Evolutionary Optimization using Enterprise Grid Based on Alchemi Framework

Wacław Kuś [1] and Tadeusz Burczyński [1,2]

[1] Silesian University of Technology, Department for Strength of Materials and Computational Mechanics, Poland, email: waclaw.kus@polsl.pl
[2] Cracow University of Technology, Institute of Computer Modelling, Poland, email: tadeusz.burczynski@polsl.pl

Abstract. This paper describes the evolutionary optimization performed using enterprise grid approach. The Alchemi framework is used. The Alchemi is built on top of .NET framework. The evolutionary algorithm works in parallel form and uses Alchemi Executors to perform computations of fitness function values for chromosomes. The experimental tests measuring Alchemi overhead during computations are presented.

1 Introduction

Grids are computer resources connected with networks. The resources in grids are processors, memories, storages, visualization equipment, scientific equipment, software. The enterprise and desktop grids are built on top of dedicated and non dedicated computers in firms and laboratories. Another definition according to [1] is as follows: Enterprise Grid Computing is managed architecture that aggregates the IT resources of a business data center into dynamically assignable pools. The Alchemi framework [4], [3] provide tools to create and use such grids. The serial and parallel evolutionary algorithm are well known and in common use nowadays [6], [5]. The parallel evolutionary algorithm based on Alchemi framework is presented in the paper. The parallelization is performed during one step of the evolutionary algorithm - in the fitness function evaluations for chromosomes. The goal of the paper is to measure the overheads of presented evolutionary algorithm implementation connected with use of Alchemi framework and enterprise grid approach. The simple mathematical function is used. The results allows us to predict the speedup which can be obtained using Alchemi and the problems with different fitness function evaluation time.

2 The Alchemi framework

The Alchemi framework is based on Windows .NET. This makes Alchemi useful only on hardware using Windows operating system. The number of computers with Windows OS installed is quite big, mostly because of ease of use and the wide palette of programs. There are also many other operating systems used in laboratories and firms. Fortunately the Alchemi can interface with other grid components even if they are not using the

Windows OS. The gridbus [2] resource broker can be use with Alchemi-based and Globus-based parts of a grid. The Alchemi consists of the following elements:

- Alchemi Manager - the central host with scheduling capabilities, one manager is needed for grid or part of grid
- Alchemi Executors - the hosts performing computations
- Alchemi Cross Platform Manager - web services based manager with ability to communicate with non-Alchemi parts of grid (e.g. gridbus resource broker)

The Alchemi Manager host can also be used as a Executor. The security policy is based on usernames and passwords. The users can be grouped. The end-users, executors and administrator groups are available by default. The information about users, tasks, jobs, executor hosts are stored in database connected with Alchemi Manager (the SQL Server, MySQL and "in-memory" databases are supported). The communication between Alchemi Manager and Executors are performed using TCP/IP selected ports. These ports need to be available through firewalls. The architecture of the enterprise grid based on Alchemi is shown in Fig. 1.

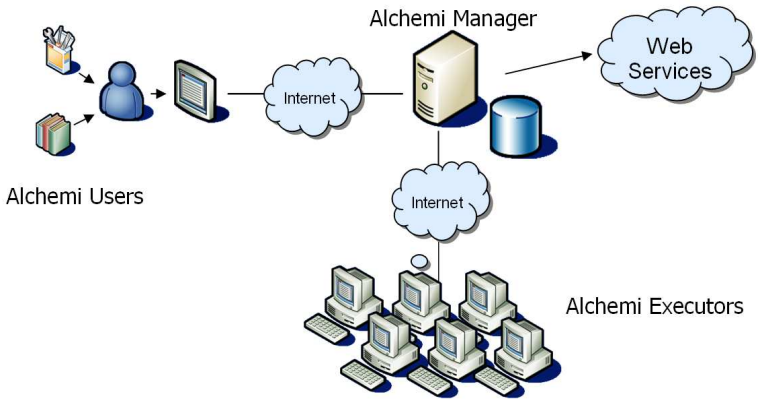

Figure 1. Architecture of enterprise grid based on Alchemi framework.

The most important advantage of Alchemi framework is the API provided for grid applications developers. The architecture of Alchemi framework is presented in Fig. 2. The Alchemi can be used without using Alchemi API, the executables for submitting, controlling and retrieving task and jobs are provided. The Alchemi is written in C# using .NET framework. The execution of grid applications is performed using remote threads running on executors. Communication between remote threads is prohibited (in current version 1.0.3). The task submitted to the grid consists of threads. The Alchemi framework contains also Alchemi Console for monitoring tasks, threads and executors. The Alchemi Manager and Executor can work as a system service. The executing remote threads have lower priority - this is important when non dedicated computers are present in the grid. The Executor can also be used in special mode in which communication is initialized only by Executor. This mode allows to hosts to be used behind firewall and using NAT.

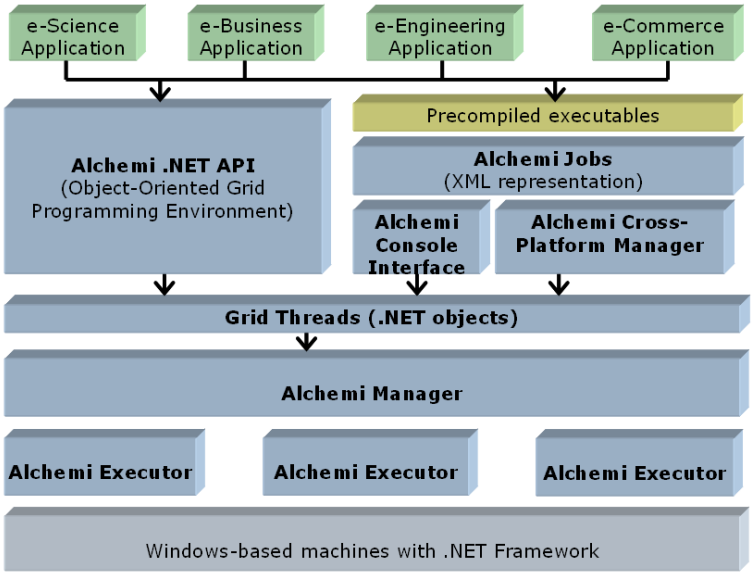

Figure 2. Alchemi architecture.

The lack of information about resources available for grid application is one of the Alchemi's disadvantages. Information about memory, processor speed, empty disk space is important for most grid applications.

3 Numerical test

The overhead of Alchemi framework has been measured for one and two executors. The test configuration is presented in Fig. 3. The tests were performed for different number of chromosomes in population. The simple fitness function F was used because the goal of the experiment was not to find the best evolutionary algorithm parameters, but to determine the overhead. The fitness function was as follows:

$$F(ch) = g_1 \cdot g_2 \tag{1}$$

where g_1 and g_2 are chromosomes **ch** genes. The constrains on genes values were $[0, 1000]$. The number of generations was equal to 100.

The fitness function was computed using a program called solver. The parameters of the program defines input file with chromosome and output file. The program writes the fitness function to the output file. The size of the program and input data files was about 50kbytes. The size of the output file is few bytes. The transfer of the solver program, input and output files is performed for each chromosome (each chromosome can be evaluated by different Executor). The 1Gb LAN was used during tests. The test were performed 20 times. The average results for one generation obtained with one and two executors are presented in Table 1.

The average execution of one chromosome using one executor was $0.085s$ and using two executors $0.074s$. The time need for compute all chromosomes from population is smaller

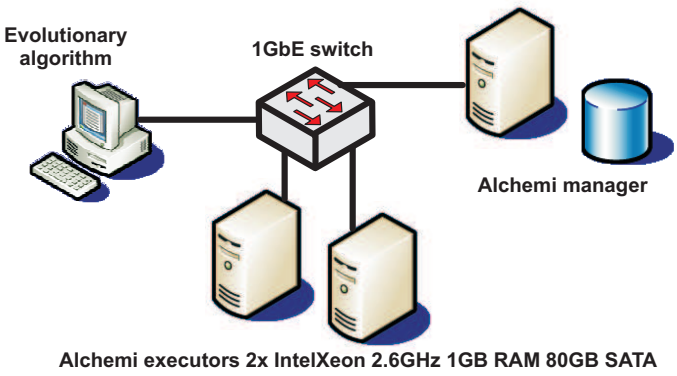

Figure 3. The test configuration.

Table 1. The average tests results for one generation.

number of chrom. *nchr*	One executor		Two executors	
	time needed to compute all chromosomes in population [s] $t_{Alchemi}(nchr, 1)$ (Alchemi overhead)	time needed to compute fitness function for one chromosome [s]	time needed to compute all chromosomes in population [s] $t_{Alchemi}(nchr, 1)$ (Alchemi overhead)	time needed to compute fitness function for one chromosome [s]
10	1.1105	0.1110	1.1941	0.11941
25	2.2405	0.0896	1.6480	0.06592
50	3.4859	0.0697	2.9332	0.05866
100	7.0986	0.0710	5.2400	0.0524

when two executors are used and the number of chromosomes is 25 or more. The use of Alchemi framework for mathematical function optimization is profitless, but presented tests allows us to predict Alchemi overhead during other tasks, like optimization of mechanical structures. The Alchemi overhead has value very close to the times presented in Table 1 because the fitness function evaluation time is very small and can be neglected. The Table 2 presents predicted speedups for different fitness function evaluations times for two cases - with one and with two executors. The speedup s is measured as:

$$s = \frac{t_1}{t_n} \tag{2}$$

where t_1 is time needed to perform optimization using one processor without Alchemi, the tn is wall time needed to perform the optimization with use of Alchemi framework and n executors. The t_1 is obtained from multiplying number of chromosomes $nchr$ and time need for evaluate one fitness function value by solver t_{fi} for different problems i ($t_{fi} = 0.1, 0.5, 1, 2, 5, 10, 30$ seconds)- the time need for evolutionary operators, selection etc. is omitted:

$$t_1 = nchr \cdot t_{fi} \tag{3}$$

The t_n is computed using formula:

$$t_n = \frac{nchr \cdot t_{fi}}{n} + t_{Alchemi}(nchr, n) \tag{4}$$

where n is a number of executors, $t_{Alchemi}(nchr, n)$ is Alchemi framework overhead for different number of chromosomes (the times are taken from Table 1.). The results presented in Table 2 can be used during planning optimization with use of time consuming fitness function evaluators. The under 1 speedup - slow down of computations will be achieved when one executor will be used - this is obvious. The speedup for two executors is under 1 for considered numbers of chromosomes. The problems where evaluation of fitness function value takes 0.5s can obtain speedups of computations above 1. This is true for many optimization problems including optimization of mechanical structures.

Table 2. The predicted speedups.

One executor case							
number of chrom.	speedup for $t_{fi} =$ 0.1	speedup for $t_{fi} =$ 0.5	speedup for $t_{fi} =$ 1	speedup for $t_{fi} =$ 2	speedup for $t_{fi} =$ 5	speedup for $t_{fi} =$ 10	speedup for $t_{fi} =$ 30
10	0.4738	0.8183	0.9000	0.9474	0.9783	0.9890	0.9963
25	0.5274	0.8480	0.9178	0.9571	0.9824	0.9911	0.9970
50	0.5892	0.8776	0.9348	0.9663	0.9862	0.9931	0.9977
100	0.5848	0.8757	0.9337	0.9657	0.9860	0.9930	0.9976
Two executors case							
10	0.5903	1.3535	1.6144	1.7867	1.9088	1.9534	1.9842
25	0.8627	1.5827	1.7670	1.8763	1.9486	1.9740	1.9912
50	0.9203	1.6199	1.7900	1.8892	1.9541	1.9768	1.9922
100	0.9766	1.6534	1.8103	1.9004	1.9589	1.9793	1.9930

4 Conclusions

The implementation of parallel evolutionary algorithm using Alchemi framework has been presented. The overhead of the Alchemi framework during tests on simple mathematical function were measured. The prediction of speedups for optimization problems with different times needed to compute fitness function value were shown. The future research will be concentrated on verifying presented predicted results experimentally.

Acknowledgement

This research is financed from the Polish science budget resources in the years 2005–2008 as the research project.

Bibliography

[1] Enterprise grid alliance, grid computing overview, http://www.gridalliance.org/.

[2] Gridbus project web pages http://www.gridbus.org/.

[3] Project alchemi sourceforge web page http://sourceforge.net/projects/alchemi.

[4] L. Akshay, B. Rajkumar, R. Rajiv, and V. Srikumar. Peer-to-peer grid computing and a .net-based alchemi framework. In *High Performance Computing: Paradigm and Infrastructure, Laurence Yang and Minyi Guo (ed.)*, pages 1–21. Wiley Press, 2005.

[5] J. Arabas. *Evolutionary algorithms lectures(in polish).* WNT, 2001.

[6] Z. Michalewicz. *Genetic algorithms + data structures = evolutionary algorithms.* Springer-Verlag, 1996.