Dynamic Frequency Scaling Algorithms for Improving the CPU's Energy Efficiency

Ionut Anghel, Tudor Cioara, Ioan Salomie, Georgiana Copil, Daniel Moldovan, Cristina Pop
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
{ionut.anghel, tudor.cioara, ioan.salomie, georgiana.copil, daniel.moldovan, cristina.pop}@cs.utcluj.ro

Abstract—**This paper approaches the problem of improving the service center servers CPU's energy efficiency by executing dynamic frequency scaling actions and performing tradeoffs between CPU's computational performance and its power consumption. Two different algorithms are designed and implemented: an immune inspired algorithm and a fuzzy logic based algorithm. The immune inspired algorithm uses the human antigen as a model to represent the server power / performance state. Using a set of detectors the antigens are classified as self for optimal power consumption state or non-self for non-optimal power consumption state. For the non-self antigens a biologically inspired clonal selection approach is used to determine the actions that need to be executed to bring the server's CPU in an optimal power consumption state. The fuzzy logic based algorithm adaptively changes the processor performance states to the incoming workload. The algorithm also filters workload spikes because frequent p-states transition costs can outweigh the benefit of adaptation.**

Keywords-dynamic frequency scaling; CPU energy efficiency; fuzzy logic; immune inspired.

I. Introduction and Related Work

The past decade in the field of computing has been marked by the advent of web-based applications and online services, which translated into an exponential growth of the underlying service center infrastructure. As the price and demand for energy continue to rise, service center power consumption must be considered as a first-class resource that is carefully controlled along with performance.

The GAMES (Green Active Management of Energy in IT Service Centers) [1] research project aims at developing a set of innovative methodologies, metrics, services and tools for the active management of energy efficiency of IT service centers. In GAMES the service center energy consumption problem is tackled both at service center and local server level.

At service center level resource consolidation and virtualization techniques are employed to combine workloads that are executed on different machines (servers) so that an optimal number of computing resources are always used [2]. At server level power management techniques are used to optimize the energy consumption of the most power-hungry components within a server like CPU, HDD, or RAM.

In this paper we propose the development of *human immune system inspired* and *fuzzy logic based* Dynamic Frequency Scaling (DFS) algorithms for improving the servers CPU energy consumption while trying to preserve the performance levels required by the servers running tasks. The algorithms use virtualization as an abstraction level for a uniform management of the server running tasks, without worrying about application dependencies and low-level details. The virtualized tasks are considered to be annotated with Quality-of-Service (QoS) requirements representing the tasks' performance request levels. Both the algorithms are based on the Advanced Configuration and Power Interface (ACPI) [3], an open industry standard which allows an operating system to directly control the power-saving aspects of its underlying hardware. ACPI allows for dynamically changing the CPU power states (p-states) and implicitly its frequency at run-time so that the CPU energy consumption is minimized. The values of p-states depend on the CPU type, but usually $P_{state-0}$ is always the highest-performance state (highest frequency, highest power/energy consumption), while $P_{state-1}$ to $P_{state-n}$ are successive lower-performance states.

The *human inspired DFS technique* is similar with an artificial immune system associated to a server and is able to detect non-optimal server power consumption states (similar to biological pathogens) and to take the appropriate actions required to bring the server into an optimal state (similar to the biological immune response). This DFS 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 power consumption historical data with the goal of identifying associations between the server power consumption states and the appropriate DFS optimization actions. Such an association is represented as an artificial immune cell composed from a detector (server power consumption state) and an effector (the DFS optimization actions). The self-optimization stage is designed as a control feedback loop with the following MAPE phases: Monitoring, Analyzing, Planning and Execution. In the monitoring phase, power 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 power consumption state) or non-self (non-optimal power consumption state) using the set of detectors obtained in the initialization stage. The planning phase determines the DFS actions (effectors) that need to be taken to bring the server in an optimal power 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.

The *Fuzzy logic DFS technique* is based on defining and using a fuzzy function capable of filtering the situations in which the workload fluctuates for short periods of time because the transition cost (in terms of consumed energy) can outweigh the benefit of the adaptation.

The state of the art approaches regarding the CPU DFS concern about application performance and infrequent but inevitable workload peaks and lead to conservative approaches [4]. Khargharia proposes a theoretical methodology and an experimental framework for autonomic power and performance management of high-performance server platforms [5]. Similarly, in [6] the performance-power management problem is decomposed into smaller sub-problems implemented locally by individual controllers within each hardware component. The main problem with DFS techniques is that power states changes determine performance, and sometimes energy penalties [7]. Algorithms have been developed for DFS by considering the power / performance characteristics of main hardware devices, mainly the processor and the hard disk drive and, to some extent, the system memory (mostly hardware controllers) [8]. Bircher and John [9] perform an ample analysis of power management on recent multi-core processors using the features provided by regular operating systems. The use of virtualization together with DFS techniques raises serious challenges due the lack of power metering for virtual machines [10]. Unlike physical servers that provide in-hardware or outlet level power measurement functions, the power required by a virtual machine when hosted on a certain hardware configuration is difficult to be estimated [11].

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 [12]. The adoption of biological principles such as decentralization, autonomy, natural selection or symbiosis in the process of designing and building services on top of server farms is a novel research direction [13]. 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 such as pheromone emission, replication, migration, energy exchange. Each of these actions has an associated energetic cost.

The rest of this paper is organized as follows: Section II details the human immune system inspired DFS algorithm, Section III presents the fuzzy logic approach, Section IV shows the experimental results comparing the algorithms effectiveness in saving energy, while Section V concludes the paper.

II. Immune Inspired DFS

The following sub-sections present the human immune system related concepts and processes together with their mapping and usage for optimizing the CPU energy consumption using DFS actions.

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). With this aim, the immune system provides a set of countermeasures organized in defenses layers able to detect and eliminate pathogens [15]. 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. However, these two defenses layers are unspecific (they do not make any distinction between pathogens) and do not change during the lifetime of the living organism. In time, pathogens adapt themselves and in consequence the immune system will no longer have the suitable detector and effector cells to use. This is the moment in which the third defenses 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.

The generated cells pass through several phases until they are considered as part of the immune system. 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.

B. From Biological Immune Systems to CPU DFS

To optimize the CPU energy consumption using DFS actions the adaptive immune system's defense layer specific concepts and processes are applied (see Table 1 for a mapping of the human immune system concepts and processes to the server energy optimization problem).

The immune inspired DFS algorithm has two main stages: an *initialization stage (offline)* and a *self-optimization stage (run-time)*.

Within the *initialization stage* (or training stage) the CPU is monitored for a specific period of time, workload/p-state data is collected and the corresponding DFS actions are identified. The collected workload/p-state data forms the detector part of an immune cell ($A^{Detector} = (W^{Detector}, P_{state-i})$) while the corresponding DFS action ($Action_{DFS}$) form the effector part of the immune cell. The immune cells are stored in a knowledge base which has a similar role with the immune memory.

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

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

Within the *self-optimization stage* the appropriated DFS actions that need to be executed as a result of CPU workload variations are dynamically determined at run-time. The self-optimization stage adaptive immune system inspired processes are organized according to the self adaptive system MAPE phases: Monitoring, Analysis, Planning and Execution.

In the *monitoring phase*, CPU workload and p-state snapshots are taken at regular time intervals and formally represented using a biologically inspired antigen model $A_t = (W_t, P_{state-i})$. A snapshot includes data about the CPU utilization level and the corresponding timestamp. Also in this phase the server deployed virtualized tasks requests for CPU are recorded.

In the *analysis phase* the current antigen is classified as self (optimal power consumption state) or non-self (non-optimal power consumption state) using a set of predefined detectors. For each CPU p-state its technical datasheet defines the minimum ($W^{MIN}_{P_{stat\ e-i}}$) and maximum ($W^{MAX}_{P_{state\ -i}}$) workload values for which the CPU power consumption is optimal. A CPU optimal power consumption state represents a CPU snapshot for which its workload value is between $W^{MIN}_{P_{state\ -i}}$ and $W^{MAX}_{P_{sta\ te-i}}$. A detector is an antigen (A^{optim}) which represents the CPU optimal workload value for a certain p-state. The self/non-self discrimination is performed by evaluating the affinity between the current antigen (A_t) and the minimum and maximum workload values corresponding to the current antigen p-state:

$$Affinity^{self}(A_t, A^{optim}) = \frac{2*W_t}{W^{MIN}_{P_{state\ -i}} + W^{MAX}_{P_{state\ -i}}} \qquad (1)$$

If the calculated affinity is in the $[1 - T_{self}, 1 + T_{self}]$ interval, the current antigen represents an power efficient state in which the CPU p-state matches the current workload value. Otherwise the planning phase must be triggered to decide on the DFS actions to be executed to bring the CPU in an power efficient state. The threshold T_{self} is empirically selected by means of experiments. By varying the T_{self} the threshold the performance/energy consumption of the CPU is directly influenced as follows: a smaller value for T_{self} implies that the emphasis will be on CPU's performance while a higher T_{self} implies that the emphasis will be on CPU's energy savings.

In the *planning phase* the DFS actions that need to be taken to bring the server in an optimal power consumption state are determined. The planning phase may decide on three different types of DFS actions: increase the CPU p-state or decrease the CPU p-state. In the planning phase the initial phase constructed knowledge base is queried to determine if similar CPU workload/p-state values have been previously encountered. Relation (2) is used to measure the similarity between the knowledge base immune cells (the detector part) and the current antigen:

$$\varphi(A_t, A^{Detector}) = W_t - W^{Detector} \qquad (2)$$

If the calculated similarity is higher than a threshold the immune cell effector DFS action is executed. Otherwise new DFS actions that need to be executed are selected using a clonal selection algorithm (see Fig. 1).

```
1.  Algorithm CLONAL-SELECTION
2.  Input: $IM$ − $Imunne\ cell\ with\ the\ highest\ afinity$
3.             $to\ the\ current\ antigen$
4.         $C_{Nb}$ − $the\ no.clones\ to\ be\ generated$
5.         $M = \{p\uparrow, p\downarrow\}$ $the\ mutation\ step$
6.         $T_{self}$ − $affinity\ threshold$
7.  Output: $Action_{DFS}$ − $the\ DFS\ action$
8.  Comments: $IM_r$ − $set\ of\ randomly\ generated\ effectors$
9.                 $usedfor\ diversity$
10.
11. begin
12.   $IM \leftarrow SELECT - SIMILAR - CELLS\ ()$
13.   repeat
14.     $IM_C = CLONE(IM, C_{Nb})$
15.     $IM_m = MUTATE(IM_C, M)$
16.     $IM_r$ = GENERATE-RANDOM()
17.     $E$ = SELECT-TOP($IM_C, IM_m, IM_r$)
18.   until $1 - T_{self} \le Affinity^{analysis}(E, A^{optim}) \le 1 + T_{self}$
19.                                          or $(cutoff\ reached)$
20.   return $E.\ Action_{DFS}$
21. end
```

Figure 1. The clonal selection algorithm

The first algorithm step is to select all the initial stage immune cells for which the calculated similarity value is lower than the defined threshold but close to it (see Fig. 1 CLONAL-SELECTION algorithm lines 12). The selected immune cells are cloned and each clone is being mutated using the following main mutations: $p\uparrow$ - scale up CPU DFS action and $p\downarrow$ scale down CPU DFS action (see Fig. 2 MUTATE lines 9-13). If the mutated immune cells represent an power efficient state the algorithm stops and the immune cell effector is selected for execution (see Fig. 1 CLONAL-SELECTION algorithm lines 18-20).

```
1.  Algorithm MUTATE
2.  Input: $IM_C$ – set of immune cells to be mutated
3.           $M = \{p\uparrow, p\downarrow\}$ the mutation step
4.  Output: $IM_m$ – mutated immune cells
5.  Comments: UTILITY – returns the immune cell
6.                          with the highest
7.  begin
8.     $IM_m = \emptyset$
9.     $Current_{IM} = SELECT - IM\ ()$
10.    $IM_m\uparrow = MUTATE - UP(Current_{IM}, p\uparrow)$
11.    $IM_m\downarrow = MUTATE - DOWN(Current_{IM}, p\downarrow)$
12.    $Current_{IM} = HIGH - UTILITY(Current_{IM}, IM_m\uparrow, IM_m\downarrow)$
13.    $IM_m = IM_m \cup \{Current_{IM}\}$
14.    return $IM_m$
15. end
```

Figure 2. Immune cell mutation algorithm

III. Fuzzy Logic DFS

The fuzzy logic DFS algorithm filters the situations in which the workload fluctuates for short periods of time because the transition cost (in terms of consumed energy) can outweigh the benefit of the adaptation. Therefore, if the CPU is currently in a low power state and the load exhibits an isolated spike, the CPU must not enter full mode, because even if there will be a delay in servicing the requests, it is tolerable in the context of the overall computational throughput.

Given the above considerations, we have decided to design and implement a processor power management algorithm based on fuzzy controller. The approach is somewhat similar to the one proposed in [16], however instead of using it to globally manage the service center sizing according to incoming workload, we have modified and upgraded it to control the processor's capacity. The advantage of fuzzy-logic is the ability to filter-out noise and to adapt progressively to changes.

The fuzzy controller has two input variables: the processor usage ratio (the current processor power state) and the workload represented by the instructions that the processor must execute. For each of the processor power states, we define three fuzzy sets, LOW, MEDIUM and HIGH, represented by f_l , f_m and f_h functions. After each time window, the fuzzy controller (represented by f_c function) evaluates the f_l, f_m and f_h functions for the current load and updates f_c value according to following equation:

$$f_c = \delta * f_{c-1} + \propto * f_h(cpuPowerState, cpuLoad) - \beta * f_l(cpuPowerState, cpuLoad) + \gamma * f_m(cpuPowerState, cpuLoad) \qquad (3)$$

Figure 3 presents the fuzzy logic DFS algorithm for processor. If the fuzzy controller equation evaluation (lines 9-12) reaches the values 0 or 1, the CPU is transitioned to an inferior (lines 17-19), respectively a superior power state (lines 13-16) and its f_c value is reset to the default value of 0.5 (lines 15 and 19).

```
1   Algorithm FUZZY-LOGIC-DFS
2   Input: serverCPU – the processor which is dynamic power managed
3   Output: CPU in an optimized power state
4
5   begin
6     set cpuPowerState to MAX_CPU_POWER_STATE
7     while TRUE
8       read cpuLoad for serverCPU
9       set fuzzyControl = DELTA * fuzzyControl +
10                          ALPHA * fh(cpuPowerState, cpuLoad) -
11                          BETA * fl(cpuPowerState, cpuLoad) +
12                          GAMMA * fm(cpuPowerState, cpuLoad)
13      if cpuPowerState < MAX_CPU_POWER_STATE && fuzzyControl >= 1 then
14        TransitionCPUtoHigherPState()
15        fuzzyControl = 0.5
16      endif
17      if cpuPowerState > MIN_CPU_POWER_STATE && fuzzyControl <= 0 then
18        TransitionToLowerPState()
19        fuzzyControl = 0.5
20      endif
21    endwhile
22  end
```

Figure 3. Fuzzy logic based DFS algorithm for processor

By varying α, β, γ and δ the trade-off between energy efficiency and performance degradation can be controlled: a large α or β value means that the load spikes will not be filtered, while small α and β induce delays that lead to performance degradation and thus increasing energy efficiency.

IV. Case Study and Results

To test and validate the proposed DFS algorithms we have used an IBM server having an Intel i7 processor of 3GHz and 6 GB of memory. The Intel i7 processor provides 14 power states (p-states), each state having different associated frequencies and implicitly different power consumption values (see Table 2).

TABLE II. The Processor P-States

P-State	Frequency (GHz)	Frequency (%)	Power Cons. (W=J/s)
0	2.93	100	95
1	2.79	95.22	83.41
2	2.66	90.78	72.87
3	2.53	86.34	63.33
4	2.39	81.56	54.75
5	2.26	77.13	47.06
6	2.13	72.69	40.24
7	2.00	68.25	34.21
8	1.86	63.48	28.94
9	1.73	59.04	24.36
10	1.60	54.60	20.44
11	1.46	49.82	17.13
12	1.33	45.39	14.36
13	1.20	40.95	12.09

For testing purposes, CentOS [18] operating system with 5.5 kernel version is installed as host operating system on the IBM server. To monitor and collect the CPU usage information cpufreq-utils [17] kernel utility was used. cpufreq-utils also allows for setting the CPU p-state and frequency at any desired value. On top of the described infrastructure reside the human inspired and fuzzy logic based DFS algorithms.

To evaluate the energy efficiency capabilities of the two DFS algorithms, tests have been run for two different scenarios: worst case and average case. The worst case scenario occurs for CPU–intensive workloads when the CPU is always used at its maximum frequency. The average case refers to usual CPU workloads, with medium intensity and frequent spikes, for which the CPU usage varies somewhere between 40 and 100 %. Using a power meter (ISO-TECH IPM3005 [20]) we have measured the IBM server instant power consumption (W=J/s) in both test cases. The IBM server energy consumption was determined as the total power consumed in the time period needed to execute the generated workload.

A. The Worst Case Scenario

To generate the worst case testing scenario highly intensive workloads the Super PI CPU benchmarking software [19] was used. The Super PI CPU benchmark is based on computing 2^n digits of π (n is maximum 25). We have randomly started several PI digits generators at different time intervals, thus obtaining the workload for which the results presented in the below figures were obtained.

The same generated workload is executed and the following CPU management algorithms are evaluated: (1) CentOS OnDemand CPU frequency governor, (2) immune inspired DFS algorithm and (3) fuzzy logic based DFS algorithm. For the second and third management algorithms the CentOS CPU frequency governor is deactivated.

As it can be seen in Fig. 4 for the same executed workload the average power consumed by the IBM server was about 143.5 W when the fuzzy logic based DFS algorithm was used, 147.1 W for the CentOS OnDemand CPU frequency governor and 105.6 W for the immune inspired DFS.

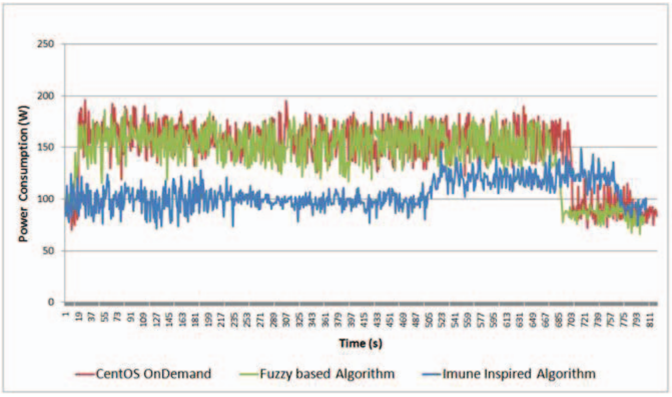

Figure 4. The server power consumption chart when executing the same CPU intensive workload with different management algorithms

The estimated energy consumption was about 116378 J (32.3 Wh) for the fuzzy logic algorithm, 119298 J (33.1 Wh) for CentOS OnDemand and 85641 J (23.8 Wh) for the imune inspired algorithm. This results in an energy efficiency increase of 35 % for the immune inspired DFS algorithm compared with CentOS OnDemand and fuzzy logic based DFS algorithms.

The high energy consumption of the CentOS OnDemand CPU frequency governor is justified by the fact that, regardless the workload oscillations, the CPU is over-provisioned and constantly held in P-0, the highest power consumption state (see Fig.5).

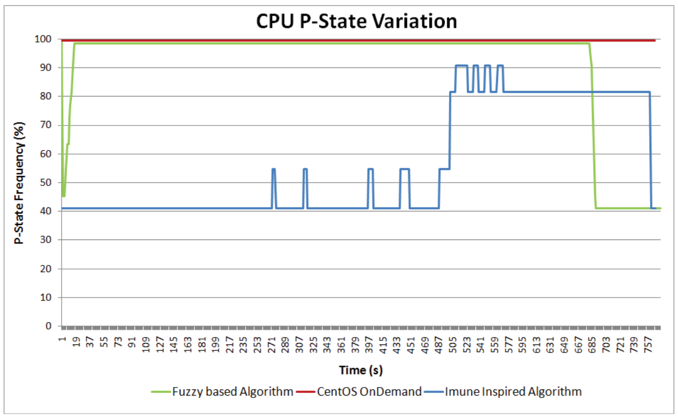

Figure 5. CPU Frequency and p-state variation when executing the same CPU intensive workload with different management algorithms

When the fuzzy logic based DFS is used as management algorithm, the CPU p-state oscillates frequently, depending on the workload intensity and its duration (see Fig. 5). The workload sensitivity and in consequence its capability to filter the workload spikes, leads to the fuzzy logic based DFS algorithm energy efficiency increase, compared with CentOS OnDemand CPU frequency governor. The small difference (2.5%) is caused by the fact that the workload is extremely intensive and the CPU is held at P-0 for performance reasons.

The immune inspired DFS algorithm takes advantage on the biological systems self-adaptivity and manages to avoid the CPU high frequency p-states even for very intensive workload spikes. This fact leads to low energy consumption but a higher performance penalty. We have estimated for the immune inspired DFS algorithm a performance penalty / degradation of 8.71% compared with the CentOS OnDemand, while for the fuzzy logic based DFS the performance degradation can be neglected. The higher performance penalty increases the workload execution time when immune inspired DFS was used with about 60 seconds.

B. The Average Case Scenario

To create the average case scenario a matrix multiplication program, which generates workload that moderately stresses the CPU was developed and used. The obtained workload was repeatedly executed and the three different CPU frequency management algorithms were independently activated.

Analyzing the power consumption chart from Fig. 6 it can be noticed that the best power consumption results (114,73 W) were obtained when the immune inspired DFS was used. For the execution of the same workload the IBM server consumes about 135,05 W when is managed by the fuzzy logic based DFS and 153.16 W for the CentOS OnDemand algorithm. The estimated energy consumption for the current test case was about 132484 J (36.8 Wh) for the fuzzy logic algorithm, 150249 J (41.7 Wh) for CentOS OnDemand and 112550 J (31.2 Wh) for the immune inspired algorithm.

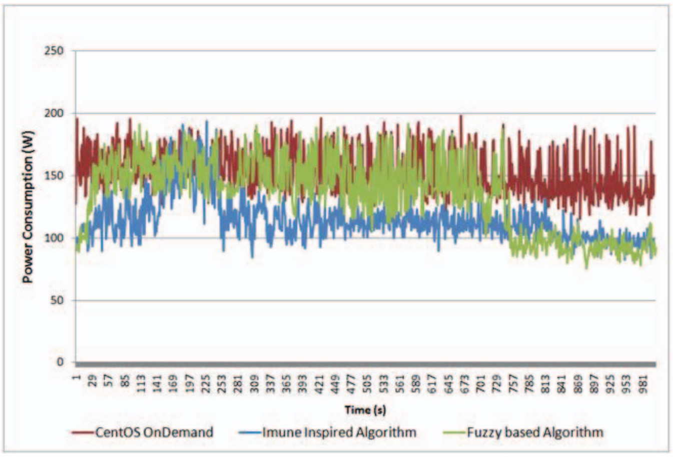

Figure 6. The server power consumption chart when executing the same CPU moderately intensive workload with different management algorithms

When the CentOS OnDemand CPU frequency governor is active, the CPU is almost constant for the entire period (in P-0, the highest power consumption state) (see Fig. 7).

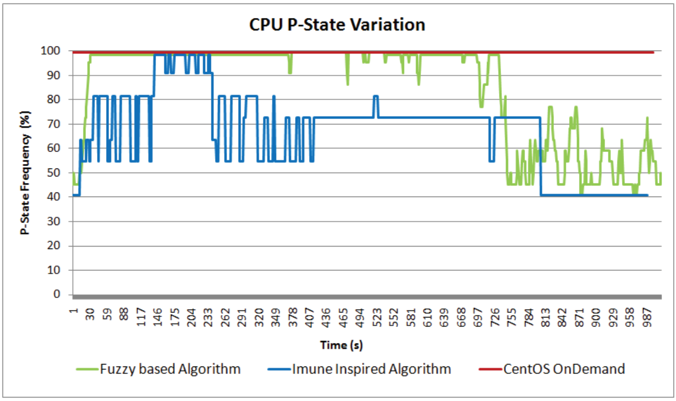

Figure 7. CPU frequency and p-state variation when executing the same CPU moderately intensive workload with different management algorithms

On the other hand, given the average intensity of the workload (about 40%-100%), the fuzzy logic based DFS keeps the CPU in P-0 for most of the time (see Fig. 7). Despite this, it achieves an energy efficiency increase of 13% (compared with CentOS OnDemand), without having any performance degradations.

The immune inspired DFS keeps the CPU in low energy consumption p-states thus enabling more power savings (33.6% compared with CentOS OnDemand). Due to the workload characteristics (moderate intensity), the management algorithms performance penalty is smaller compared to the high intensity workload case. For the immune inspired algorithm we have estimated a performance penalty of about 7.7% compared with the fuzzy logic based DFS and CentOS OnDemand frequency governor.

V. CONCLUSIONS

This paper proposes two types of CPU frequency management algorithms: an immune inspired algorithm and a fuzzy logic based algorithm. The algorithms perform a tradeoff between CPU's computational performance and its energy consumption by executing dynamic frequency scaling actions. The test case scenarios show that the optimal energy consumption CPU management algorithm is the immune inspired algorithm, while the worst energy consumption results were obtained for the CentOS OnDemand algorithm. This high energy consumption saving has as downside a moderate performance penalty.

ACKNOWLEDGMENT

This work has been supported by the European Commission within the GAMES project [1] funded under EU FP7.

REFERENCES

[1] GAMES FP7 Research Project, http://www.green-datacenters.eu/

[2] M. Uddin and A. Rahman, “Server Consolidation: An Approach to Make Data Centers Energy Efficient & Green“, International Journal of Scientific & Engineering Research, Volume 1, Issue 1, 2010.

[3] ACPI - Advanced Configuration and Power Interface, http://www.acpi.info

[4] J. Chase, and R. Doyle, “Balance of power: energy management for server clusters”, In Proceedings of the Eighth Workshop on Hot Topics in Operating Systems, ISBN 0-695-1040-X, 2001.

[5] B. Khargharia, S. Hariri, M. Yousif, “Autonomic power and performance management for computing systems”, Cluster Computing, Vol. 11 (2), 167 – 181, ISSN:1386-7857, 2008.

[6] N. Wang, N. Kandasamy, A. Guez, “Distributed Cooperative Control for Adaptive Performance Management”, IEEE Internet Computing, vol. 11 (1), 31-39, 2007.

[7] R. Minerick, V. Freeh, P. Kogge, “Dynamic Power Management using Feedback”, In: Proceedings of Workshop on Compilers and Operating Systems for Low Power, 2002.

[8] R. Gupta, S. Irani, S. Shukla, “Formal Methods for Dynamic Power Management”, In: IEEE/ACM international conference on Computer-aided design, 874, 2003.

[9] L. Bircher, L. John, “Analysis of Dynamic Power Management on Multi-CoreProcessors”, In: Proc. of the 22nd annual international conference on Supercomputing, 327-338, 2008.

[10] A. Kansal, F. Zhao, J. Liu, et al, “Virtual machine power metering and provisioning”, Proceedings of the 1st ACM symposium on Cloud computing, 39-50 , ISBN:978-1-4503-0036-0, 2010.

[11] J. Stoess, C. Lang, L. Bellosa, “Energy Management for Hypervisor-Based Virtual Machines”, USENIX Annual Technical Conference, 2007.

[12] 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.

[13] 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.

[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] D. Floreano, C. Mattiussi. "Bio-Inspired Artificial Intelligence". MIT Press, 2008.

[16] Z. Chen, Y. Zhu, M. Yuan, "Towards Self-Optimization in Utility Computing using Fuzzy Logic Controller", In proceedings of the IEEE International Conference on Service Operations and Logistics, and Informatics, Beijing, China, (August 10-12 2005), 1092-1095.

[17] cpufreq utility, http://www.kernel.org/pub/linux/utils/kernel/cpufreq/cpufrequtils.html.

[18] CentOS Enterprise Linux Distribution, http://www.centos.org/.

[19] Super Pi benchmark, http://www.super-computing.org/.

[20] ISO-TECH IPM3005, http://www.iso-techonline.com/products/iso-tech-electrical-installation-testers.html