simulation. The second phase of the matrix building process consists of determining conductivity coefficient λ for each node. As a result of this step diagonal matrix λ is created. Since conductivity coefficient matrix λ is temperature-dependent, this operation must be performed in each subsequent time-step.

This approach simplifies the process of building the global matrix of coefficients on graphic processors by allowing matrix $\mathbf{K}^*$ to be built on a general purpose processor.

2.2. Computer experimental numerical verification

Calculations were performed for the two different regions with the use of the five finite element meshes of the each region. Sizes of these meshes ranged from several thousand to several hundred thousand nodes.

Times of simulations performed with the GPUs were compared to the results obtained with the authorial CPU-based sequential software. This software uses the approach to building the system of equations which was presented in previous section. The computer simulation on the graphics processor was performed on the two graphic devices equipped with an nVidia GT 200 processor with 240 CUDA cores which has a peak performance of about 1 Tflops in single float operations (Operating System: Debian Wheezy, CUDA version 3.2).

The casting material is aluminium alloy with the addition of copper and the mold is assumed to be made from steel. The initial temperature of pouring was equal to 960 K and this value was the value of the initial condition temperature for the region of the cast in the performed calculations. The initial temperature of the mold was set at 560 K. Newton's boundary condition is assumed on all the surfaces of the mold, assuming heat exchange with the environment with a coefficient equal to 100 W/m^{2}K. The ambient temperature has a value of 300 K in all boundary conditions. The heat exchange between the casting and the mold is obtained from a type IV boundary condition, which assumes the heat exchange through the insulation layer of the conductivity coefficient of the separating layer to be 800 W/m^{2}K.

Figure 3 shows the speed up obtained in computer simulations, for three different analyzed regions, depended on the size of the task (number of nodes in the finite element mesh). In small tasks the speed up is minor. The speed-up growth increases with the size of the finite element mesh. This can be explained by the time taken to copy the data from the system memory to GPU memory and to copy the results into the system memory from the graphic device memory (first and last time-step). Results were obtained using the authorial software, created from scratch without outside libraries.

No differences were noted in the process of the computer simulation implemented on a general purpose processor and graphics processor. Slight differences were noted between the temperatures values at the nodes of the finite element mesh on the results obtained on the GPU and a general purpose processor. These differences are due to the implementation of the standard floating-point on the GT200 processor and the chosen method of rounding. In the computer computation we never have a certainty that the (A + B) + C = A + (B + C).

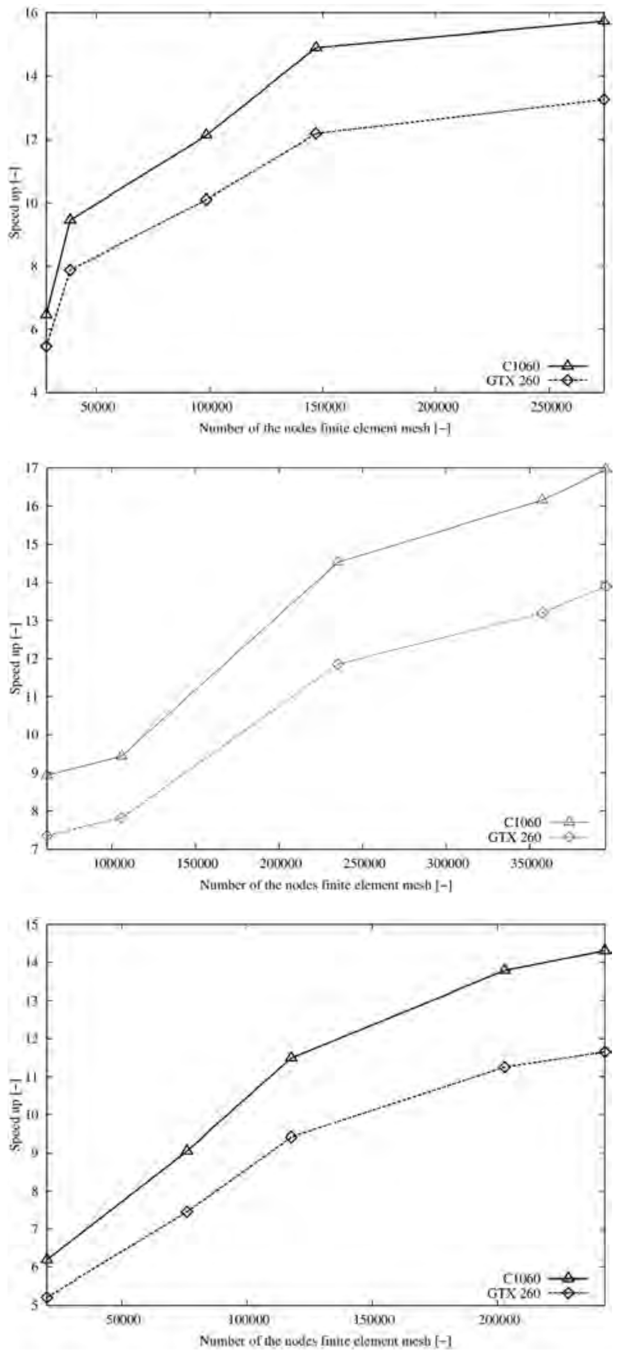

Fig. 3. Speed up obtained with use of GPU-based simulation software for each analyzed region

Conclusions

In this article, the authors present a new method of parallelization for the computer simulation of the casting solidification process and its implementation on graphics processors compatible with CUDA architecture. The proposed method divides the process of building the system of equations into the two phases. This solution improves the efficiency with which the available system resources are used. The great advantage of the developed solution is that it is easily adapted to the different architectures of multicore processors.

The speed-up observed during the computer simulation of the casting solidification process confirms that the use of graphic processors in engineering simulations brings significant benefits. It was also noted that with an increase in the number of unknowns in the system of equations, the time needed to solve such a system increases linearly. At the same time, as the size of the task increases so does the speed-up observed in the computations with the graphic processor.

The modifications to the numerical model produce slight differences in the temperatures in the nodes. However, these are just minor differences that do not affect the simulation of the casting solidification process. These differences result from the different way in which the value of the coefficient of thermal conductivity sensor is determined for the node, and not for the finite elements. This leads to a speed-up in computations performed in a sequential way (on the CPU).

Having regard to the above results, it can be stated that using graphic processors in engineering simulations seems to be a viable solution as this approach can significantly reduce the time needed for research.

References

[1] Meng H.T., Nie B.L., Wong S., Macon C., Jin J.M., GPU accelerated finite-element computation for electromagnetic analysis, Antennas and Propagation Magazine, IEEE 2014, 56, 2, 39-62.

[2] Che S., Boyer M., Meng J., Tarjan D., Sheaffer J.W., Skadron K., A performance study of general--purpose applications on graphics processors using CUDA, Journal of Parallel and Distributed Computing 2008, 10, 68, 1370-1380.

[3] nVidia, CUDA C Best Practices Guide. Design Guide v. 6.0, February 2014.

[4] Pospichal P., Schwarz J., Jaros J., Parallel genetic algorithm solving 0/1 knapsack problem running on the GPU, 16th International Conference on Soft Computing MENDEL 2010, Brno University of Technology, 64-70.

[5] Lee C., Wei X., Kysar J.W., Hone J., Measurement of the elastic properties and intrinsic strength of monolayer graphene, Science 2008, 321, 385-388.

[6] nVidia, Optimization. OpenlCL Best Practices Guide, 2010.

[7] Zienkiewicz O., The Finite Element, Volume I, the Basis 5th Edition, Butterworth-Heinemann, Oxford 2003.

[8] Sczygiol N., Szwarc G., Wyrzykowski R., Numerical modeling of equiaxed structure formation during solidification of a two-component alloy, 2nd European Coference on Computational Mechanics, Kraków 2001, 820-821.