Research of Genetic Algorithm for searching optimal configurations of computing cluster with virtual machine nodes

I. Bilokon, S. Pohorilyi

The genetic algorithm proposed, its parameters defined and researched to solve optimal configurations searching problem for computing cluster with virtual machine nodes via dynamic cluster architecture reconfiguration; experimental data presented and approaches for genetic algorithm parameters quality evaluation suggested.

Introduction

The rapid development of modern hardware-software virtualization technologies and the desire of computational resources utilization optimizing lead to urgent problem of building compute clusters with dynamically reconfigurable architecture based on virtual machines (VM) [1, 2]. In the papers [1, 2] expediency of Microsoft Hyper-V R2 virtualization platform usage in conjunction with Microsoft HPC Server 2008 suite to create such a computing systems was proved. Implementation and running of compute cluster with VM nodes requires solving several tasks. Most of them are relevant hardware and software integration, management infrastructure development, user interfaces creation, optimization of computational resources workload.

In the papers [2, 3, 4] expediency of genetics algorithm (GA) usage to search for optimal configurations was proved and central processing unit (CPU) performance counters utilization for configurations quality evaluation was suggested.

The aim of this paper is to develop and research GA to solve optimal configurations searching problem for computing cluster with dynamically reconfigurable architecture with virtual machine nodes.

The genetic approach to optimize the workload of cluster resources

Let's consider a computing cluster at a certain period of time, during which the status of its task queue remains unchanged (static queue). In order to minimize waiting for computational resources it is essential to find such cluster configurations and the task queue processes per node distribution, where competition for the resources will be minimal.

Formulation of the optimal cluster configurations search problem.

According to the cluster computational resources utilization optimization problem formalization presented in [4], there is following formulation of optimal configurations searching problem: find z to meet conditions:

$$\begin{cases} F(z) \to max \\ (\forall i \in \{1, 2, \ldots, N\})\,[H_i(z) \leq \sum_k L_{ik}] \\ N = const \end{cases} \quad (1)$$

Here z — computing system configuration, $F(z)$ — the total load of processors of all VM hosts (VMH), $H_i(z)$ — the total load of processors of all VMs located on VMH with index i, L_{ik} — load of k-th CPU core of i-th VMH, N — number of VMH. Let's use approximations suggested in paper [4] which allow switching from VM configurations to tasks' processes grouping via equating the number of VM's processors and the number of tasks' processes on the VM.

GA parameters definitions

Formally, searching methods via GA could be defined as a following function [5]:

$$GA = GA(P_0, N, L, f, \Omega, \Psi, \theta, T) \tag{2}$$

Where P_0 — initial population, N — number of chromosomes in population, L — number of genes in chromosome, f — fitness function (FF), Ω — selection operator, Ψ — crossover operator, θ — mutation operator, T — stop criteria. Let's consider GA parameters definition according to the problem.

The peculiarity of the GA application to the formulated problem is significant epistasis (internal relationship between genes). This phenomenon negatively impacts the search, so it is necessary to reduce the impact in operations' definitions.

Genes representation. Selection and mutation operators

Let's represent each chromosome in population as following matrix:

$$I = A_{ij} \tag{3}$$

A_{ij} is number of i-th task processes located on VMH with index j. Each row of the chromosome representation matrix (CRM) is process distribution configuration for corresponding task and each column is process configuration of all tasks on corresponding VMH. In this research we are going to compare 2 types of selection: proportional selection and tournament selection [5].

Mutation operation is one item modification with supplied probability per each row of the CRM. Mutation probability is calculated separately for each row.

FF

In case of single task in queue FF could be calculated using following formula [4]:

$$\phi_i(A_{ij}) = \frac{\sum_j \overline{X}_j}{t} \tag{4}$$

Where $\overline{X}_j$ is average computing nodes' CPU load calculated in certain time intervals, t is time interval when measurements were performed. In practice it is advisable to use normalized FF:

$$f_i = \frac{\phi_i}{\phi_i^{max}} \tag{5}$$

Let's calculate FF for set of tasks as follows:

$$f(A_{ij}) = \sum_i f_i \tag{6}$$

Because performance measurements for FF occur only for certain key configurations [4], let's perform FF estimation for other cases using related principles from [4]. Let's consider the approach in more detail.

Let r is number of processes for some task, p is number of processes per VM, g is number of VMs containing computing processes per VMH and

$$g = \begin{cases} r \div p & , r \mod p = 0; \\ r \div p + 1 & , r \mod p > 0. \end{cases} \tag{7}$$

Then relationship between these values would be following: $p * g \leq r + p$. For key configurations these values would be following [4]:

- K1: $p = 1$, $g = 1$;
- K2: $p = \begin{cases} C_{max} & , r > C_{max} \\ r & , r < C_{max} \end{cases}$, $g = \begin{cases} r \div p & , r \mod p = 0; \\ r \div p + 1 & , r \mod p > 0. \end{cases}$, C_{max} — max available number of VM's CPU parallel cores;
- K3: $p = \begin{cases} C_{max} & , r > C_{max} \\ r & , r < C_{max} \end{cases}$, $g = 1$.

Let's split all the configurations available into 2 classes: those that can be represented by values r, p, g ($z = z(r, p, g)$) and those that cannot. Let's consider FF evaluation algorithm for first class configurations.

1. $z(r, p, g)$ match any Ki configuration? Yes: $f(z) = f(Ki)$, goto step 9.
2. $p * g > C_{max}$? Yes: goto step 5.
3. $r > C_{max}$? Yes: $f(r, p, g) = f(r, C_{max}, 1) * p/C_{max}$. Goto step 5.
4. $f(r, p, g) = f(r, r, 1) * p/r$.
5. Search for FF value for configurations with the same $p * g$ value (varying r parameter). If exists then $f(r, p, g) = f(q, p, g) * q/r$ (or $f(r, p, g) = f(q, g, p) * q/r$, depending on what value exists).
6. If the value was found on previous steps goto step 9.
7. Search for existing FF values $f(p*(g-k), i, p*(g-k)/i)$, where k can be 0 or 1, i decreases from C_{max} to $p * (g - k)/D_{max}$, D_{max} is max applicable number of VM on VMH. If found: $f(r, p, g) = f(p * (g - k), i, p * (g - k)/i) * p * (g - k)/r$. Goto step 9.
8. $f(r, p, g) = 0$.
9. End.

FF values for configurations that belong to the second class could be identified by casting to first class configurations:

$$f = \sum_{l} \frac{f(z_l)}{Q} \tag{8}$$

z_l is first class configuration formed from tasks' processes configuration that are located on VM with index i as though remaining VMs have the same process configuration; Q is number of VMs that contain processes of the task.

Crossover operator

Within this research we will use 2 variants of crossover.

- Per-row crossover. Implemented as crossover of corresponding CRM's rows for 2 individuals (each row represents process configuration of certain task in cluster) using one-point crossover [5].
- Table crossover. Implemented as choosing random CRM's item and running segment through it that cuts "isosceles triangle" from the CRM. Then swap values that appear in the triangle for 2 individuals.

In both cases crossover is performed between 2 generations (previous and current) that results next generation. In this paper we will compare 2 variants of choosing individuals for crossover:

- random choice;
- choice based on similarity. The similarity is estimated by the formula:

$$\delta = \sum_{ij} (a_{ij} - b_{ij})^2 \tag{9}$$

Such an approach should minimize the negative impact of epistasis during crossover but significantly increase algorithm complexity.

Initial Population. Number of chromosomes in population

Search progress can greatly depend on initial population because right choice of initial population members favors faster convergence. Taking into account the above definition of genes and FF, during generation of an initial population using pseudo random approach the following criteria should be considered:

- the total number of task processes in a chromosome is an even number and should not exceed the maximum value (in our case, this is the maximum number of processes, which were tested with key configurations);
- processes in chromosome should be distributed as equally probable as possible for each VMH.

Number of chromosomes in population will be kept constant and equal to 256.

Testing GA for solving optimal configurations searching problem

GA quality estimation approaches

The peculiarity of any GA is a significant dependence of the result from the algorithm parameters [5]. Because of the absence of general approaches to determine GA parameter values an important problem is parameter selection for particular GA implementation. Let's determine following comparison criteria to solve the problem: comparison with exhaustive search (ES), best individual search progress test [6], average individual search progress test.

Results of the experiments

Data for FF computation was retrieved as a result of performance counter measurements [4] for queue with 3 tasks each of which is LINPACK utility. In figure 1 a function of best individual found during search process is shown for ES and GA with per-row crossover, random parent choice and mutation probability equals 0,001. Selection type is proportional. The chart shows that GA

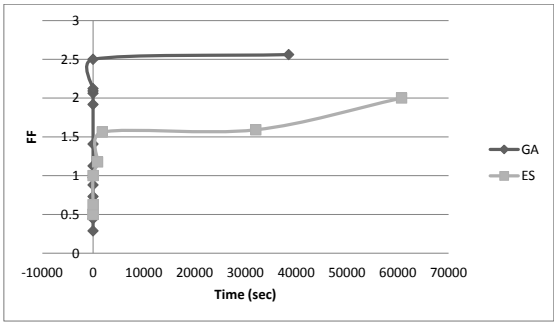

Figure 1. Comparison of best individual search speed for GA and ES.

best individual search is significantly faster than ES.

In figure 2 functions of FF depending on generation for different mutation rates are shown. GA has similar parameters. The best result coming with mutation rate 10^{-6}.

Figure 3 shows similar dependencies but for GA with tournament selection for various tournament rates (rate value determines number of individuals that participate in the tournament). Mutation rate is 10^{-6}. The best result is observed if tournament rate equals 5. Result hardly differs from the previous one so it may indicate the futility of results improving by selection operator choice.

A common problem of the algorithms mentioned is the premature convergence of average FF to value that is significantly smaller than the maximum found. This can happen as a result of low fittest offspring creation from good fittest parent individuals because of epistasis. To reduce the negative impact let's utilize Table crossover. Results are shown in figure 4. On the chart (a)

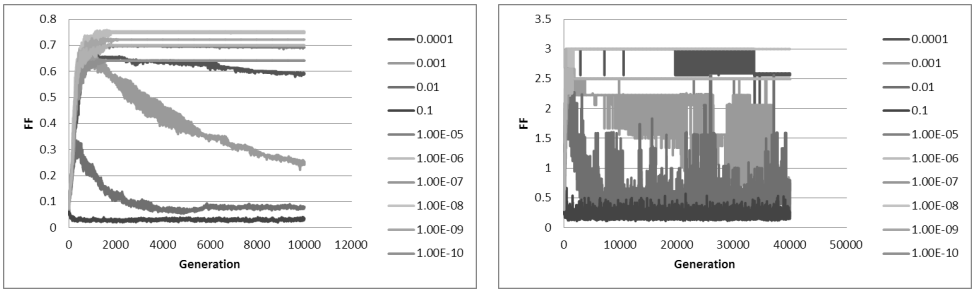

Figure 2. Average (a) and maximum (b) value of FF depending on generation for various mutation rates.

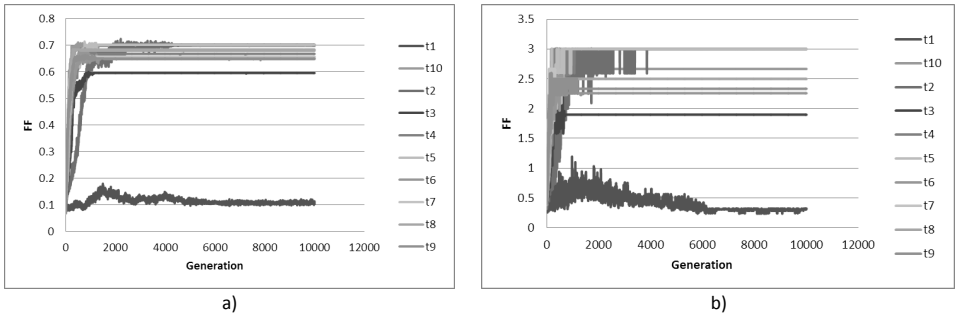

Figure 3. Average (a) and maximum (b) value of FF depending on generation for various tournament rates.

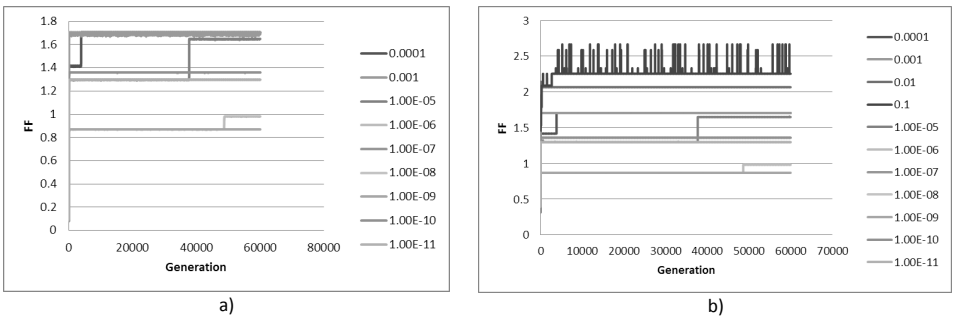

Figure 4. Average (a) and maximum (b) value of FF depending on generation for various mutation rates (Table crossover).

we can see FF average value improvements although premature convergence persists. Also we can see search capabilities deterioration, which indicates that the probabilistic nature of Per-row crossover covers much wider range of possible variants.

Let's return to Per-row crossover. Now to reduce the negative impact of epistasis we will use chromosome pair choice for crossover based on similarity. The results are shown in figure 5. In case of mutation rate equals $0,1$ we can see growth of both average and maximum FF values. In figure 6 test results for the same algorithm but for much more generation number and mutation rate $0,1$ are shown. We can see more monotonous nature of both average and maximum FF value growth than in previous cases. This indicates further research of such an approach to be promising. Relatively high mutation rate contributes the best result because in order to extend search area it is essential to compensate low probabilistic approach for parent chromosomes. In further research we are going to investigate possibility to implement more probabilistic way of choosing parent chromosomes that could lead to lower algorithm complexity and search time. Also we are going to improve GA to include random access memory size into FF evaluation.

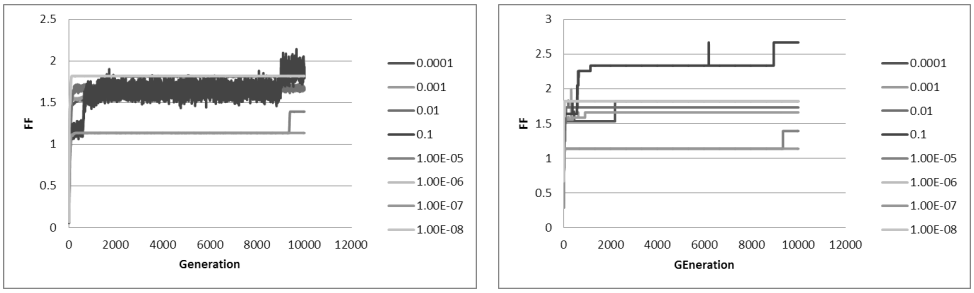

Figure 5. Average (a) and maximum (b) value of FF depending on generation for various mutation rates (crossover based on similarity).

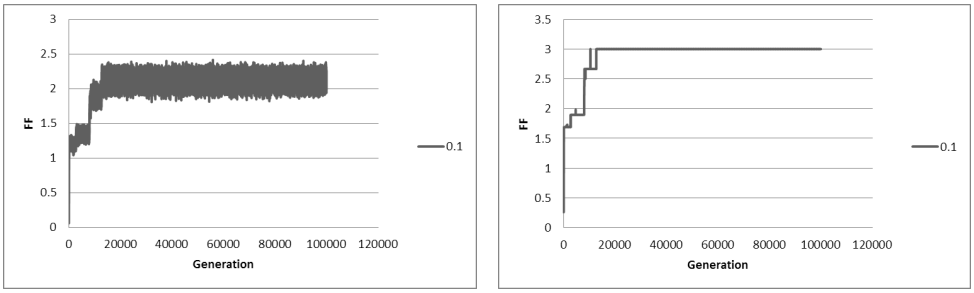

Figure 6. Average (a) and maximum (b) value of FF depending on generation for mutation rate 0,1 (crossover based on similarity).

Conclusion

1. The GA proposed to solve optimal configurations searching problem for computing cluster with dynamically reconfigurable architecture with virtual machine nodes.
2. Suggested approaches for GA parameters quality evaluation allowed perform GA parameters comparison.
3. GA parameters comparison showed that:
   (a) negative impact of the epistasis phenomena should be got rid via crossover based on similarity;
   (b) more probabilistic nature of crossover improves GA search capabilities.

References

[1] I.V. Bilokon, D.B. Gryaznov, S.D. Pogorilyy, Building of dynamically reconfigurable computing architecture using virtualization technologies, Bulletin of the Taras Shevchenko National University of Kyiv, Series: Radiophysics and Electronics, 2010, N 14, P 4-7.

[2] S.D. Pogorilyy, I.V. Bilokon, Y. V. Boyko, Dynamic reconfiguration of computing cluster resources, Mathematical Machines and Systems, 2012, N 3, P 3 - 18.

[3] I.V. Bilokon, D.B. Gryaznov, S.D. Pogorilyy, Testing of computing system configuration impact on parallel algorithm software implementations, Proceedings of the VII International conference "Electronics and applied physics", October, 19-22, 2011, Kyiv, Ukraine.

[4] Pogorilyy S., Bilokon I., About computational resources utilization problem for compution cluster with virtual machine nodes, Problems in programming, 2012, N 2-3.

[5] Non-iterative, evolutionary and multiagent methods of fuzzy logic and neural network models synthesis: Monography, edited by S. Subbotin, Zaporizhya: ZNTU, 2009, P 98 - 162.

[6] D. Goldberg, K. Deb A Comparative Analysis of Selection Schemes Used in Genetic Algorithms, Proc. of 1st workshop FOGA/CS-90 on July 15-18, Bloomington, US, 1990, San Mateo, 1991, p. 69 - 93.

Authors

Ivan Vasylovych Bilokon — post graduate studend, faculty of radiophysics, Taras Shevchenko national university of Kiev, Kiev, Ukraine; E-mail: *deimos@univ.net.ua*

Serhii Demianovych Pohorilyi — Doctor of Technical Sciences, Professor, department of computer engineering, faculty of radiophysics, Taras Shevchenko national university of Kiev, Kiev, Ukraine; E-mail: *sdp@rpd.kiev.ua*