A R C H I V E S O F M E T A L L U R G Y A N D M A T E R I A L S
Volume 60 2015 Issue 1
DOI: 10.1515/amm-2015-0037

K. KONOPKA*, K. MIŁKOWSKA-PISZCZEK*, L. TRĘBACZ*, J. FALKUS*

IMPROVING EFFICIENCY OF CCS NUMERICAL SIMULATIONS THROUGH USE OF PARALLEL PROCESSING

POPRAWA WYDAJNOŚCI OBLICZEŃ NUMERYCZNYCH SYMULACJI COS POPRZEZ WYKORZYSTANIE PRZETWARZANIA RÓWNOLEGŁEGO

The study presents the findings of research concerning the possibilities for application of parallel processing in order to reduce the computing time of numerical simulations of the steel continuous casting process. The computing efficiency for a CCS model covering the mould and a strand fragment was analysed. The calculations were performed with the ProCAST software package using the finite element method. Two computing environments were used: the PL-Grid infrastructure and cloud computing platform.

Keywords: continuous casting of steel, efficiency of numerical calculations, numerical modelling, ProCAST

W pracy przedstawiono wyniki badań dotyczących możliwości zastosowania przetwarzania równoległego w celu skrócenia czasu trwania obliczeń numerycznych symulacji procesu ciągłego odlewania stali. Przeanalizowano wydajność obliczeń dla modelu COS obejmującego krystalizator i fragment pasma. Obliczenia przeprowadzono przy pomocy pakietu oprogramowania ProCAST w oparciu o metodę elementów skończonych. Wykorzystano dwa środowiska obliczeniowe: infrastrukturę PL-Grid i chmurę obliczeniową.

1. Introduction

Numerical modelling of production processes allows an improvement in quality of the final product, as well as an increase in efficiency and a reduction of the process failure frequency. For the continuous steel casting process the modelling of the temperature field is key. Using numerical models in industrial practice is time-consuming and often requires that a number of simulations are carried out with different sets of boundary conditions, initial conditions and material parameters – taking up valuable time [1].

Thanks to progress in computer science, in particular in the design of more and more efficient processors multi-core computing units are commonly applied, allowing simultaneous operation of several programs. This paper presents an analysis of the possibilities of using of multiple processor cores for conducting numerical simulations of the continuous steel casting process. Shortening computing time is particularly important for verification of the model formulated, also at the stage of the production process optimisation using the created numerical models.

2. Optimisation of the computing time of numerical simulations

Three basic approaches allowing us to shorten the computing time of numerical simulations using the finite element method can be distinguish; these are:

– mathematical model optimisation
– solver source code optimisation
– parallel and distributed processing.

Each of these methods has its advantages and disadvantages and can be potentially applied at a different stage of the problem solving process.

2.1. Mathematical model optimisation

A mathematical model can be optimised by selecting the type and size of finite elements used, so that the obtained solution has the maximum accuracy and at the same time as few elements are used for the construction of the model as possible. Minimisation of the finite element number translates into a less complex system of equations that are solved by the solver in each iteration. One of the methods applied for numerical simulations of processes with intensive heat transfer is finite element mesh refinement in a place where a high gradient of temperature is expected [2, 3]. The mathematical model must be optimised already at the stage of its construction. This optimisation method has a drawback, which is its direct influence on the accuracy of the solution obtained – in particular the application of finite elements which are too large often results in an increase in numerical errors [4]. The next drawback of such a solution is the need to carry out verification after each implemented modification of a mathematical model; this is time-consuming, particularly in the case of complicated numerical simulations.

* AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY, FACULTY OF METALS ENGINEERING AND INDUSTRIAL COMPUTER SCIENCE, AL. A. MICKIEWICZA 30, 30-059 KRAKÓW, POLAND

2.2. Solver source code optimisation

The second of the above-mentioned approaches is only possible for those software packages in which the source code is available. It concerns particularly dedicated software and packages made available under a number of licences from the Open Source group. The following may be presented as examples of methods of solver source code optimisation applied in practice - the use of a set of extensions of processor instructions accelerating operations on matrices and vectors, or the use of FORTRAN programming language, which was designed particularly taking into account the efficiency of the compiled code. The necessity of accessing the solver source code should be considered a disadvantage of this method. We need to emphasise that for this optimisation method the enhancement of solver efficiency largely depends on the quality of solutions implemented by the user.

2.3. Parallel processing

Parallel processing means executing more than one instruction within a specific time unit. In practice, parallel processing is processing that uses more than one processor (or a processor core for multi-core processors) simultaneously for performing numerical computing. The applied processing units (cores) may be located within the same computer system – Symmetric Multi Processing (SMP) or within different computer systems – Distributed Memory Processing (DMP). To enable more than one processor core to be used during computing the solver must make such functionality available. In particular an algorithm for breaking down a task into smaller sub-tasks is necessary. Each of the sub-tasks may be assigned to a different processor (processor core), whereas for distributed computing,a communication method between individual processes run on various computer systems is needed. The commercially available computer applications for the modelling of crystallisation processes are mostly based on the Finite Element Method [5, 6], the FEM can also be used for modelling microstructure and heat treatment process [7]. There are a few systems in the market for modelling the continuous casting of steel:

– ProCAST (ESI) for Windows and Linux OS [8],
– THERCAST (Transvalor) for Windows and Linux OS [9,10,],
– FLOW-3D Cast (Flow Science Inc.) for Windows OS [11],
– CC Master (Expresslab) [12].

Only the first two packages enable calculations to be performed on a number of processors/machines.

3. Construction of a CCS numerical model

A 3D model of a mould with a height of 900 mm and a wall thickness of 40 mm, was designed with SolidWorks software. Filling the mould with liquid steel was assumed at a level of 850 mm. A strand with an arc radius of 10.5 m, and dimensions of 220×1100 mm, and a length of ca. 0.5 m was designed.

A mesh of 3D finite elements was applied onto the mould and the strand. The mesh contains tetrahedral elements with an average size of 10 mm. In addition, to properly reflect the heat transfer process in the mould – mostly the shell formation process – two layers of elements with an average size of 5 mm were assumed in the area of contact of the strand with the mould wall and along the whole length of the strand. The 3D element mesh for this model has about 240,000 tetrahedral elements of various sizes. For individual strand fragments the finite element size in the mesh varies from 5 mm to 20 mm. The most dense mesh is located in the area of the mould and directly under the mould. The model of mould and strand is presented in Fig. 1.

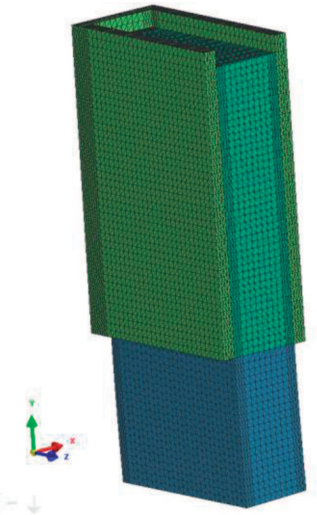

Fig. 1. FEM model

3.1. Boundary and initial conditions

A description the heat transfer model in the continuous steel casting process is a complex task as all three mechanisms of heat transfer occur within this process. The temperature field may be calculated with a numerical model of the steel continuous casting process by solving the generalised Fourier equation. The solution of the Fourier equation should meet those boundary conditions declared on the cast strand surface [13,14]. The surfaces for which boundary conditions were introduced were broken down into four groups:

1. Contact of the solidifying strand surface with the inner side of the mould
2. The outer side of the mould
3. The surface of the liquid steel meniscus
4. The secondary cooling zone

In the formulated model, heat transfer between the cast strand and the mould surfaces is performed through the gap. The heat transfer area in the mould is divided into two zones. In the first zone no presence of a gaseous gap is assumed; the whole space between the strand and the mould is filled with the mould powder. As a result of metal contraction, a gaseous gap develops at a certain distance from the meniscus. This gap separates the mould powder layer from the mould surface, additionally insulating the strand. In this study, the maximum value of the heat transfer coefficient of 1600W/m^{-2}K was assumed. A change in the value of the heat transfer coefficient is related to the assumed model of air gap formation.

The outer side of the mould is cooled with water flowing through channels. Heat is transferred by forced convection. Due to the way heat is absorbed by water flowing through the channel, it is hard to determine the heat transfer coefficient for the mould channels according to the available equations. To find the Nusselt number, the equation (1) was chosen

$$Nu = 0.021 Re^{0.8} \underset{w}{\overset{0.43}{\mathrm{Pr}}} \left(\frac{\mathrm{Pr}_W}{\mathrm{Pr}_K}\right)^{0.25} \quad (1)$$

where: index w – for the mean water temperature in a channel; index k – for the strand surface temperature

In order to compute the mean heat transfer coefficient, the following equation (2) was used for the outer side of the mold:

$$\alpha_W = \frac{Nu\lambda_w}{d_k} x_k \tag{2}$$

where Nu is the Nusselt number; λ_w is the thermal conductivity for the cooling water / W/mK; x_k is the share of water-cooled mould area; d_k is the cooling channel diameter / mm. The heat transfer coefficient was calculated at 24 000 W/m^{2}K.

After leaving the mould, the strand surface is cooled with a water spray and in the air. The heat flux carried away from the surface of the cooling cast strand is proportional to the temperature difference of the strand surface and the cooling medium. If the solidifying cast strand surface temperature is unknown, a simplified formula (3) may be applied, allowing determinationof the heat transfer coefficient:

$$\alpha_{spray} = 10v + (107 + 0,688v)w \tag{3}$$

where v is the water drop velocity /m/s; w is the water flux density / dm^3/m^{2}s. For the secondary cooling zone, based on the numerical values of water flux density, a set of heat transfer coefficients was calculated for each of the spray zones.

4. Computing variants and computing environments

In order to determine the possibilities for using parallel processing to compute the temperature distribution in the whole volume of the cast strand, a number of simulations were performed using the prepared model. The number of simulation iterations was set at 4000, which allowed all tested variants to reach a steady state. At the same time it enabled an acceptable computing time to be maintained; this was key due to the necessity of repeating the measurement several times in order to eliminate the impact of external factors, such as the loading of computing resources with other tasks executed by e.g. the operating system. The calculation efficiency was verified for two environments: the PL-Grid grid environment and the cloud computing based on the XenServer 6.2 built for the needs of the research. For the grid platform, measurements were made for 1, 2, 4 and 8 processor cores. For the cloud computing, measurements were made for 1, 2 and 4 cores; measurements for 8 cores were not possible because of the hardware configuration of the workstations comprising the cloud. In both cases the 64-bit solver version was used.

4.1. PL-Grid environment

The first of the tested computing environments was the PL-Grid platform. The system consists of computing clusters located in a number of research units in Poland. Each of the units make a few hundred computing nodes with diverse hardware configurations available for the needs of the PL-Grid platform users. The users may conduct computing by using software packages installed on the platform, dedicated, area-oriented computing services and by running dedicated software packages. An example of an area-oriented computing service is a solution implemented by the authors which also uses the ProCast package solver and allows running simulations of the steel continuous casting process. As part of the PL-Grid platform it is also possible to use specialised systems, allowing one to conduct General-Purpose Computing on Graphic Processor Units (GPGPU) and using the vSMP technology that allows hardware resources of a few servers to be aggregated. This solution is particularly useful for running applications with especially high requirements in terms of operating memory and processor cores [15]. During the research the solver of the ProCAST 2013.5 package was used in the Linux version. The input files were prepared with a preprocessor in the Microsoft Windows version.

4.2. Cloud computing

The other one of the tested solutions was a computer cloud, configured especially for the needs of the research. Solutions based on cloud computing use the virtualization technique, which allows the creation of many virtual computers – called virtual machines – within a single, physical computer. Physical computers comprising a cloud are called nodes, while one of the basic features is the possibility of smooth migration (transfer) of virtual machines between the individual cloud nodes without the need to interrupt their operation. The implemented cloud comprised four computers, one of them was the pool master, the next one provided disc space for the data storage, and the other two performed the role of computing nodes, whereon the computing was performed. The communication between the cloud computers was provided by a 1Gb/s Ethernet network. The XenServer 6.2 software was used for construction of the cloud (the pool master and the computing nodes) and Linux Debian 7.2 for the data storage node. The selection of software was dictated by its availability under the OpenSource license. The computing was performed within a single virtual machine, wherein the Microsoft Windows Server 2008 R2 system and the ProCAST 2013.5 package in the version dedicated for this system were installed. The simulation was prepared beforehand with a ProCAST package preprocessor and then copied to the virtual machine hard drive.

5. Results

Averaged computation times for each of the variants are presented below. The results for the PL-Grid platform and the Linux system solver are presented in Table 1. Computation times for the solution using the cloud and the Microsoft Windows system solver are shown in Table 2. The results in the form of graphs are presented in Fig. 2 and 3 respectively. It should be emphasised, that because of the different hardware configuration of machines comprising both platforms and two different operating systems installed, the results should not be compared directly with each other.

TABLE 1

PL-Grid infrastructure – average computation time

	1 CPU core	2 CPU cores	4 CPU cores	8 CPU cores
Computation time [s]	10936	6400	3426	2101

TABLE 2

Cloud computing – average computation time

	1 CPU core	2 CPU cores	4 CPU cores
Computation time [s]	4367	2371	1541

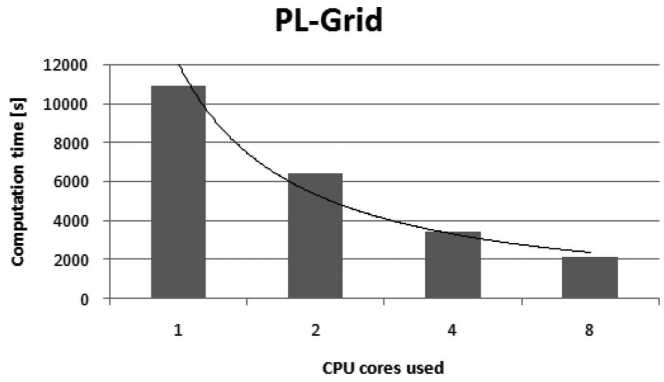

Fig. 2. Average computation time, PL-Grid infrastructure

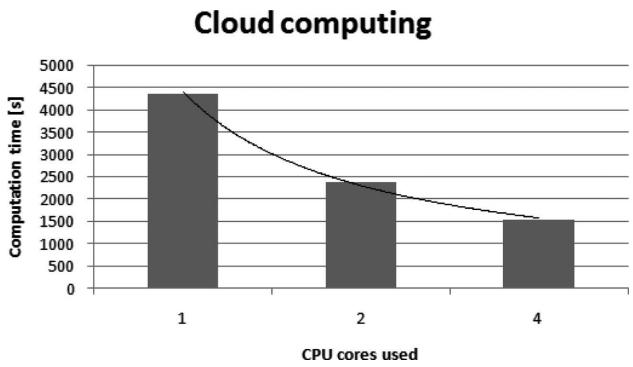

Fig. 3. Average computation time, cloud computing

6. Summary and conclusions

As part of the research, a mathematical model covering the mould and a fragment of the strand cast was created and next it was used to carry out efficiency tests with various numbers of processor cores. It follows from the analysis of data in Tables 1 and 2 that using many processor cores for carrying out numerical calculations can shorten the computing time. The improvement of efficiency was observed both for the grid platform using the Linux system and the dedicated version of the ProCast package solver, as well as for the solution that used virtualisation and a Microsoft operating system with an appropriate solver version. In both cases an increase in the number of cores used lead to a similar reduction of computing time in terms of percentage, allowing us to state that the tested platforms are equally suitable for conducting parallel computing. The non-linear nature of the trend line on the graphs presented in Fig. 2 and 3 indicates that the use of more and more processors for computing results in decreasing efficiency. This is also caused by the growing communication overheads arising from the necessity of information exchange between the individual solver processes. For the created mathematical model the use of four processor cores is a good compromise between the obtained platform efficiency and the financial outlays necessary to build this platform.

Received: 20 March 2014.

Acknowledgements

This research was supported in part by PL-Grid Infrastructure.
This research was supported through statutory funds of the AGH UST project no. 11.11.110.293.

REFERENCES

[1] M. Rywotycki, K. Miłkowska-Piszczek, L. Trebacz, Identification of the boundary conditions in the continuous casting of steel, Archives of Metallurgy and Materials **57**, 385-393 (2012).
[2] A. Buczek, A.Burbelko, P. Drożdż, M. Dziarmagowski, J. Falkus, M.Karbowniczek, Tomasz Kargul, K. Miłkowska-Piszczek, M. Rywotycki, K. Sołek, W. Ślęzak, T. Telejko, L. Trębacz, E. Wielgosz, Modelowanie procesu ciągłego odlewania stali – monografia, Radom 2012.
[3] K. Miłkowska-Piszczek, PhDThesis: Opracowanie i zastosowanie numerycznego modelu procesu COS do wyznaczenia technologicznych parametrów odlewania stali S235, AGH University of Science and Technology, Kraków 2013.
[4] W. Rachowicz, Metoda elementów skończonych i brzegowych. Podstawy kontroli błędu i adaptacji, Politechnika Krakowska, Kraków 2012.
[5] O.C. Zienkiewicz, R.L Taylor, J.Z. Zhu, The Finite Element Method: Its Basis and Fundamentals. 6th Ed. Oxford: Butterworth-Heinemann, 2005.
[6] T.R. Hsu, The Finite Element Method in Thermomechanics, Allen and Unwin, London 1986.
[7] L. Madej, H. Paul, L. Trebacz, W. Wajda, M. Pietrzyk, Multi billet extrusion technology for manufacturing bi-layered components, CIRP Annals – Manufacturing Technology **61**, 1, 235-238 (2012).
[8] Procast software, [on-line], https://www.esi-group.com/software-services, (21.05.2014).
[9] THERCAST software, [on-line], http://www.transvalor.com/, (21.05.2014).
[10] K. Vollrath, Casting simulation using numerical processing becomes more important in steel mills, Stahl und Eisen **133**, 45-53 (2013).
[11] X.J. Liu, S.H. Bhavnani, R.A. Overfelt, Simulation of EPS foam decomposition in the lost foam casting process. J. Mater. Process. Technol. **182**, 333-342 (2007).
[12] CC Master, [on-line], http://www.expresslab.co.kr/home/products/?view=CC-Master, (21.05.2014).
[13] K. Miłkowska-Piszczek, J. Falkus, Calculation of the boundary conditions in the continuous casting of steel process, Metalurgija **53**, 4, 571-573 (2014).
[14] K. Miłkowska-Piszczek, M. Korolczuk-Hejnak, An analysis of the influence of viscosity on the numerical simulation of temperature distribution, as demonstrated by the CC process, Archives of Metallurgy and Materials **58**, 4, 1267-1274 (2013).
[15] PL-Grid Oferta dla użytkowników, [on-line], http://www.plgrid.pl/oferta/, (21.05.2014).