Publishing House
AKAPIT

COMPUTER METHODS IN MATERIALS SCIENCE
Informatyka w Technologii Materiałów

Vol. 16, 2016, No. 3

EVALUATION OF EFFICIENT COMPUTATIONAL WORK DIVISION IN PARALLEL MONTE CARLO GRAIN GROWTH ALGORITHM

MATEUSZ SITKO*, ŁUKASZ MADEJ

AGH University of Science and Technology, 30 Mickiewicza Av., Krakow, 30-059 Poland
**Corresponding author: msitko@agh.edu.pl*

Abstract

Implementation of parallel version of the Monte Carlo (MC) grain growth algorithm is the subject of the present paper. First, modifications of the classical MC grain growth algorithm required for the parallel execution are presented. Then, schemes for the MC space division between subsequent computational threads/nodes are discussed. Finally, implementation details of different parallelization approaches based on OpenMP and MPI are presented and compared.

Key words: Monte Carlo, OpenMP, MPI, parallelization, grain growth

1. INTRODUCTION

Metals and their alloys are widely used as a constructional materials in many practical industrial sectors related to automotive, marine or aerospace. The main reason for that is associated with possibility to acquire final products characterized by sophisticated exploitation properties obtained via precisely controlled thermo-mechanical processing. In that case, the initial microstructure of investigated material as well as its further evolution plays crucial role as it is directly related to final mechanical properties.

Major groups of mechanisms controlling microstructure evolution during thermo-mechanical processing are phase transformation and recrystallization processes (Humphreys & Hatherly, 2004). The latter are thermally activated phenomena, which result in restoration of the deformed microstructure and reduction of strain hardening effects. Unfortunately, finding correlation between recrystallization parameters and material properties is not an easy task, which usually involves series of experimental investigations (Sun et al., 2013). The procedure is rather expensive and time consuming. One of the solution, is to support mentioned experimental investigations by a series of numerical analysis. Increasing availability of high performance computing units and related economical aspects make the simulation a powerful investigation tool not only for scientific but also practical industrial applications. Numerical simulations of recrystallization phenomena (e.g. static recrystallization SRX) can be realized with numerical approaches based on the conventional continuum (Dong et al., 2016; Su et al., 2016) as well as discrete models (Sieradzki & Madej, 2013). The later ones are considered within the preset work to develop a full field model of microstructure evolution.

Development of a full field model usually requires generation of explicit representation of initial material microstructure. A lot of works dedicated to this subject have been recently published in the scientific literature (Madej et al., 2011; Scholtes et al., 2016). The most straight forward way to obtain such a microstructure representation is to process microscopy images (Madej et al., 2011). As a result, an

ISSN 1641-8581

exact representation of all micro scale features can be obtained. Unfortunately, the approach in 3D space is still very complex and time consuming (Goins & Holm, 2016). Thus, often numerical approaches based on e.g. Cellular Automata or Monte Carlo are used to provide synthetic microstructures both in 2D and 3D (Ivasishin et al., 2006; Rollett & Manohar, 2004). Unfortunately, often the drawback of these approaches is again excessive computing time in 3D computational space.

That is why, the goal of the present paper is to evaluate possibilities to decrease the computing time by parallelization of the Monte Carlo grain growth algorithm.

The main idea of Monte Carlo grain growth method is based on minimization of Energy – *H* in material. Energy of one particular cell is higher when a cell has more neighbours with different id. Energy is calculated twice, once for initial cell orientation, and then for new orientation (random value from *Q-1*, available states). Reorientation of cell state is accepted with following formula:

$$w = \begin{cases} \exp\left(\dfrac{-\Delta G_i}{k_b T}\right) & for\, \Delta G > 0 \\ 1 & for\, \Delta G \le 0\,, \end{cases} \tag{1}$$

where: $\Delta G_i = H_{before\ reorientation} - H_{after\ reorientation}$, k_b - Boltzmann constant, *T*- temperature.

```
PARALLEL PART
#pragma omp parallel for default(shared) private(i, j, head, P, counter,
analyzedCell, neighbours, wynik) schedule(static, (noOfCells))

for(j = 0; j < Number of points in parallel part; j++)
{
       //Randomly select point
       point P = pointGen->nextPoint(thread_num, head, counter, result);

       Check MC grain growth algorithm transition rules for selected point

}
SEQUENTIAL PART
for(int x = 0; x < Number of points in sequential part; x++){
{
       //Randomly select point
       point P = pointGen->nextPointSingleThread();
       Check MC grain growth algorithm transition rules for selected point

}
```

Fig. 1. *Main MC algorithm loop*

Detailed information on sequential version of the MC algorithm have already been widely discussed in the scientific literature see e.g. (Williamson & Delplanque, 2016; Mason et al., 2015; Mason, 2015). Thus, present work focuses mainly on modifications that have to be introduced to parallelize the MC algorithm using two different approaches based on:

– single physical processor using Open Multi-Processing technique (OpenMP),
– multiple computational nodes using Message Passing Interface (MPI) standard.

2. PARALLEL VERSION OF THE MC GRAIN GROWTH ALGORITHM BASED ON OPENMP

When parallelization of any computational problem is considered the major concern is to develop efficient communication procedures to exchange data between parts of the analysed domain. In the present work authors developed an algorithm within OpenMP standard, where MC computational domain was divided into equal volumetric parts. Because two threads cannot use the same MC cell in the computational space at the same time, authors additionally decided to create buffer zones at the interfaces of each analysed subdomain, which are computed in a sequential manner. Implementation details with modification of classical MC algorithm are presented in figure 1. Parallelization of main MC grain growth algorithm loop was based on core element of OpenMP interface for thread creation: the *pragma omp parallel for*.

Proposed solutions have been tested on the MC spaces with different geometries and different sizes in X, Y and Z directions as presented in table 1.

Initially, the MC space was divided into equal parts along the X-axis and classical Moore neighborhood type as well as periodic boundary conditions have been selected.

where: $P(w)$ – speedup for w computational nodes, T_i – computational time for one node, $T_{\|}(w)$ – computational time for w nodes.

Table 1. *MC spaces in X, Y and Z directions with overall size of 9 and 20 million cells.*

X Size	3000	9000	1000	300	300	100	20000	1000	2000	100	100
Y Size	3000	1000	9000	100	300	300	1000	20000	100	2000	100
Z Size	1	1	1	300	100	300	1	1	100	100	2000

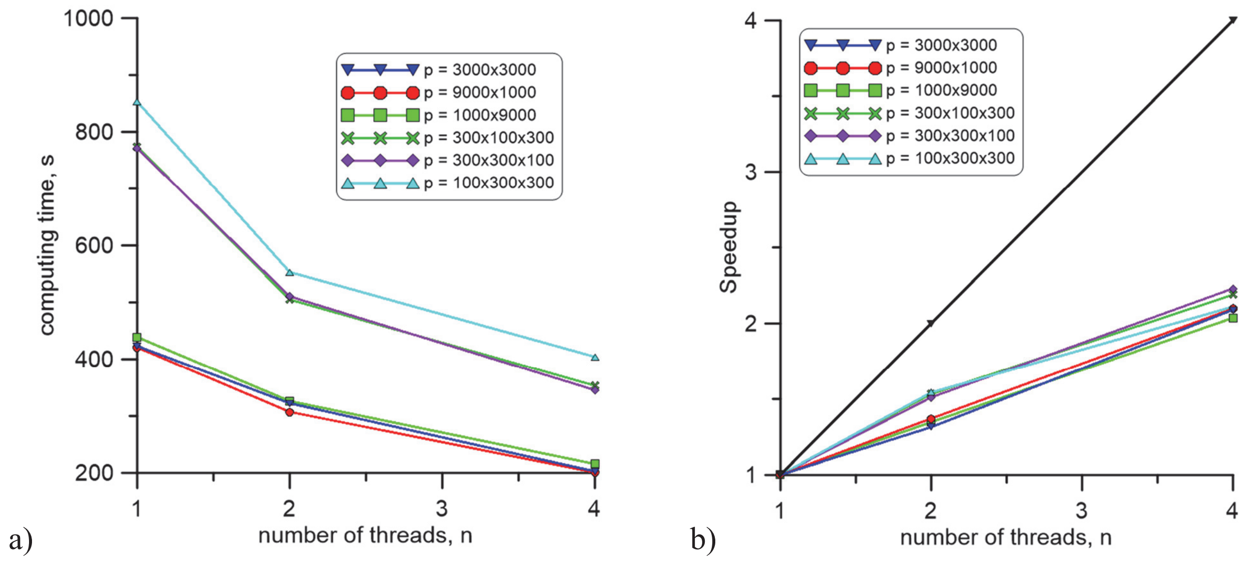


Fig. 2. *Results from simulations with 9 million MC cells with Moore neighborhood and OpenMP technique a) simulation time, b) corresponding computation speedup.*

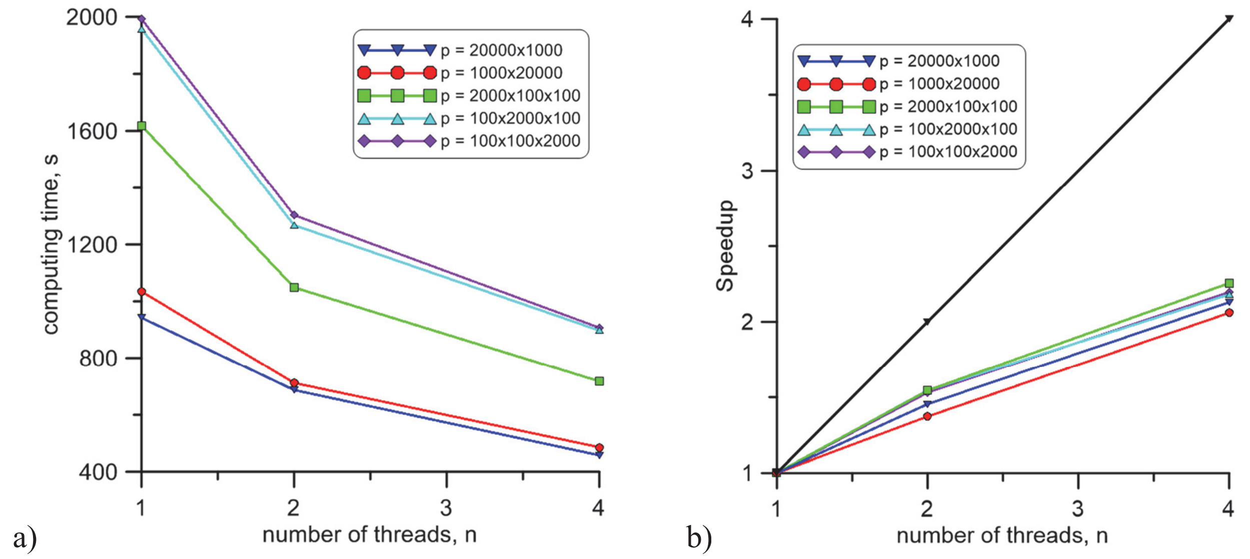


Fig. 3. *Results from simulations with 20 million MC cells with Moore neighborhood and OpenMP technique a) simulation time, b) corresponding computation speedup.*

Capabilities of the developed OpenMP version of the MC model are shown in figure 2 and figure 3 by presenting computational times and speedups calculated as:

$$P(w) = T_i / T_{\|}(w), \qquad (2)$$

Different computation time reduction presented in figure 2a and figure 3a for the same amount of cells in computational domain are caused by inefficient distribution of work related to selected domain decomposition along the X axis. To eliminate influence of space shape a series of modifications in

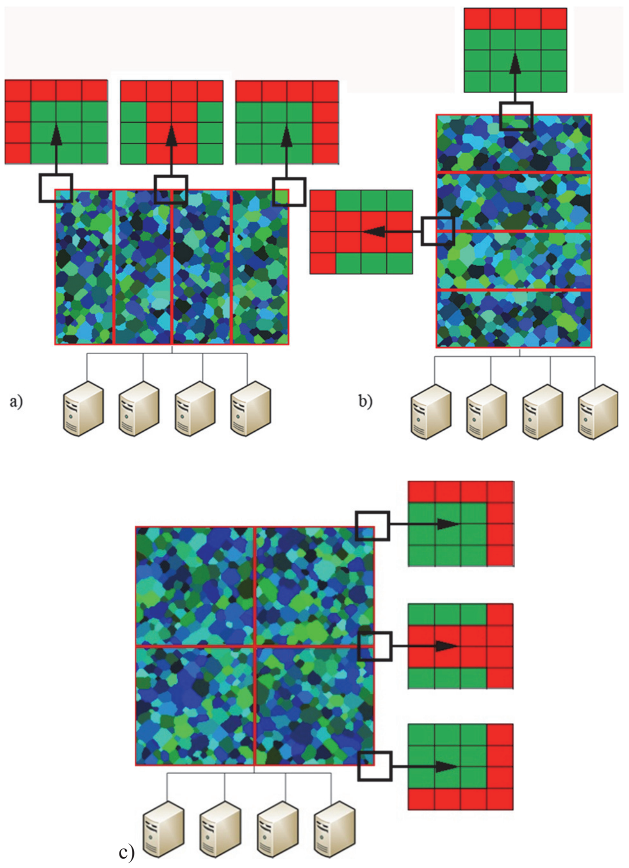


Fig. 4. Space division schemes for 2D computational domain a) horizontal (2D-case0), b) vertical (2D-case1), equal pieces (2D-case2).

```
case 0:{

Calculate number of cell in buffer zone
Calculate number of each node
Allocate memory for all lists

for(each cell in X dimension){
        for(each cell in y dimension){
                for(each cell in z dimension){

                        if(cell belongs to buffer zone){
                                Add new point to sequential points list
                        }
                        else{
                                Add new point to parallel points list
                        }
                }
        }
}
break;
}
```

Fig. 5. Example of MC cell selection at the interfaces of different subdomains.

functions responsible for efficient and flexible division of computational domain have been introduced. This is crucial as the geometry of the investigated MC space representing material microstructure can be significantly different e.g. the microstructure after rolling processes will be elongated along X-axis, microstructure after extrusion on the other hand will be elongated along Y-axis and in another scenario only square region of material will be subjected for investigation. To account for these variations, authors proposed different MC space division schemes: horizontal, vertical and equal squares. Schematic representation of each division scheme for 2D is presented in figure 4 and for 3D in figure 5, respectively.

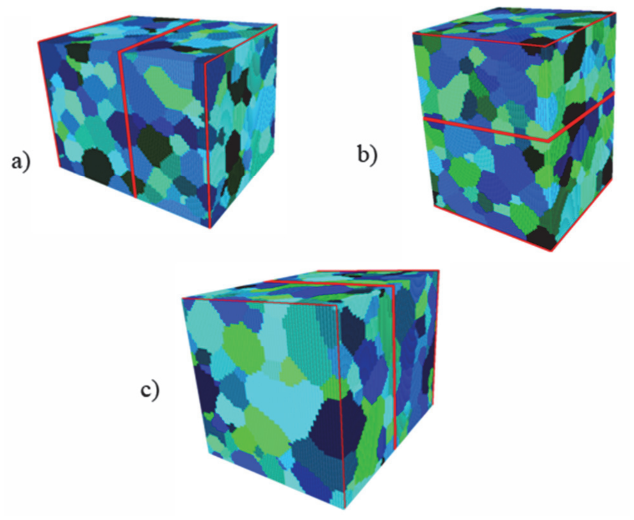
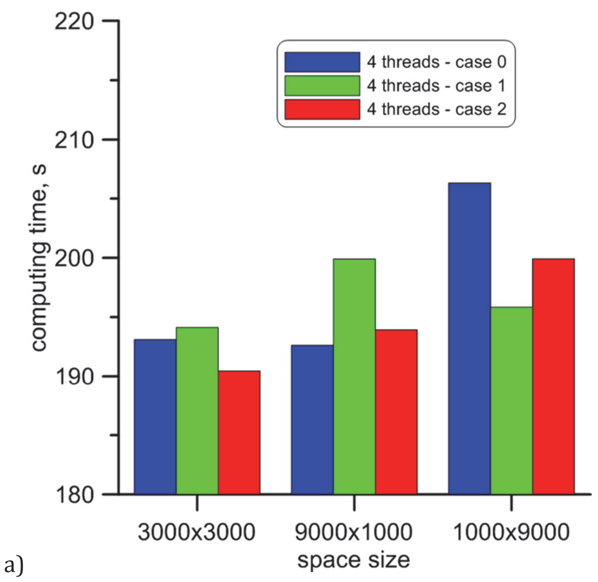


Fig. 6. Space division schemes for 3D computational domain a) horizontal OX (3D-case0), b) vertical OY (3D-case1), c) horizontal OZ (3D-case2).

Moreover additional function for periodic and absorbent boundary condition both in 2D and 3D computational domains have to be introduced. Implementation details in the form of pseudo-code for one of division scheme are presented in figure 6.

To highlight the requirement for proper computational space division a set of additional analysis was realized. Different division schemes (figure 4 and figure 5) have been used during investigation of grain growth realized in 3 different MC space geometries. Results from realized set of simulations are presented in figure 7 and figure 8.

authors have implemented procedure for automatic selection of proper space division scheme based on investigated geometries of computational domain. However, in general, because computational speedups obtained with the use of OpenMP standard



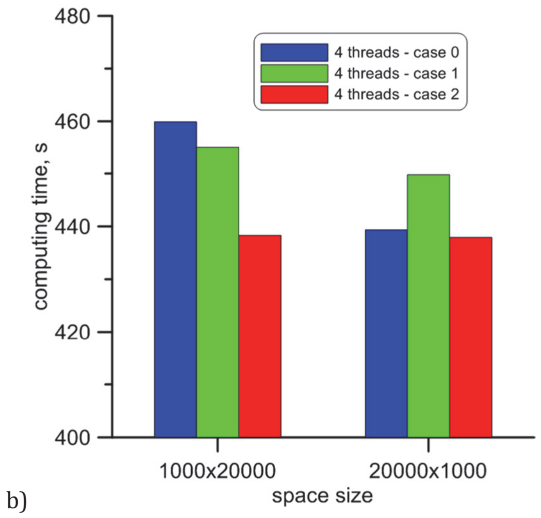
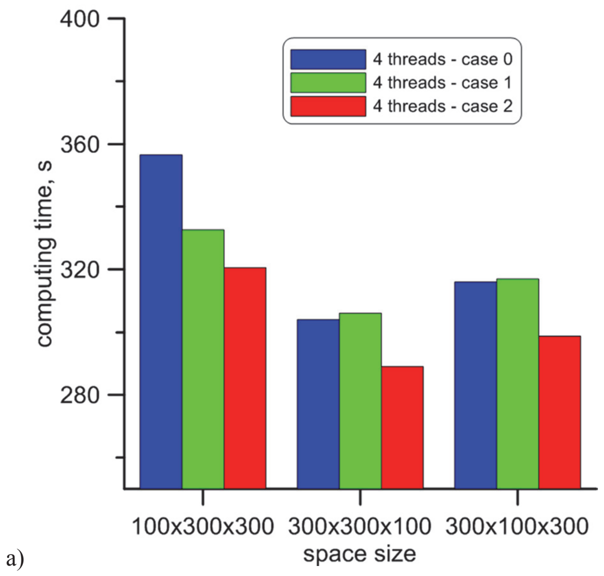


Fig. 7. Comparision of computational time for different space geometries and different division schemes a) 9 million, b) 20 million cells in 2D space.



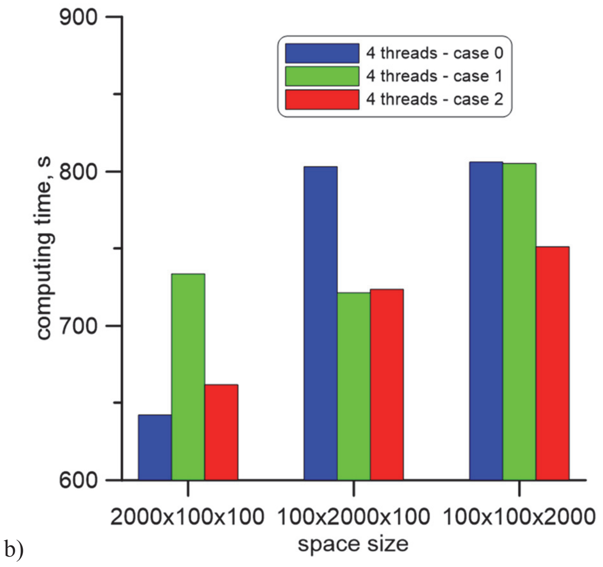


Fig. 8. Comparision of computational time for different space geometries and different division schemes a) 9 million, b) 20 million cells in 3D space.

When 3D simulations are considered (figure 8) proposed space division schemes have still not been efficient for all microstructures shapes. Thus, authors proposed additional division cases presented in figure 9.

Results from all implemented division schemes are presented in figure 10.

As seen in figure 10 in some cases simulation time can be reduced by over 2 minutes, only by selection of proper space division scheme. That is why are not satisfactory, another approach based in MPI was used in the present work.

3. PARALLEL VERSION OF MC GRAIN GROWTH ALGORITHM WITH MPI

When parallelization based on MPI standards is introduced, implementation concepts have to be additionally extended. Parallel execution with MPI requires running the application multiple times and

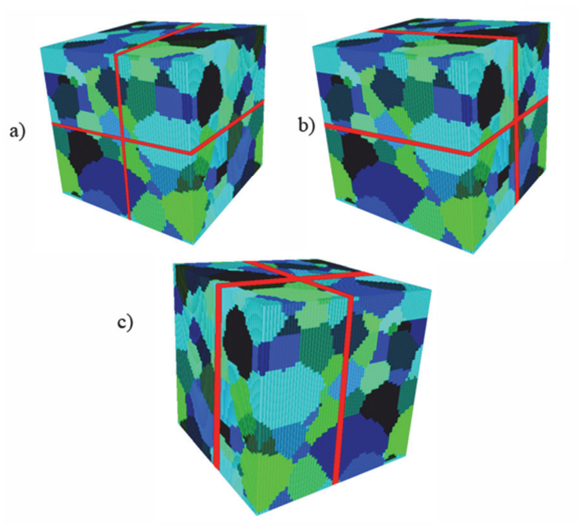


Fig. 9. Space division schemes for 3D computational domain a) XY (3D – case3), b) YZ (3D – case4), c) XZ (3D – case5).

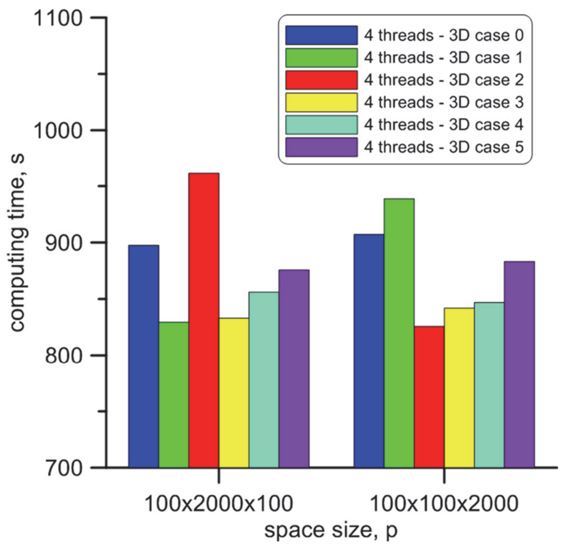


Fig. 10. Comparision of computational time for different space geometries and different division schemes for 20 million cells in 3D space.

At the end of the MC step:

- Update boundary cells list.
- Send information about new cell state to all neighbors nodes.
- Wait for information from neighbors nodes.
- Receive new states of boundary cells.

Fig. 11. Communication between different computational nodes.

also development of efficient communication protocols between application instances. Communication between subsequent nodes work in parallel manner. To fulfil these requirements, authors have introduced series of source code modification namely:

– New data tape *mpi_cell_type* was proposed (new data tape stores general information about the MC cell state).
– At each computational node MC space was extended with additional boundary cells to store information about states of cells from neighboring nodes.
– Communication was handled by non-blocking send (*MPI_Isend*) and blocking receive (*MPI_Recv*) (figure 11)
– Automatic selection of proper space division scheme based on geometries of computational domain.

Results from simulations with MPI standard and mentioned modifications, realized with different number of nodes from 1-20 for investigated space geometries are presented in figure 12 and figure 13. During simulation a single physical work station with 2 Intel Xeon X5650 processor with 12 threads and 12MB cache memory each was used. However, the solution is prepared for the Zeus cluster architecture where more than 20 MPI nodes can be used. Because all simulation were performed on different number of MPI processes working on a single node, transfer exchange rates were limited by the memory bandwidth.

As seen in figure 13, the MPI implementation allows to obtain speedup around 12 times. To eliminate influence of any computational environment aspects, each calculation was recalculated five times for the same set up and eventually averaged data are presented. The speedup reduction for simulation with different nodes number is most probably related with cache memory limitations on a single work station. In the future work, authors plan to performed simulation on a grid platform, to eliminate this problem. On the other hand authors plan to used OpenCV and CUDA in modern computer centers with graphic card units to evaluate possibilities of alternative solution for parallelization.

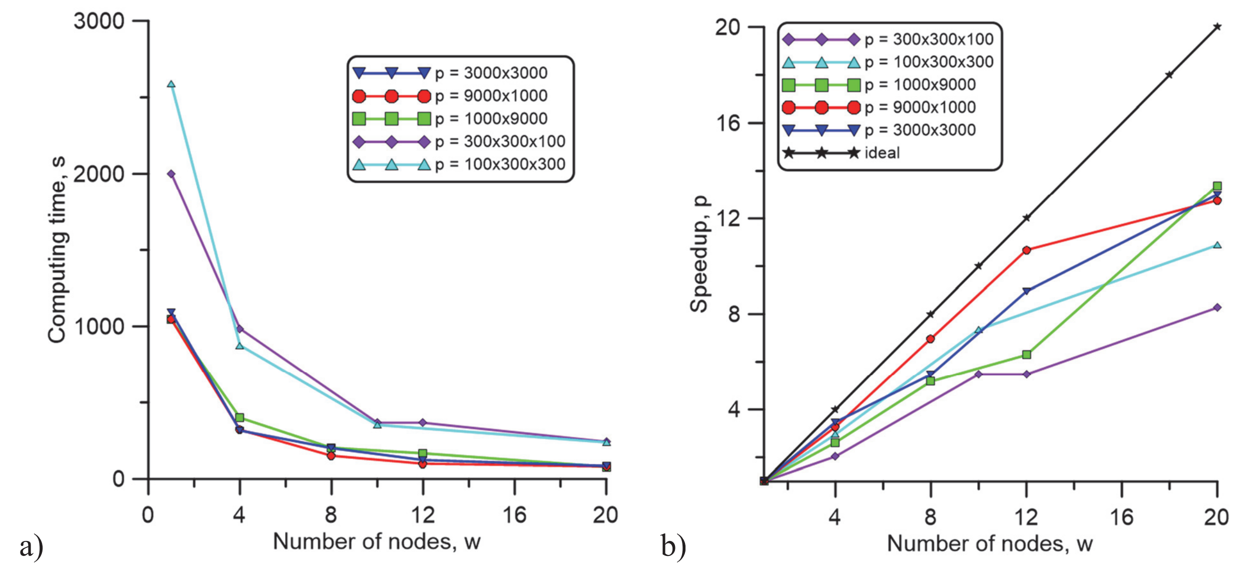


Fig. 12. *Results from simulations with 9 million MC cells and MPI standard a) simulation time, b) corresponding computation speedup.*

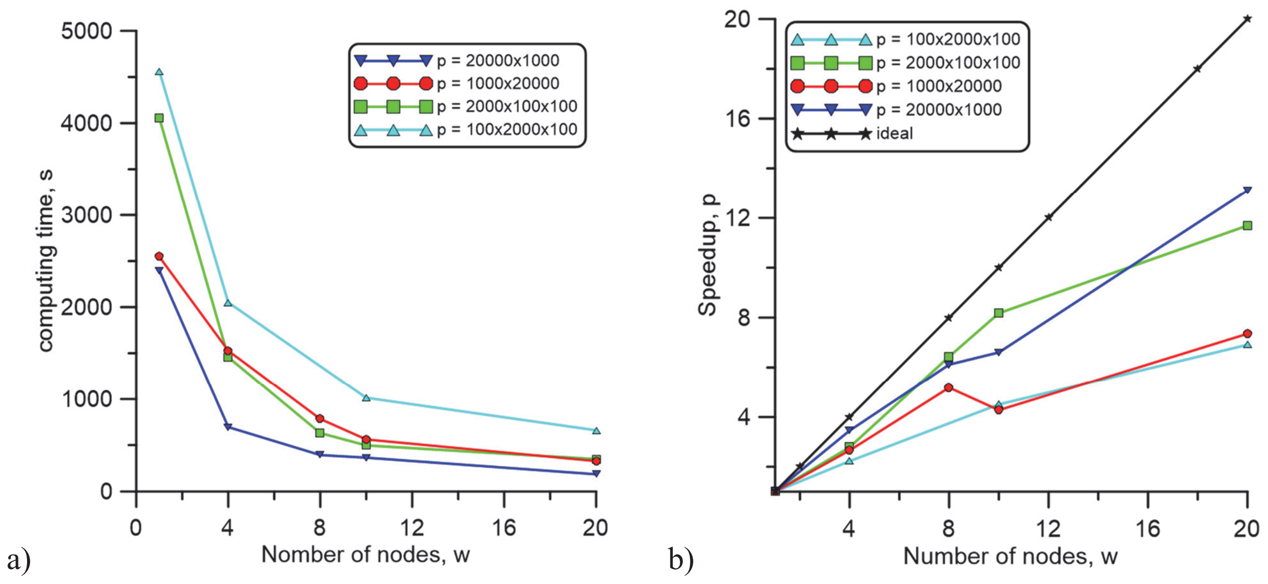


Fig. 13. *Results from simulations with 20 million MC cells and MPI standard a) simulation time, b) corresponding computation speedup.*

4. CONCLUSIONS

Based on the presented investigation it can be concluded that:

- Parallelization of the code based on MPI and OpenMP standards provides decrease in computing time.
- When the number of computing nodes increases, the computing time decreases to some extent. For OpenMP authors obtained speedup around 2, on the other hand MPI allows to speedup computation up to 12 times.
- For MPI computing time reduction for more than 10-12 nodes is insignificant. Such behavior is not beneficial from practical point of view. However, running several different calculations at the same time for various processing conditions may be a solution to this issue.
- With increasing number of nodes the speedup decreases. This is related to the increasing number of messages passing between nodes.

ACKNOWLEDGEMENT

Financial assistance of the POIG PLGird+ project is acknowledged. Financial assistance of the POIG.02.03.00-00-096/10 as part of the PLGrid Plus project is acknowledged.

REFERENCES

Dong, D., Chen, F., Cui, Z., 2016, Static recrystallization behavior of SA508-III steel during hot deformation, *Journal of Iron and Steel Research*, 23, 466-474.

Goins, P.E., Holm, E.A., 2016, The material point Monte Carlo model: a discrete, off-lattice method for microstructural evolution simulations, *Computational Materials Science*, 124, 411-419.

Humphreys, M.J., Hatherly, M., 2004, Recrystallization and related annealing phenomena, second ed. Elsevier, Oxford.

Ivasishin, O.M., Shevchenko, S.V., Vasiliev, N.L., Semiatin, S.L., 2006, A 3-D Monte Carlo (Potts) model for recrystallization and grain growth in polycrystalline materials, *Materials Science and Engineering A*, 422, 216-232.

Madej, L., Rauch, L., Perzynski, K., Cybulka, P., 2011, Digital Material Representation as an efficient tool for strain inhomogeneities analysis at the micro scale level, *Archives of Civil and Mechanical Engineering*, 11, 661-679.

Mason, J.K., 2015, Grain boundary energy and curvature in Monte Carlo and cellular automata simulations of grain boundary motion, *Acta Materialia*, 94, 162-171.

Mason, J.K., Lind, J., Lia, S.F., Reed, B.W., Kumar, M., 2015, Kinetics and anisotropy of the Monte Carlo model of grain growth, *Acta Materialia*, 82, 155-166.

Rollett, A.D., Manohar, P., 2004, The Monte Carlo Method, in Continuum Scale Simulation of Engineering Materials: Fundamentals - Microstructures - Process Applications, eds Raabe, D., Roters, F., Barlat, F., Chen, L.-Q., Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, FRG. doi: 10.1002/3527603786.ch4, 77-114.

Scholtes, B., Boulais-Sinou, R., Settefrati, A., D. Muñoz, D.P., Poitrault, I., Montouchet, A., Bozzol, N., Bernacki, M., 2016, 3D level set modeling of static recrystallization considering stored energy fields, *Computational Materials Science*, 122, 57-71.

Sieradzki, L., Madej, L., 2013, A perceptive comparison of the cellular automata and Monte Carlo techniques in application to static recrystallization modeling in polycrystalline materials, *Computational Material Science*, 67, 156-173.

Su, J., Sanjari, M., Kabir, A.S.H., Jonas, J.J., Yue, S., 2016, Static recrystallization behaviour of magnesium AZ31 alloy subjected to high speed rolling, *Materials Science & Engineering A*, 662, 412-425.

Sun, L., Muszka, K., Wynne, B.P., Palmiere, E.J., 2013, On the interactions between strain path reversal and dynamic recrystallisation in 316L stainless steel studied by hot torsion, *Materials Science and Engineering A*, 568, 160-170.

Williamson, A., Delplanque, J.-P., 2016, Investigation of dynamic abnormal grain growth using the Monte Carlo Potts method, *Computational Materials Science*, 124, 114-129.

EFEKTYWNY PODZIAŁ PRZESTRZENI OBLICZENIOWEJ W RÓWNOLEGŁEJ WERSJI ALGORYTMU ROZROSTU ZIAREN NA BAZIE METODY MONTE CARLO

Streszczenie

W pracy przedstawiono implementację równoległej wersji algorytmu rozrostu ziaren z wykorzystaniem metody Monte Carlo (MC). W pierwszej części pracy zostały przedstawione modyfikacje klasycznego algorytmu rozrostu ziaren bazującego na metodzie MC, pozwalające na równoległe wykonanie aplikacji. Następnie zostały opisane różne podziały przestrzeni obliczeniowej pomiędzy poszczególne subdomeny obliczeniowe. Wyniki przedstawionej implementacji opartej na OpenMP oraz MPI zostały zaprezentowane oraz porównane pod kontem przyspieszenia obliczeń oraz maksymalnej redukcji czasu wykonania symulacji.

Received: October 14, 2016
Received in a revised form: November 27, 2016
Accepted: December 16, 2016