

The Performance of Geothermal Field Modeling in Distributed Component Environment

A. Piórkowski, A. Pięta, A. Kowal, T. Danek
Department of Geoinformatics and Applied Computer Science,
AGH University of Science and Technology, Cracow, Poland.
{pioro, tdanek} @agh.edu.pl, apieta@geol.agh.edu.pl

Abstract—An implementation and performance analysis of heat transfer modeling using most popular component environments is a scope of this article. The computational problem is described, and the proposed solution of decomposition for parallelization is shown. The implementation is prepared for MS .NET, Sun Java and Mono. Tests are done for various operating systems and hardware platform combinations. The performance of calculations is experimentally indicated and analyzed. The most interesting issue is the communication tuning in distributed component software – proposed method can speed up computational time, but the final time depends also on the network connections performance in component environments. These results are presented and discussed.

Index Terms—heat transfer modeling, component platforms

I. INTRODUCTION

Heat transfer phenomenon plays an important role in such physical problems in Earth science as volcanoes, intrusions, earthquakes, mountain building or metamorphism. Analytically exact solution of heat transfer equation exists only for geological models with simple geometry. In many geologically realistic situations it is necessary to use numerical approaches to obtain valid models of these important processes. Heat-conduction equation and its finite difference solution is a problem which is easy to parallelize. This fact is a great advantage because modeling in huge complex media carry on for a long period of time and in presence of thermal anisotropy can be a very time consuming process [1], [2].

The main disadvantages of created program codes is introduction of newer and newer software and hardware solutions that caused in very short life cycle of created codes. One of the methods used to overcome the problem of short life cycle is use of the component technologies.

II. COMPONENT TECHNOLOGIES

Portability, security and independence of hardware are the main advantages of component-based software. A possibility of component code reuse enables coders cooperation. The most popular component platforms are Sun

Java [3] and MS .NET [4]. These platforms are developed by commercial providers. There is a free and open alternative for one of them (MS .NET) – a platform Mono [5]. The .NET applications can be run under control of Mono.

The component platforms use a managed code technique to enable hardware independency and portability. This technique is a *bytecode* in Sun Java and *CLI* in MS .NET. The translation to native code by Just-In-Time method is a bottleneck in terms of performance. There are numerous algorithms for optimization to speed up calculations.

The subsystem of communication is one of the most important parts of component environment in terms of performance in case of parallelization. Sun Java offers a RMI mechanism [6] that enables remote procedure calling, the same function is domain of .NET Remoting for MS .NET and Mono.

III. TEST ENVIRONMENT AND APPLICATIONS

A. Hardware specification

The experimental tests of heat transfer modeling computations were performed in two different environments.

The first environment was a notebook:

- 1 processor (Intel Pentium M 1,86 GHz),
- 1 GB RAM.

The second environment was a cluster of 30 PC computers:

- 1 processor with 2 cores (Hyper-Threading Technology Intel Pentium 4 2,8 GHz),
- 1 GB RAM,
- Gigabit Ethernet network adapter.

All nodes of cluster were joined by Gigabit Ethernet switch.

B. Used Operating Systems and Component Environments

The performance of component seismic computing was measured in following operating systems: Linux (Fedora Core 3, kernel: 2.6.12-1.1381 [smp]), MS Windows 2000 (SP4), MS Windows XP (SP2) and MS Windows Vista.

There are two applications for heat transfer modeling written in Java and C# language. To run Java code we use Sun

Java SE virtual machine, version 1.6.0. The C# code was tested under control of MS .NET Framework 2.0 and Mono 1.2.6.

IV. HEAT TRANSFER MODELING

A. Basics of heat transfer modeling

Component solutions were tested using heat-conductive equation. In two-dimensional isotropic medium this equation can be written as:

$$\frac{\partial T}{\partial t} = \frac{\lambda}{\rho \cdot c} \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial z^2} \right) + \frac{Q_w}{\rho \cdot c} \quad (1)$$

where $T(x, z)$ is temperature, $\lambda(x, z)$ is thermal conductivity, $c(x, z)$ is specific heat capacity, $\rho(x, z)$ is density, t is time and Q_w denotes the heat generated in the volume element of the medium during unit time (heat production rate).

Adopting the finite difference to approximate the above equation and using the Simple Explicit Method [7] one can obtain:

$$t_{i,j}^{k+1} = t_{i,j}^k + \alpha \Delta t \left[\frac{t_{i+1,j}^k - 2t_{i,j}^k + t_{i-1,j}^k}{(\Delta x)^2} + \frac{t_{i,j+1}^k - 2t_{i,j}^k + t_{i,j-1}^k}{(\Delta z)^2} \right] \quad (2)$$

$$i = 1, \dots, n_x; \quad j = 1, \dots, n_z$$

where α is thermal diffusivity, Δt is the time sampling interval, Δx and Δy are distances between grid points in the x and z directions respectively, n_x and n_y are grid points amount in the x and z directions respectively.

To achieve numerical stability the following relationship must be fulfilled [7].

$$\alpha \Delta t \left[\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta z)^2} \right] \leq \frac{1}{2} \quad (3)$$

We used two types of boundary conditions: convective boundary conditions (4) at the top and bottom of the model and Neumann boundary conditions (5) elsewhere.

$$\lambda \frac{\partial T}{\partial n_i} \Big|_{r_i} + h_i T(r_i, t) = f_i(r_i, t) \quad (4)$$

$$\lambda \frac{\partial T}{\partial n_i} \Big|_{r_i} = f_i(r_i, t) \quad (5)$$

where:

- r_i - coordinate at the boundary, i - one of the boundaries,
- n_i - the outward-facing normal vector on the body surface,
- h_i - the heat transfer coefficient,
- f_i - the specified function,
- λ - thermal conductivity.

B. An experimental model

There is a simple model prepared for simulation. The parameters of this model are presented in Table I and this model is presented on fig. 1. The model contains of a part of rock at the left and water at the right. The solutions of heat transfer simulation for 10 and 100 years are presented in the fig. 2 and fig. 3.

TABLE I.
PARAMETERS OF SIMPLE MODEL

Parameters of the experimental model	
model dimensions [m x m]	250 x 250
spatial grid steps [m]	1
depth to heat layer [m]	250
temperature of heat layer [°C]	100
thermal conductivity of land [W/m*K]	0,000001201
thermal conductivity of water [W/m*K]	0,000000128
end time [years]	100
time step [days]	1
initial temp. of land (250m) / water	100°C / 0°C

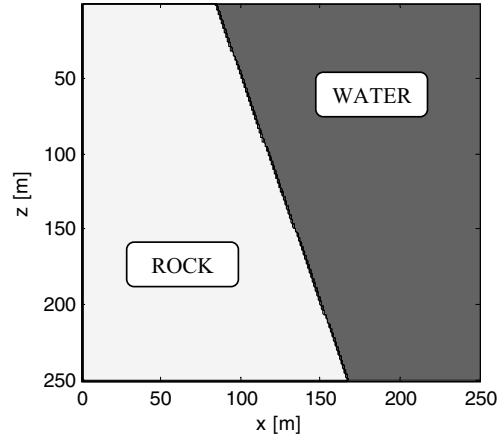


Fig 1. The simple model for simulation.

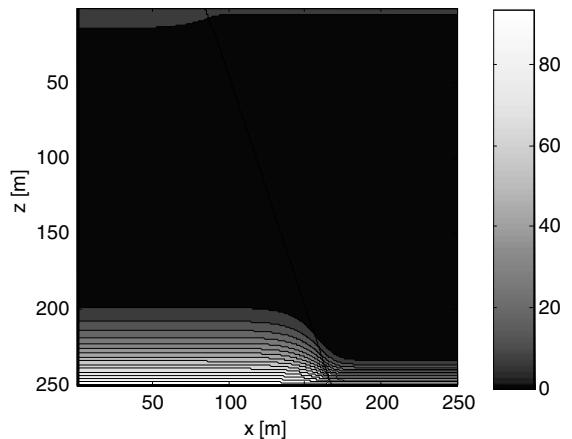


Fig 2. The solution of heat transfer simulation for 10 years.

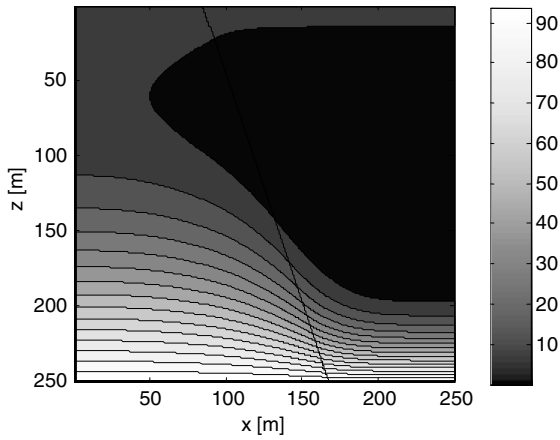


Fig 3. The solution of heat transfer simulation for 100 years.

C. Serial case

At the start we have measured the times of heat transfer modeling computations for the serial case. The experiment took a place in two hardware environments. The reached times are collected in Table II and presented in fig. 4.

TABLE II.
AVERAGE TIMES OF COMPUTATIONS FOR SINGLE MACHINE

	Avg. times in E1 [s]			Avg. times in E2 [s]	
	MS Vista	MS XP	FC 3	MS 2000	FC 3
.NET	99,22	95,73	-	142,43	-
Mono	110,86	107,04	107,93	231,93	181,84
Java	112,64	103,88	113,03	200,22	136,49

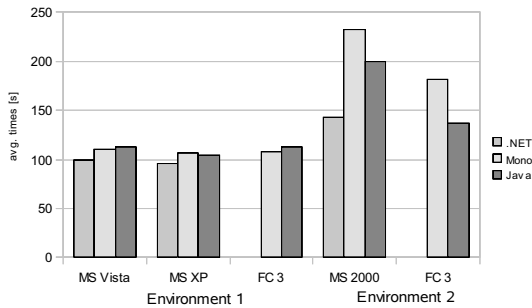


Fig. 4. Graphical comparison of times of computations for various operating systems, programming platforms and hardware.

The fastest computations have been reached in environment 1, under control of MS Windows XP, using .NET Framework. In second environment the fastest code was run under control of Linux using Sun Java platform.

D. Parallel case

Geological model described in the previous chapter and the same set of parameters were used for the parallel modeling of the heat transfer equation. Parallelism was introduced by domain decomposition of the computational domain (Fig. 5) and master slave paradigm (Fig. 6).

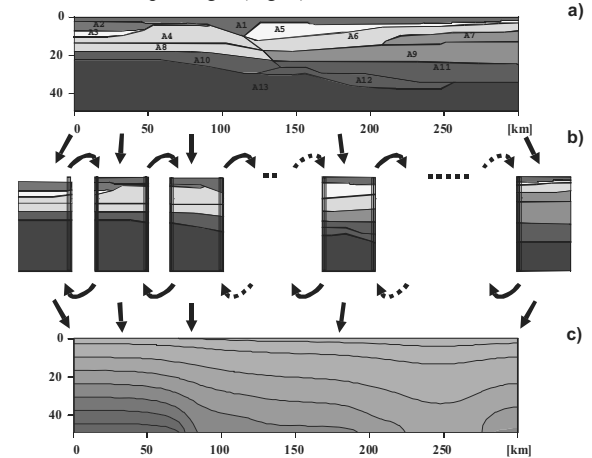


Fig 5. The domain decomposition schema. Global computational domain (a) is divided into smaller subdomains. Results of computation performed by slave nodes (b) are collected by master node (c)

Global computational domain was divided into as much subdomains as independent processors got involved in concurrent heat transfer equation solving. Assignment of computational subdomains to independent processors made computational process much faster and limited the risk of local

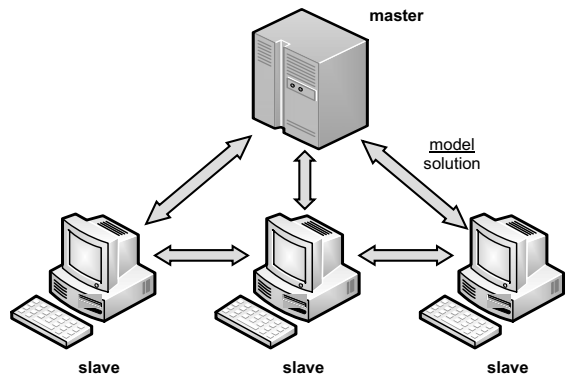


Fig 6. The environment for distributed heat transfer modeling.

RAM exceeding. One of the machines hosted master process, other hosted only slave processes. Master was responsible for creation of numerical model representation, division of the global computational domain, assigning and sending data to slave nodes and finally collecting heat flow modeling results. Slave processes performed computation and were exchanging subdomain border nodes after every step of calculation.

The intra slave communications was the most time consuming part of parallel algorithm. For a test model described above slaves exchanged 80MB of data for about two thousand times. Parallel algorithm with such dense communication pattern strongly limits number of slave nodes especially for small computational models. The entire tests were performed in computational hardware environment described in the previous section. Computation times received for various component-based software solutions are presented in Table IIIA and Table IIIB. There are minimal and average times of computing for consecutive numbers of nodes presented in figure 7. The speed up of parallelization is shown in the fig. 8.

TABLE IIIA

MINIMAL TIMES OF PROCESSING FOR HEAT TRANSFER MODELING

No of proc	Minimal time of processing [s]				
	.NET	Mono	Java	Mono	Java
	Win 2000	Win 2000	Win2000	FC 3	FC 3
1	179,47	241,30	240,33	169,21	166,54
2	126,17	162,86	140,58	115,22	113,26
3	119,92	160,42	127,95	132,05	109,79
4	108,92	141,52	116,02	111,13	91,70
5	104,61	134,28	104,70	103,63	82,91
6	107,77	135,00	95,94	109,62	70,67
7	107,27	137,20	89,79	98,72	74,13
8	104,13	134,59	85,49	98,86	81,32

TABLE IIIB

AVERAGE TIMES OF PROCESSING FOR HEAT TRANSFER MODELING

No of proc.	Average time of processing [s]				
	.NET	Mono	Java	Mono	Java
	Win 2000	Win 2000	Win2000	FC 3	FC 3
1	182,87	242,48	241,14	169,84	167,17
2	128,35	163,07	141,50	115,95	116,34
3	125,38	161,06	128,90	133,28	112,90
4	113,55	142,58	117,35	113,53	95,47
5	106,90	137,31	105,23	130,39	86,76
6	113,72	137,30	96,90	115,68	72,19
7	117,23	138,53	90,95	99,75	76,37
8	109,18	137,19	86,92	101,08	84,04

V. DISCUSSION

The shortest time of processing heat transfer modeling for a single case was reached for MS .NET platform in environment 1. In environment 2 (cluster of PC-s) the fastest combination was Sun Java and Linux (there was no Windows XP installation). For the same environment in distributed case the leader was also Sun Java as platform and Linux as operating system (similar scores are for synthetic tests [9]). The interesting point is the speed ups for these technologies - the network usage by communication was a bottle-neck for .NET and Mono - the speed up is not especially growing for four processors and more. The speed

up for Sun Java is higher for higher number of processors. It proves the necessary of parallelization for such a big computational problem like heat transfer modeling. Similar results were reached for seismic field modeling [8], but in this case the network transfer was smaller and speed ups results were close for all platforms.

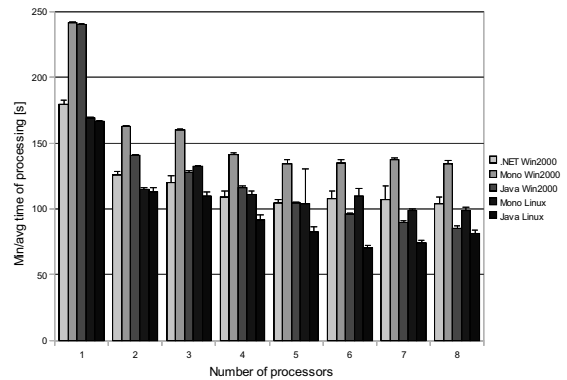


Fig 7. The minimal and average times for distributed simulation.

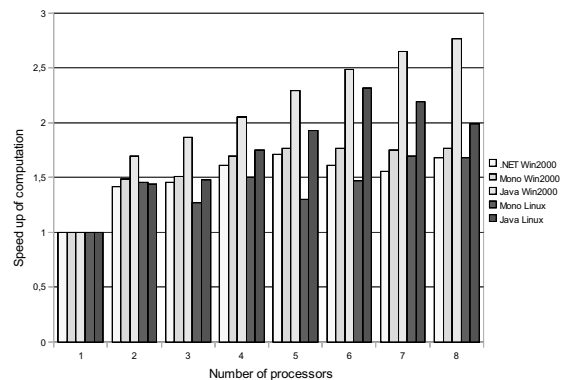


Fig 8. The speed up for distributed simulation.

VI. SUMMARY

There is an ability of construction the scientific numerical code in component platforms described in this article. The problem of heat transfer modeling is presented and the way of parallelization is shown. The performance of distributed solutions in case of parallelization is the main scope of experimental tests. Obtained speed up results proofs, that parallelization is a good way to solve a problem of computations efficiency.

ACKNOWLEDGMENT

This work was financed by the AGH - University of Science and Technology, Faculty of Geology, Geophysics and Environmental Protection as a part of statutory project number 11.11.140.561.

REFERENCES

- [1] A. Pięta, "Heat transfer modeling in deep geothermal application," in Science and supercomputing in Europe - report 2006, Cineca, HPC-Europa, Pan-European Research Infrastructure on High Performance Computing. Bologna 2006, Italy, p. 338–341.
- [2] M. Wróblewska, A. Pięta, "Deep lithosphere thermal modeling against surface temperature condition," in Near Surface 2008 - 14th European meeting of Environmental and engineering geophysics, 15–17 September 2008, Cracow, Poland.
- [3] D. Reilly., M. Reilly: Java Network Programming and Distributed Computing. Addison Wesley, 2002.
- [4] M. MacDonald: Microsoft .NET Distributed Applications: Integrating XML Web Services and .NET Remoting. Microsoft Press, 2003
- [5] H.J. Schonig, E. Geschwinde: Mono Kick Start. Sams, 2003.
- [6] E. Pitt, K. McNiff: java.rmi: The Remote Method Invocation Guide. Addison Wesley, 2001.
- [7] J. C. Tannehill, D.A. Anderson, R.H. Pletcher: Computational Fluid Mechanics and Heat Transfer, Second Edition, Taylor & Francis, Washington, 1997.
- [8] A. Kowal, A. Piórkowski, T. Danek, A. Pięta, "Analysis of selected component technologies efficiency for parallel and distributed seismic wave field modeling," in Innovations and Advances in Computer Sciences and Engineering, Springer, 2010.
- [9] A. Piórkowski, and D. Plodzien, "Efficiency analysis of the server-side numerical computations," in Computer Networks, 16th Conference, CN 2009, Wisla, Poland, June 16-20, 2009, Communications in Computer and Information Science, Springer Berlin Heidelberg, 2009.