Immune-inspired Technique for Optimizing Server's Energy Consumption

Tudor Cioara, Cristina Bianca Pop, Ionut Anghel, Ioan Salomie, Mihaela Dinsoreanu,
Irina Condor and Fodor Mihaly
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
{tudor.cioara, cristina.pop, ionut.anghel, ioan.salomie, mihaela.dinsoreanu}@cs.utcluj.ro

Abstract— **This paper presents an immune-inspired technique for optimizing a server energy consumption. The proposed technique is similar with an artificial immune system associated to a server, aiming to detect non-optimal server energy consumption states and to take the appropriate actions that would bring the server into an optimal state. The optimization technique has two main stages: an initialization stage and a self-optimization stage. In the initialization stage the server is monitored for a specific period of time to collect energy consumption historical raw data for identifying associations between the server energy consumption states and the appropriate optimization actions. In the self-optimization stage, energy consumption server state snapshots are taken at regular time intervals and formally represented using a biologically-inspired antigen model. The obtained antigen is then classified as self (optimal energy consumption state) or non-self (non-optimal energy consumption state) using a set of detectors obtained in the initialization stage. For non-self antigens a biologically-inspired clonal selection approach is used to determine the actions that need to be taken to bring the server in an optimal energy consumption state.**

Keywords—energy consumption; immune-inspired; self-optimizing; clonal selection; negative selection

I. INTRODUCTION AND RELATED WORK

Nowadays IT infrastructures and data centers are becoming more and more complex. This complexity leads to high energy consumption and in this context, the need for developing techniques that optimize energy consumption while preserving performance requirements is a must. It has been noticed that biology offers many clues regarding the construction of optimization systems. Biological systems are efficient, robust, flexible, self-organizing, self-repairing, self-optimizing, self-protecting, self-adapting, all these characteristics being desired in any IT system. Therefore, it seems natural that by inspiring from the nature observed processes we can develop IT systems able to (*i*) self-organizing and self-optimizing themselves like colonies of ants, flocks of birds or banks of fish, (*ii*) self-protecting and self-managing themselves as the autonomous nervous system or (*iii*) self-healing themselves as the immune system does. However we should be aware that there is a limit for this inspiration. We should not attempt to fully imitate biological systems because it is likely to have very limited success, as the flight pioneers have noticed [1]. The methodology required to develop biologically-inspired techniques includes the following steps: (1) the analogies between biology and IT systems at the conceptual level should be identified, (2) the biological entities, relationships and processes should be modeled from the computer science perspective and (3) a reverse mapping should be performed for validation purposes.

Our vision is that techniques for optimizing the server's energy consumption can be designed and developed by inspiring from biology. In this paper, we present an immune-inspired technique for optimizing server energy consumption while preserving its set of Green Performance Indicators and Key Performance Indicators (KPI) [2] [3]. This technique is similar with an artificial immune system associated to a server being able to detect non-optimal server energy consumption states (similar to biological pathogens) and to take the appropriate actions (similar to the biological immune response) required to bring the server into an optimal state. The optimization technique has two main stages: an initialization stage and a self-optimization stage. Within the initialization stage the server is monitored for a specific period of time to collect energy consumption historical raw data with the goal of identifying associations between the server energy consumption states and the appropriate optimization actions. Such an association represents an artificial immune cell composed of a detector (server energy consumption state) and an effector (the optimization actions).

The self-optimization stage is designed as a control feedback loop with the following MAPE phases: **M**onitoring, **A**nalyzing, **P**lanning and **E**xecution. In the monitoring phase, energy consumption server state snapshots are taken at regular time intervals and formally represented using a biologically-inspired antigen model. The analysis phase classifies the current antigen as self (optimal energy consumption state) or non-self (non-optimal energy consumption state) using the set of detectors obtained in the initialization stage. The planning phase determines the actions (referred as effectors) that need to be taken to bring the server in an optimal energy consumption state using a biologically-inspired clonal selection approach. Within clonal selection, the affinity between the existing effectors and the current antigen is evaluated. The high affinity effectors are cloned and mutated to obtain the best effector to be run in the execution phase aiming at bringing the server into an optimal state.

978-1-4244-8230-6/10/$26.00 ©2010 IEEE

In the **biologically-inspired computing** domain literature we have identified two main research directions: *immune-computing* and *swarm intelligence*.

Immune-computing uses ideas inspired by the biological immune system specific concepts and processes to design algorithms and techniques used to solve complex problems such as, learning, optimization and adaptive control [24]. The most important immune-inspired computational models relevant for the current research are the *negative selection* and the *clonal selection* models.

The negative selection models are inspired by a biological process aiming at training immature immune T-cells to correctly discriminate self molecules from the non-self ones. The first negative selection algorithm [4] was designed to generate detectors (T-cells) able to identify situations in which unauthorized changes led to the appearance of non-self strings (non-self antigens) in a set of self-strings that had to be protected [22]. In [5], the negative selection algorithm is applied to network intrusion detection. In this case, self-cells are represented as the frequently occurring data paths (source ip address, destination ip address, communication port), while non-self cells are considered to be the data-paths that are not normally observed on the network.

The *clonal selection* models are inspired by the biological process in which B-cells specialize through affinity maturation and somatic hyper-mutation to provide a specific immune response when a newly pathogen is encountered. The first proposed clonal selection algorithm [6] was applied to function optimization and pattern recognition. In the case of function optimization, the candidate solutions represent the immune cells, while the antigen is the function that needs to be optimized. The value of the function for a particular solution is similar to the affinity between the immune cell and the antigen. In pattern recognition problems, the aim of applying the clonal selection algorithm is to produce a set of elements which can be used to recognize specific patterns [8]. The clonal selection algorithm can be used to solve other optimization problems, such as combinatorial optimization [7]. In [9], the clonal selection algorithm is used for selecting the optimal solution in automatic Web service composition. In this approach, a candidate Web service composition solution is mapped to an immune cell, a multi-criteria function that evaluates the QoS and semantic quality of a composition solution is mapped to an antigen, while affinity is the value of the multi-criteria function for a specific composition solution.

Swarm intelligence aims at designing techniques and algorithms inspired by the collective behavior of social insects, birds, fish and even humans. The most important techniques that have been developed are *ant colony optimization (ACO)* [10] and *particle swarm optimization (PSO)* [11].

The *ant colony optimization techniques* are inspired by the foraging behavior of ants. This technique considers a set of artificial ants which identify the solutions of an optimization problem and use a stigmergic type of communication (similar to real ants that communicate through the trails of pheromone they leave on their way) to exchange information about the quality of the solutions they find [10]. Ant colony optimization can be used to solve combinatorial optimization problems [12] and to find the shortest path in telecommunication networks [13]. The stigmergic communication using pheromones is used in [14] as a means of communication between agents in a wireless sensor network. Agents emit replication or migration pheromones encoding sensor data according to their local and external network conditions [14]. [15] applies pheromone-based communication in pervasive environments as a means of communication between mobile agents. [16] presents a method for data clustering inspired by the way ants organize objects in clusters according to their properties. Within this method, a set of artificial ants pick up items occupying random (initially) positions on a grid and deposit them in areas containing other similar items.

The *particle swarm optimization* techniques are inspired by the collective behavior of birds in search of food. In this technique a number of artificial particles which are collectively moving in a search space are used to find the global optima [22]. The technique is applied to various domains such as function optimization [17] or data clustering [18].

Regarding the use of **biologically-inspired techniques for power management**, few approaches can be found in the literature. Since biological systems naturally tend to conserve their energy, many simple principles found in the biological systems might be used in IT power management [19]. The adoption of biological principles such as decentralization, autonomy, natural selection or symbiosis in the process of designing and building applications or services on top of server farms is a novel research direction [20]. A service is designed as a biological entity, equivalent to an individual bee in a bee colony that competes or collaborates for computing resources. Using natural selection principles, the services that waste energy (i.e. services that gain resources but fail to use them) are banned for execution. In [14] a biologically-inspired agent based approach is used to manage the energy consumption in a wireless sensors network. The agent behavior focuses on biologically-inspired actions (e.g. pheromone emission, migration, death), each of them having an energetic cost associated. Through these actions, the life time and the state of each agent evolve autonomously and there is no need of a centralized control unit. In [21] swarm intelligence is combined with immune mechanisms to design network applications that adapt to dynamic changes in the network.

This paper is structured as follows: section II presents an immune-inspired model for server energy consumption optimization, section III describes the biologically-inspired algorithms, section IV presents the experimental results while section V concludes the paper and shows the future work.

II. IMMUNE-INSPIRED MODEL FOR SERVER ENERGY CONSUMPTION OPTIMIZATION

After a thorough analysis of the biologically-inspired computing models we reached the conclusion that the biological immune system would represent a good source of inspiration in designing a technique for energy optimization. In the following sub-sections we present the immune concepts and processes that are relevant for our approach as well as the

mapping of these concepts to the problem of energy optimization.

A. Biological Background

The main objective of the biological immune system is to protect a living organism against harmful pathogens (antigen presenting cells that might cause diseases). In order to accomplish this, the immune system provides a collection of countermeasures able to detect and eliminate pathogens [22]. These countermeasures are provided by the immune system constituents who circulate throughout the body in the blood and the lymph and perform their functions in a distributed manner, interacting only through localized rules [25].

The immune system can be viewed as a layered architecture. When a pathogen attacks a living organism it first needs to penetrate an external defense layer formed by the skin or the membranes that cover organs. If the pathogen succeeds to go through this first layer it interacts with a second defense layer which is represented by the innate immune system. The innate immune system consists of a set of detector and effector immune cells capable of rendering the pathogen harmless [22]. However, these two first defense layers are unspecific (they do not make any distinction between pathogens) and do not change over time.

In time, pathogens adapt themselves such that the immune system will no longer have the suitable detector and effector cells to use. This is the moment in which the third defense layer, corresponding to the adaptive immune system, comes into action by generating the appropriate detector and effector cells. It is very important that the new cells do not detect the self-cells as a pathogen, since this will lead to autoimmune diseases. Furthermore, the new cells must be efficient in the fight with the pathogen.

The generated cells pass through several phases until they are considered as part of the immune system [22]. The first phase consists of a negative selection process: each cell is tested against molecules of the host for auto-reactivity and it is eliminated if it proves to attack the host tissue. Next, through clonal selection, new and more specific effectors are generated: the effectors with the highest affinity with respect to the pathogen clone themselves. The clones are submitted to an affinity maturation process aiming at increasing their specificity for the invading pathogen. The resulting matured clones are submitted to a new selection process in which the clones having low affinity are eliminated while the ones with high affinity are differentiated into plasma and memory cells. The plasma cells secrete antibodies used to immediately eliminate the pathogen, while memory cells are kept in the immune memory to assure that the immune response to a similar pathogen encountered in the future will be much faster.

In conclusion, the immune system issues two types of immune responses: (*i*) a slower primary immune response - issued at a first encounter with an unknown pathogen, situation that requires the generation of new detector and effector immune cells and (*ii*) a faster secondary immune response - issued at an encounter with a known pathogen, situation that requires the use of the appropriate immune cells stored in the immune memory.

B. From Biological Immune Systems to Energy Optimization

To optimize the energy consumption at a server level we have considered the adaptive immune system's defense layer. One of the most important issues that need to be addressed in designing biologically-inspired computing techniques is to map the biological concepts to the problem that needs to be solved. In table I we present a mapping of adaptive immune system concepts and processes to the problem of server's energy optimization.

TABLE I. MAPPING OF IMMUNE SYSTEM CONCEPTS TO THE PROBLEM OF SERVER'S ENERGY OPTIMIZATION

	Immune System Concept	**Energy Optimization**
System	Adaptive immune system	Used to identify new server energy consumption states and determine the associated optimization actions
Entities	Antigen	The current server energy consumption state
	Self antigen	Optimal server energy consumption state
	Non-self antigen	Non-optimal server energy consumption state
	Immune cell - Detector	Values of server energy consumption states
	Immune cell - Effector	Optimization actions
	Clone	Copy of the optimization actions
	Immune memory	Knowledge base: server energy consumption states and associated optimization actions
Processes	Primary response	First encounter with an unknown server energy consumption state, generate the appropriate optimization actions
	Secondary response	Encounter with a known server energy consumption state, select the adaptation action plan from the knowledge base
	Affinity between antigen and immune cell	Similarity between the current server energy consumption state and a state stored in the knowledge base
	Negative selection	Generates and improves a set of predefined detectors
	Clonal selection, Affinity maturation	Generates, improves and selects the optimization actions

An *antigen* represents the current energy consumption state of a server. The energy consumption state is indirectly given by the workload and resource usage snapshot in a specific timestamp. We formally define an antigen as:

$$A_t = (SRL_t, PRL_t) \quad (1)$$

In (1), SRL_t is the list of current resources usage for the server in the moment of time t (see also (2)) and PRL represents the list of processes requests for server resources (see also (3)).

$$SRL_t = \{levelCPU_t, levelHDD_t, levelMEM_t\} \quad (2)$$

We have modeled in our antigen only those server resources on which Dynamic Power Management actions (i.e.

modify the resource performance state to save energy) can apply (Hard-disk, Memory and Processor).

$$PRL_t = \{levelCPU_t, levelHDD_t, levelMEM_t\} \quad (3)$$

We define a *self-antigen* as an energy consumption state in which no optimization actions are required. On the other hand, a non-self antigen represents a non-optimal energy consumption server state in which optimization actions have to be generated, selected and enforced. The self/non-self discrimination of an antigen part of the negative selection process is performed by comparing it to two predefined optimal antigens: the *idle antigen* that represents the optimal idle state of the system and the *active antigen* that represents the optimal active state of the system. The levels of the active optimal antigen are considered as *levelCPU* = 7, *levelMEM* = 7, *levelHDD* = 2, while the levels of the idle optimal antigen are considered as *levelCPU* = 1, *levelMEM* = 1 and *levelHDD* = 0. The affinity *a*, between current antigen / idle antigen and the afinity between current antigen / active antigen is determined as shown in (4) (for current antigen / idle antigen).

$$a(SRL_t, SRL_{idle}) = \sum_{levelR_S \in SRL_t, levelR_{idle} \in SRL_{idle}} |levelR_S - levelR_{idle}| \quad (4)$$

If one of these affinities is below a certain threshold, *T*, then the antigen is self, otherwise it is non-self:

$$\begin{cases} a(SRL_t, SRL_{idle}) < T \,||\, a(SRL_t, SRL_{active}) < T \Rightarrow self \\ otherwise \Rightarrow non-self \end{cases} \quad (5)$$

A detector matches the structure of an antigen and represents a server energy consumption state for which the optimization actions are already known (relation (6)). The detector is used to classify the antigens in self and non-self.

$$D = (SRL_D, PRL_D) \quad (6)$$

The effector is a structure which contains the optimization actions that need to be executed for a non-self antigen (relation (7)). It is described by the sequence of actions to be taken for reaching an optimal state.

$$E = \{a_1, a_2, ..., a_n\} \quad (7)$$

The execution of an optimization action transitions the server from the energy state of time *t* into a *t+1* time state (relation (8)). We have modeled and used only DPM actions such as modifying the frequency of the processor, putting the hard disk to sleep or on stand-by .

$$a : SRL_t \rightarrow SRL_{t+1} \quad (8)$$

A detector and the optimization actions associated form an immune cell (see (9)) which is stored in an immune memory knowledge base (see (10)).

$$IC = (D, E) \quad (9)$$

$$IM = \{IC_1, IC_2, ..., IC_n\} \quad (10)$$

The immune memory is limited in size and cannot keep every immune cell ever produced. The immune cells are replaced according to how much and how recently they were used. The usefulness of each cell is computed in the following way:

$$U(IC) = \alpha Times\_used + \beta Time\_rank \quad (11)$$

where *Times_used* represents how many times the immune cell *IC* was used so far and *Time_rank* represents how recently the immune cell *IC* was used. The constants $\alpha, \beta \in [0,1]$ give the relative importance of each of the two components.

In the energy optimization problem we use the behavior of the adaptive immune system for the situation in which a new server energy consumption state is encountered. In this case, the first step is to search the immune memory knowledge base to determine if a previous similar situation exists by measuring the affinity between the knowledge base immune cells (the detector part) and the current antigen (server consuming state). We compute the affinity between a detector and an antigen using Manhattan distance (relation (12)) between their *SRLs* (resources of the system) and *PRLs* (requests of the processes).

$$a(A, D) = \sum_{levelR_A \in SRL_A, levelR_D \in SRL_D} |levelR_A - levelR_D| + \sum_{levelR_A \in PRL_A, levelR_D \in PRL_D} |levelR_A - levelR_D| \quad (12)$$

The immune celll efector execution utility for the curent antigen can be determined using the current antigen, the predicted antigen and the predefined optimal antigen (for idle and active server states).

The predicted antigen over *x* time units is generated using the the Holt-Winters Method [23] for trend forecasting and prediction. We predict the server resources required values at the moement *t+x* taking into acount the curent antigen and the incoming workload.

$$A_{t+x} = (SRL_t, PRL_{t+x}) \quad (13)$$

$$\text{where } PRL_{t+x} = \{levelR_{t+x}, R \in \{CPU, HDD, MEM\}\} \quad (14)$$

To predict the next requirement for each server resources a liniar regresion approach is used. We use three types of components in prediction (relation (15)): the estimate requirements value (*ER*), the workload trend (*TR*) and the server resources maximum workload (to prevent the prediction of requirements higher than the server maximum workload).

$$levelR_{t+x} = \min\{ER_t + xTR_t, \max(levelR)\}$$

$$ER_t = \alpha_0(ER_{t-1} + TR_{t-1}) + (1-\alpha_0)levelR_t \quad (15)$$

$$TR_t = \alpha_1 TR_{t-1} + (1-\alpha_1)(ER_t - ER_{t-1})$$

The constants $\alpha_0, \alpha_1 \in [0,1]$ have the following meaning: if α_0 is smaller, then more weight is given to recent levels and less weight is given to older levels. Similarly, α_1 controls the importance given to the current or past trends.

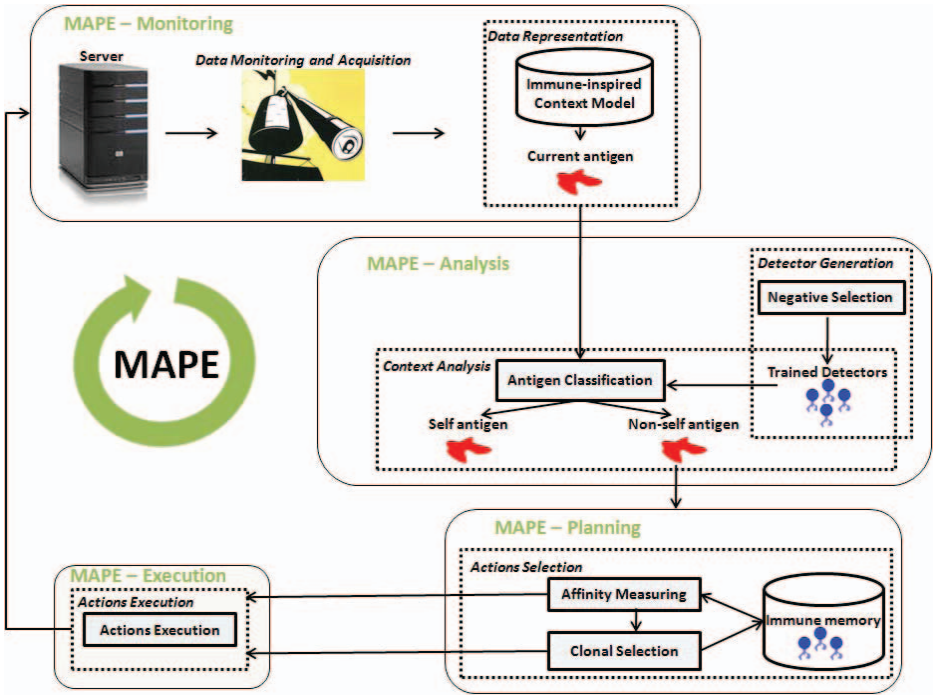

Figure 1. The biologically-inspired optimization technique MAPE phases

III. THE BIO-INSPIRED ENERGY OPTIMIZATION TECHNIQUE

The proposed biologically-inspired optimization technique has two main stages: an *initialization stage* and a *self-optimization stage*. Within the *initialization stage* the server is monitored for a specific period of time to collect energy consumption historical raw data with the goal of identifying associations between the server energy consumption states and the appropriate optimization actions. During this stage an initial immune memory knowledge base is built using a negative selection process (see Fig. 2). The immune memory knowledge base is enhanced and refined in the self-optimization stage.

1. **Algorithm NEGATIVE-SELECTION**
2. **Input:** AR – set of antigens; ; n – population size; c – cloning rate; m – mutation step; T – affinity threshold
3. **Output:** IM – immune memory
4. **Comments**: A – antigen; IS-SELF – detects if an antigen is self/ non-self; D – detector; SRL_A/PRL_A –system resource levels/process request levels of antigen A; E – effector; $SRL_{optimal}$ – optimal system resource levels; IM – immune memory
5. **begin**
6. **for** $i = 1$ **to** $|AR|$ **do**
7. $A = AR[i]$
8. **if** (IS-SELF(A)) **then**
9. $D = (SRL_A, PRL_A)$
10. E = CLONAL-SELECTION(D, n, c, m, T)
11. $IM = IM \cup \{(D,E)\}$
12. **end if**
13. **end for**
14. **end**

Figure 2. Negative selection algorithm

In the following sub-sections we focus on the bio-inspired optimization technique, the *self-optimization stage*, presenting its MAPE phases and highlighting the main algorithms used at each phase (Fig. 1).

A. Monitoring phase

In the monitoring phase, energy consumption server state snapshots are taken at regular time intervals and formally represented using a biologically-inspired antigen model. A snapshot includes information about the CPU, HDD and MEM utilization levels. Also in this phase the processes server resources requests are recorded. The algorithm used to create the antigen for the current energy consumption state is presented in Fig. 3.

1. **Algorithm CREATE-ANTIGEN**
2. **Input:** P – active process request levels; SRL – system resource levels
3. **Output:** A – Antigen
4. **Comments**: GET-MAX – returns the maximum CPU/HDD process request level; GET-SUM – computes the sum of the memory process request levels
5. **begin**
6. $levelCPU_t$ = GET-MAX(P_{CPU})
7. $levelMEM_t$ = GET-SUM(P_{MEM})
8. $levelHDD_t$ = GET-MAX(P_{HDD})
9. A = (SRL, ($levelCPU_t$, $levelHDD_t$, $levelMEM_t$))
10. **return** A
11. **end**

Figure 3. Antigen creation algorithm

B. Analysis phase

The analysis phase classifies the current antigen as self (optimal energy consumption state) or non-self (non-optimal

energy consumption state) using the set of detectors obtained in the initialization phase. The self/non-self discrimination (see Fig. 4) is performed by evaluating the affinity (COMPUTE-AFFINITY) between the current antigen and an idle/active antigen using relation (4).

```
1.  Algorithm SELF-NON-SELF-DISCRIMINATION
2.  Input: A – current antigen; A0 – idle antigen; AA – active antigen; T–
    threshold
3.  Output: self / non-self antigen
4.  Comments: a1, a2 – the affinity between the current antigen and the
    idle/active antigen; T – affinity threshold
5.  begin
6.     a1 = COMPUTE-AFINITY(A, A0)
7.     a2 = COMPUTE-AFINITY (A, AA)
8.     if (a1 < T) or (a2 < T) then return  self
9.     return non-self
10. end
```

Figure 4. Self / non-self discrimination algorithm

C. Planning phase

The planning phase determines the actions that need to be taken to bring the server in an optimal energy consumption state. The planning has the following steps.

Step1. The immune memory knowledge base is checked to determine if similar server energy consumption states have been previously encountered. Relation (12) is used to measure the affinity between the knowledge base immune cells (the detector part) and the current antigen (server consuming state). If a similar server energy consumption states is found the associated optimization actions are executed.

Step2. If no similar server energy consumption state is found then new optimization actions need to be generated.

Step3. Use a clonal selection-based approach on the memory cells with the highest affinity to the current energy situation (SELECT-HIGHEST-AFFINITY-CELLS) to generate the best sequence of optimization actions (see Fig. 5).

```
1.  Algorithm CLONAL-SELECTION
2.  Input: D – detector to clone; n – population size; c – cloning rate;
        m – mutation step; T – affinity threshold
3.  Output: E – new effector
4.  Comments: RH – set of effectors with the highest affinity regarding
    the current antigen, the predicted antigen and the optimal system
    resource levels; CLONE – clones the elements in RH c times; RM - set
    of mutated RH elements; m – mutation rate; RR – set of randomly
    generated effectors; SELECT-TOP – selects the best effectors;
    SELECT-BEST – selects the best effector E; UTILITY – computes
    the utility of an effector; T – utility threshold
5.  begin
6.     RH ← SELECT-HIGHEST-AFFINITY-CELLS()
7.     repeat
8.        RH = CLONE(RH,c)
9.        RM = MUTATE(RH, m)
10.       RR = GENERATE-RANDOM()
11.       RH = SELECT-TOP(RH ∪ RM ∪ RR)
12.       E = SELECT-BEST(RH)
13.    until (UTILITY(E, Acurrent, Apredicted, SRLoptimal) < T) or
                                                  (cutoff reached)
14. return E
15. end
```

Figure 5. Clonal selection algorithm

The high affinity memory cells are mutated using a reinforcement learning-based strategy (see Fig. 6).

```
1.  Algorithm MUTATE
2.  Input: RH - set of effectors to mutate, m – mutation step
3.  Output: RM - mutated set of effectors
4.  Comments: E – effector; GET-NR-RESOURCES – returns the
    number of monitored system resources; MUTATE-RESOURCE -
    modifies the value of the resource with the index j with + or – the
    mutation step m; EPLUS/EMINUS – the mutated effector; UTILITY–
    computes the utility of an effector
5.  begin
6.     RM = ∅
7.     for i = 1 to |RH |do
8.       E = RH[i]
9.       for j =1 to GET-NR-RESOURCES() do
10.        EPLUS = MUTATE-RESOURCE(E, j, m,+)
11.        if (UTILITY(EPLUS) > UTILITY(E)) then E = EPLUS
12.        EMINUS = MUTATE-RESOURCE(E, j, m,-)
13.        if (UTILITY(EMINUS) > UTILITY(E)) then E = EMINUS
14.      end for
15.      RM= RM ∪ {E}
16.    end for
17.    return RM
18. end
```

Figure 6. Mutation algorithm

D. Execution phase

In this phase the optimization actions generated in the planning phase are enforced on the server and a new immune cell is created and stored in the immune memory (see Fig. 7).

```
1.  Algorithm EXECUTE-OPTIMIZATION-ACTION
2.  Input: Acurrent – the current antigen, Ecurrent – the efector
        determined in the planning phase, IM – current version of
        the immune memory
3.  Output: server energy optimal state
4.  Comments: EXECUTE-ACTION – executes the actions encoded in
    an effector; IC – immune cell; IM – immune memory
5.  begin
6.     EXECUTE-ACTION(Ecurrent)
7.     IC = (Acurrent, Ecurrent)
8.     IM = IM ∪ { (D,E) }
9.  end
```

Figure 7. Execution phase algorithm

IV. Experimental Results

We have tested our solution on a system with a dual core CPU, two hard disks and a memory. The technical characteristics are summarized in Table 2.

TABLE II. Hardware Resources

CPU	Intel Core 2 Duo (2.4 GHz)
HDD0	500 GB, 7200 rpm
HDD1	160 GB, 7200 rpm
Memory	4 GB

The software resources involved in the testing of the self-optimizing algorithm consist of: (*i*) *Ubuntu 10.04, 2.6.32-21-generic kernel* as the server operating system, (*ii*) *Hyperic Sigar* [26], *cpufreq-info*, *hdparm* for monitoring the server components, (*iii*) *cpufreq-set*, *hdparm*, *sysctl* to enforce the

optimization actions and (*iv*) the *stress* program to generate the testing workload (see Fig. 8). The threshold T is set to 1.0, and the prediction constants are set to $\alpha_0 = \alpha_1 = 0.5$.

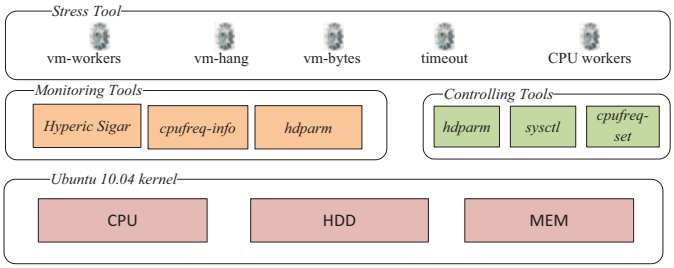

Figure 8. The self-optimizing software resources stack

For testing purposes, we have generated a number of resource-intensive tasks, along with their process requirements using the *stress* tool. The stress tool uses the following parameters to generate the proper workload:

- CPU workers – how many workers to launch running on *sqrt()*. More workers imply more CPU load.
- Vm-workers – how many workers to launch running on *malloc()/free()*. We only launch one, as we need to occupy memory without affecting the CPU.
- vm-bytes – how many bytes of memory to occupy during *malloc()*.
- vm-hang – wait period before freeing memory (in seconds).
- timeout – how long to stress the system (in seconds).

To wake the HDDs, we simply periodically create/delete a text file. A HDD is awaken if there is any attempt to access it.

During the initialization stage, the system is monitored and most representative antigens are selected. Detectors and effectors are created from these antigens, which form the basis of the innate immune system. Negative selection is applied to each detector, in order to test if it is auto-reactive. After completing the initialization stage, the optimization stage is started. The optimization stage uses a 2 seconds time interval to run all its MAPE phases.

Fig. 9 presents the adaptation actions taken by the self-optimizing algorithm to distribute the incoming workload in an energy efficient manner. It can be noticed that by correctly predicting the next antigen, the CPU, HDD and MEM states closely follow the incoming workload. In this way, the following energy inefficient situations are avoided: (*i*) putting the CPU/HDD/MEM in a low power state when there is a high workload in the near future and (*ii*) putting the CPU/HDD/MEM in a high power state for a short peak workload (over provisioned resources problem is avoided).

V. CONCLUSIONS

This paper presents an immune-inspired model and technique for optimizing the server energy consumption. The proposed technique is able to detect non-optimal server energy consumption states and to take the appropriate actions for bringing the server into an optimal energy consumption state. The results are promising, showing that the self-optimization technique is able to correctly predict the server CPU, MEM and HDD workload values. By executing dynamic power management actions, the server computing resources energy states closely follow the incoming workload.

For future work we intend to determine the server energy efficiency by means of a power meter and by comparing the server energy consumption when the server is managed using our self-optimization algorithm and when the default operating system management is applied.

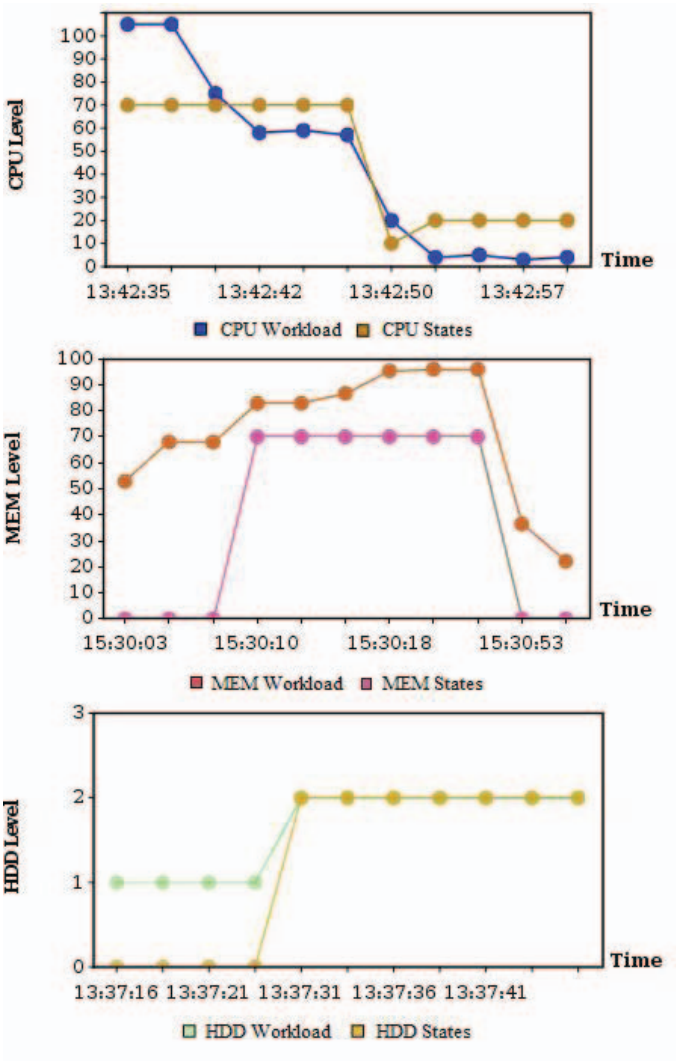

Figure 9. Server resources evolution over time

ACKNOWLEDGMENT

The work has been done in the context of the EU FP7 GAMES project [27].

REFERENCES

[1] M. G. Hinchey, R. Sterritt, "99% (Biological) Inspiration", Proceedings of the Fourth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems, pp. 187-195, 2007.

[2] A. Barroso, U. Hölzle, "The Case for Energy-Proportional Computing", IEEE Computer, vol. 40, 2007.

[3] J. Stanley, K. G. Brill, “Four metrics define data center greenness”, Uptime Institute, Whitepaper, 2007.

[4] S. Forrest, A. S. Perelson, L. Allen, R. Cherukuri, “Self-nonself discrimination in a Computer”, Proceedings of the IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, 1994.

[5] S. A Hofmeyr, S. Forrest, “Architecture for an Artificial Immune System”, Evolutionary Computation 8(4): 443 - 473. 2000.

[6] L. N. De Castro, and F. J. Von Zuben, “The clonal selection algorithm with engineering applications”, Proceedings of GECCO, Las Vegas, pp. 36–39, 2000.

[7] L. Castro, F. von Zuben, “Learning and Optimization using the Clonal Selection Principle”, Proceedings of the IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems, vol. 6, n. 3, pp. 239-251, 2002.

[8] L.N. Castro, J. Timmis, “Artificial Immune Systems: A Novel Paradigm to Pattern Recognition”, in Artificial Neural Networks in Pattern Recognition, J. M. Corchado, L. Alonso, and C. Fyfe (eds.), SOCO-2002, University of Paisley, UK, pp. 67-84, 2002.

[9] C. B. Pop, V. R. Chifu, I. Salomie, M. Dinsoreanu, I. Vartic, M. Vlad, “Immune-inspired Web Service Composition Framework”, Proceedings of the 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara (Romania), September 2009, ISBN: 978-0-7695-3964-5, pp.376-383, 2009.

[10] M. Dorigo, M. Birattari, T. Stützle, “Ant Colony Optimization - Artificial Ants as a Computational Intelligence Technique”, IEEE Computational Intelligence Magazine, 2006.

[11] J. Kennedy and R. Eberhart, ”Particle swarm optimization”, Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ. pp. 1942-1948, 1995.

[12] W. J. Gutjahr, “A converging ACO algorithm for stochastic combinatorial optimization,” in Proc. SAGA 2003, ser. LNCS, A. Albrecht and T. Steinh¨ofl, Eds., vol. 2827. Berlin, Germany: Springer Verlag, pp. 10–25, 2003.

[13] G. Di Caro, “Ant colony optimization and its application to adaptive routing in telecommunication networks,” Ph.D. dissertation, Universit´e Libre de Bruxelles, Brussels, Belgium, 2004.

[14] P. Boonma, J. Suzuki, "Biologically-inspired Adaptive Power Management for Wireless Sensor Networks," In G. Aggelou (ed.), Handbook of Wireless Mesh & Sensor Networking, Chapter 3.4.8, pp. 190 - 202, McGraw-Hill, ISBN 978-0071482561, 2008.

[15] M. Mamei, F. Zambonelli, “Pervasive pheromone-based interaction with RFID tags”, ACM Transactions on Autonomic Adaptation Systems, Vol 2 (2), Article 4, June 2007.

[16] J. Handl, J. Knowles, M. Dorigo, "Ant-Based Clustering and Topographic Mapping", Artificial Life 12(1), 2006.

[17] X. Li, “Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization”, Proceedings of Genetic and Evolutionary Computation Conference, pp. 105–116, 2004.

[18] A. Abraham , S. Das, S. Roy, “Swarm Intelligence Algorithms for Data Clustering”, In Soft Computing for Knowledge Discovery and Data Mining, Springer Verlag, Germany, ISBN 978-0-387-69934-9, pp. 279-313, 2007.

[19] B. W. Verdaasdonk, H. F. J. M. Koopman, F. C. T. van der Helm, "Energy efficient walking with central pattern generators: from passive dynamic walking to biologically inspired control", Springer-Verlag, Biological Cybernetics, Volume 101 , Issue 1 (August 2009), Pages: 49-61, 2009, ISSN:0340-1200.

[20] P. Champrasert, J. Suzuki, "SymbioticSphere: A Biologically-Inspired Autonomic Architecture for Self-Adaptive and Self-Healing Server Farms" Proceedings of the 2nd IEEE International Workshop on Autonomic Communications and Computing (ACC), June 2006.

[21] C. Lee, H. Wada, J. Suzuki, “Towards a Biologically-inspired Architecture for Self-regulatory and Evolvable Network Applications”, In F. Dressler and I. Carreras (eds.) Advances in Biologically Inspired Information Systems: Models, Methods and Tools, Chapter 2, pp. 21 - 45, Springer, ISBN 978-3540726920, August 2007.

[22] D. Floreano, C. Mattiussi. “Bio-Inspired Artificial Intelligence – Theories, Methods, and Technologies”. MIT Press, 2008.

[23] Tikunov, D. Nishimura T., "Traffic prediction for mobile network using Holt-Winter’s exponential smoothing", SoftCOM 2007, pp. 1 - 5, ISBN: 978-953-6114-95-5, 2007.

[24] L. N. Castro, "Fundamentals of Natural Computing – Basic Concepts, Algorithms and Applications", Chapman & Hall/CRC, ISBN: 1-58488-643-9, 2006.

[25] S. A. Hofmeyr, “An Interpretative Introduction to the Immune System”, In “Design Principles for the Immune System and other Distributed Autonomous Systems”, Oxford University Press, Eds, I. Cohen and L. Segel. 2001.

[26] Hyperic Sigar, http://www.hyperic.com/products/sigar

[27] GAMES Research Project, http://www.green-datacenters.eu/