**TECHNICAL TRANSACTIONS** | **CZASOPISMO TECHNICZNE**
MECHANICS | MECHANIKA

1-M/2013

TOMASZ DĘBIŃSKI*, MIROSŁAW GŁOWACKI*

# INCREASING EFFICIENCY COMPUTING OF SIMULATION ROLLING PROCESS WITH SEMI-SOLID ZONE, TESTS OF PARALLEL COMPUTING

# ZWIĘKSZENIE EFEKTYWNOŚCI OBLICZEŃ PROGRAMU SYMULACJI KOMPUTEROWEJ PROCESU WALCOWANIA ZE STREFĄ PÓŁCIEKŁĄ – ZRÓWNOLEGLENIE OBLICZEŃ

A b s t r a c t

The paper deals with the speed up of a computer program, which simulates rolling of steel with semi-solid zone. The mathematical models describing the rolling process are fully three dimensional. It leads to very long computation time while the discretization of the problem is made in finite element manner. The main objectives of the contribution are optimization of the program code, as well as making of the first look at possibilities of parallel computation in application to the presented problem. The paper has proved that efficient optimization of the program code can lead to significant shortening of the computation time also with the automatic and manual parallelization. It was concluded that adaptation of the procedures and loops to rigorous requirements of the parallel compiler is strongly recommended.

*Keywords*: *rolling process*, *parallel computation*, *computer simulation*, *numerical models*

S t r e s z c z e n i e

W artykule przedstawiono metody zwiększenia efektywności programu modelującego proces walcowania stali ze strefą półciekłą. Modele matematyczne opisujące proces walcowania są w pełni trójwymiarowe, co prowadzi do bardzo długich czasów obliczeń. Dyskretyzację problemu wykonano przy użyciu metody elementów skończonych. Głównymi celami artykułu są: optymalizacja kodu programu oraz ocena możliwości zastosowania obliczeń równoległych. Przeprowadzone testy udowodniły, że optymalizacja kodu programu może doprowadzić do znacznego skrócenia czasu obliczeń, również w połączeniu z automatycznymi ręcznym zrównolegleniem. W celu efektywnego zrównoleglenia automatycznego, konieczne jest dostosowanie procedur i pętli do rygorystycznych wymogów kompilatora.

*Słowa kluczowe*: *walcowanie*, *strefa półciekła*, *symulacja komputerowa*, *efektywność obliczeń*, *obliczenia równoległe*

* DSc. Eng. Tomasz Dębiński, DSc. Eng. Mirosław Głowacki, prof. AGH, Department of Applied Computer Science and Modelling, Faculty of Metals Engineering and Industrial Computer Science, AGH University of Science and Technology, ZI-IF UJK Kielce.

## 1. Introduction

Optimization of programs was and still is very important. Despite the increasing power of computers in accordance with Moore's Law, the optimization can not be ignored. Today, it is used in most high-level languages to facilitate programming but also reduces control over the execution of processor instructions. The compilation gives different result codes dependingon the compiler and platform. For assembler, it is assumed that the execution of each instruction corresponds to one clock cycle of the processor. C and Fortran languages are higher level. One instruction no longer corresponds to one assembly instruction. Each instruction corresponds to a number of assembler instructions. Each instruction has its counterpart in the five to eight assembler instructions [1].

Software performance depends not only on the compiler but also, on coding, on decomposition of the problem into subtasks, on used structures and libraries and on system architecture. Software performance has several meanings. The most frequently used are memory and time efficiency. Memory efficiency is a way to find an algorithm that allows to solve the problem with minimal memory usage.It cannot be ignored, because in today's systems, there is a caching and paging problem that can significantly affect the speed of the program. Time performance is responsible for numerical solution of a problem in shorter time [2, 3]

The main goals of the current paper is to determine the optimal hardware platform for computer simulation of the rolling process with semi-solid zone and code optimization for best time performance and attempt of parallelization of sequential program.

## 2. Sequential model of rolling – thermal solution

Heat transfer is one of the main phenomena accompanying the hot rolling process, which results in formation of temperature gradients inside the deformation zone and even outside of it. In the presented model, heat flow is considered in a particular area of the specimen. The discretization process leads to selection a finite number of points inside the body. Certain temperature is attributed to each of the points (nodes). The set of all the temperature values at all given points creates a space and time dependent temperature field $T = f(x, y, z, t)$. Heat transfer models are usualy based on the solution of Fourier-Kirchhoff heat conduction equation [4, 5]. The solution is based on minimization of heat flux functional, which includes relevant boundary conditions.

$$\chi = \int_V \left\{ \frac{1}{2}\left[ k_x\left(\frac{\partial T}{\partial x}\right)^2 + k_y\left(\frac{\partial T}{\partial y}\right)^2 + k_z\left(\frac{\partial T}{\partial z}\right)^2 \right] - QT \right\} \cdot dV + \int_S \left( qT + \frac{1}{2}\alpha (T - T_0)^2 \right) dS \quad (1)$$

In (1) $\lambda_i$ are anisotropic heat transfer coefficients, $Q$ – heat generation, $V$ – control volume, $S$ – body surface, $\alpha$ – heat transfer coefficient and $q$ – heat generated by friction. In [6] one can find solution of this problem in two steps. The first one is based on the Fourier-Kirchhoff equation for steady flow of heat using the finite element method. The second step is a generalization of the resulting stationary matrix equations obtained for the steady-state process with the help of the Galerkin residual method. In both cases the solution requires solution of anequations system represented in matrix form (2):

$$\mathbf{KT} = \mathbf{p} \tag{2}$$

where:

- $\mathbf{K}$ – heat capacity matrix,
- $\mathbf{T}$ – parameters vector,
- $\mathbf{p}$ – the right hand vector.

## 3. Mechanical Model

Another model, the most important one for metal forming processes, is the material plastic behaviour model. In the presented approach a three dimensional rigid-plastic model of rolling process has been applied, which in the case of large plastic deformations at very high temperatures can give good results. The power functional resulting from variational formulation of the problem is non-linear in all cases and so the solution requires strong computing power. Application of finite element discretization to analysis of spatial rigid-plastic model involves processing a large number of variational parameters. Hence, the calculation requires long computation time but results in a solution which is consistent with experimental data for both simple and complex deformation zones.

The variational approach of the rolling process requires optimization of a power functional, which in general can be expressed in the form of equation (3):

$$W = W_\sigma + W_\lambda + W_t \tag{3}$$

where:

- $W_\sigma$ – the power of plastic deformation,
- $W_\lambda$ – the incompressibility condition (penalty power),
- $W_t$ – the friction power.

The deformation process of steel in semi-solid state depends on material density changes. In this case the condition of incompressibility, which is sufficient for deformation at lower temperatures, has to be replaced with a more general condition of mass conservation [7].

The application of finite element method results in case of metal forming processes in a finite set of velocity values. All of them are parameters of the deformation field. Spatial discretization allows the optimization of deformation field for a certain time step. At the same time discretization of time coordinate is necessary to perform series of time steps under the assumption of constant strain field in each step. The incremental solution of the problem has to be constructed iteratively. Each iteration involves solving linearized system which can be expressed as a matrix equation (4)

$$\mathbf{K}\boldsymbol{v} = \mathbf{f} \tag{4}$$

where:

- $\mathbf{K}$ – the structure stiffness matrix,
- $\boldsymbol{v}$ – the nodal velocity vector,
- $\mathbf{f}$ – the right hand vector.

The process is divided into time steps. In each step ($\Delta t$) the final shape of the body is calculated from its initial shape using the velocity field resulting from optimization of the functional given by (3)

## 4. Code optimalization

Programs that simulate complex physical processes, using the finite element method, are always very complex numerically. The code which undergoes the penalization consists of approximately 19,000 lines. About 2000 of themlay in the critical section, which is part of the algorithm,where are located the most time-consuming calculations. For complex algorithms, where the computational time is very long, optimization is very important. May consist of the following steps: control flow analysis, data flow analysis (how to use variables in the program), optimizing transformations, improvement of the source code level (profiling, changing the algorithm), improvement of the level of indirect representation (improvement loop, calculating addresses ) and finally improvements at the level of the object code (use of registers, the choice of proper instructions).

The first level of optimization involves the use of compiler options for full utilization of hardware and the second is to optimize the program code in aim to take full advantage of the speed of the algorithm. For performance tests a unit equipped with 64 Intel (R) Xeon (R) CPU E7-4830@2.13GHz, and 128GB of memory was used. Computer has the ability to work in both 64-bit and 32-bit emulation, which allows to work under any operating system compatible with the x86 architecture.

For maximum performance, compilation and calculations was made on the Linux Fedora 13 system with KVM virtual machine. Calculations were made using both32-bit and 64-bit representation for verify the increase of performance and compiler quality.

Optimization for the compiler generally allows to choice of processor and the global optimization. Global optimization includes all the development techniques such as unroll of loops, conditional statements, change of the code, queuing instructions, etc. This technique is insufficient, it depends on the code and algorithm.

The study was limited to a standard optimization options for easy comparison of their effectiveness. Three levels were used (options O1, O2, O3) containing different levels of optimization algorithms. Detailed description of a compiler one can find in the help files of the operating system [8]. The program was compiled with different optimization levels for two hardware architectures 32 and 64 bit.

In both cases, g77 compilers were used because the source code waswritten in Fortran77. In order to minimize the error, in each case the calculation was performed three times, because the computing node is located on a server cluster whichcan be loaded by other users. The results are shown in Figure 1.

The presented results show the advantage of 64-bit code. The 32-bit code was executed slower, on average about 20%.

The second stage of the work was to optimize the source code. After analysis (profiling) the 184 procedures have been found in the whole code. A major influence on the program execution time has 4 procedures, which are presented in Table 1.

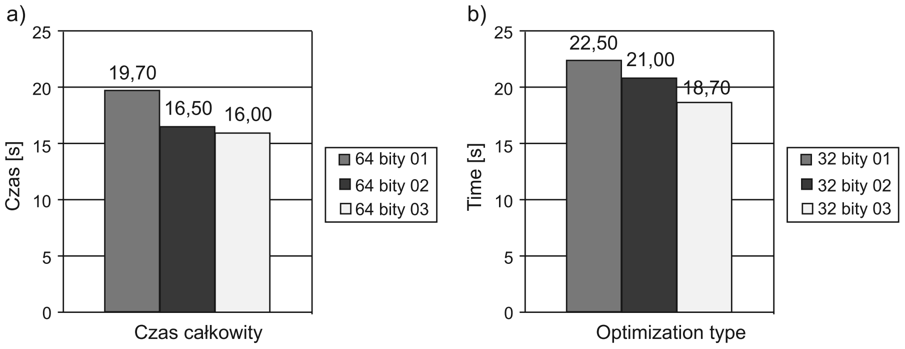

Fig. 1. Juxtaposition of computing times for 64- and 32 bit version of the program

Rys. 1. Zestawienie czasów obliczeń dla 64 bitowej (a) i 32 bitowej (b) wersji programu

Table 1

**Execution time and number of calls for selected procedures**

| % execution time | number of calls | procedure name |
|---|---|---|
| 37.16 | 4407696 | b_3d |
| 16.88 | 1098684 | poch |
| 13.18 | 492 | mp11ad |
| 5.66 | 2888 | mp11ed |
| Rest | – | other |

The main aim of the task is to improve performance of these procedures, which reduce the execution time for the entire program. The main applied techniques allowing the task are: the elimination of common subexpressions, use algebra rights, copy propagation, removing

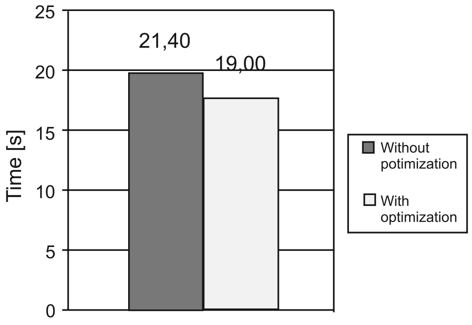

Fig. 2. Computing time before and after optimization

Rys. 2. Czas obliczeniowy przed i po optymalizacji unused code, code movement and induction variables. After analysis of the execution time and the number of calls four procedures were selected for optimization. The calculation shows improvement of the execution time byabout 5% of the overall computation time as shown in Figure 2

It should be noticed that the calculations were performed with low level of the functional optimization.

## 5. Parallel computing

The most easy way leading to a parallel version of the program is to use compiler directives. Intel Fortran Compiler v 8.1 uses automatic parallelization mechanism and allows users to change the program code with the direct placement of appropriate directives in the program. The automatic parallelization requires meeting several conditions. The operation causes automatic parallelization of loops and divides code between available processors. If the considered loops contain elements of arrays then the parallel execution of instructions is possible only in case of independency of the result on the order of execution. In order to solve the task in parallel, compiler creates n threads performing the task for its own areas of control variable.

Parallel version of the loop must satisfy the following conditions: (i) the calculations in the loop does not change the same variables in the simple iteration, (ii) loop calculations do not change the variable that is used outside of the loop, (iii) the value of anarray element in the current iteration does not depend on the values of the elements from other iterations. Parallelism is not possible in the following cases: (i) DO loop is nested in another loop that is parallel, (ii) jump instructions are beyond of the scope of DO, (iii) a loop contains a call of user subroutine, (iv) the loop contains instructions for I / O and (v) the in the iteration scalar is calculated .

The program has been subjected toautomatic parallelization. The compiler has identified 22 loops which meet the above criteria. There was no increase in productivity, the difference in computation time for the calculation on a single processor can be omitted. No improvement in performance is caused by the fact that parallelization of loops which meet the criteria is not very significant, and loops are located outside the critical section. In order to meet the requirements of the compiler the need of modification of the code is required. A very important factor is the number of processors. Further studies on a larger number of processors are planned.

## 6. Conclusions

At this stage of the work, it is necessary to use specialized tools for program analysis (profiling) and the elimination or improvement of the code causing the greatest delays. The use of even the most basic code optimization techniques and appropriate compiler options gives good results and allows the shortening of the execution time of the program. For automatic parallelization of calculations it is necessary to adapt the procedures and loops to requirementsof the compiler, which is the subject of further work. Because the problem is complex, a detailed analysis of the program performance is planned.

Optimization of programs implementing complex calculations, particularly working on a large amount of data is very important. Shortening the calculation time by a few percent in the context of the overall problem gives significant improvement.Performance analysis is necessary, in order to improve the time characteristics of the task.

*The work has been supported by the Polish Ministry of Science and Higher Education Grant N N508 585539.*

## References

[1] Bulka B., Mayhew D., *Efektywne programowanie w C++*, Wydawnictwo MIKOM, Warszawa, kwiecień 2001.
[2] Gian-Paolo Musumeci D., Loukides M., *Optymalizacja systemów komputerowych*, RM 2002.
[3] Christos H. Papadimitriou, *Złożoność obliczeniowa*, Helion, 2012.
[4] Taler J., Duda P., *Direct and inverse heat transfer problems*, WNT, Warszawa 2003.
[5] Kostkowski E., *Heat flow*, Silesian Technical University, Gliwice 2000.
[6] Głowacki M., *Thermo-mechanical and microstructural model of shape rolling*, Dissertations and Monographs, Kraków 1998.
[7] Głowacki M., *Computer simulation of rolling of semi-solid steel slabs*, Proc. 15th International symposium on Plasticity & its current applications, St. Thomas, U. S. Virgin Islands, January (2009), 10–12.
[8] SUSE Limux Enterprise (http://www.suse.com/us).