An Autonomic Algorithm for Energy Efficiency in Service Centers

Ionut Anghel, Tudor Cioara, Ioan Salomie,
Georgiana Copil and Daniel Moldovan
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
{ionut.anghel, tudor.cioara, ioan.salomie}@cs.utcluj.ro

***Abstract*—This paper addresses the problem of run-time management of a service center energy efficiency by using a context aware self-adapting algorithm. The algorithm adapts the service center energy consumption to the incoming workload by considering service center predefined Green Performance Indicators (GPI) and Key Performance Indicators (KPI). The service center energy performance context is obtained from business applications scheduled to run on service center resources, from service center IT computing resources and from service center facilities. The self-adapting algorithm detects / analyzes service center's current workload and performance changes and decides on adaptation actions for minimizing the energy consumption. A reinforcement learning approach is used to decide the best sequence of actions to be executed to bring the service center resources as close as possible to a GPI and KPI compliant state.**

Keywords – autonomic computing; green computing; energy aware; self-adapting; service center

I. INTRODUCTION AND RELATED WORK

Over the last years the energy efficiency management of IT processes, systems and data centers has emerged as one of the most critical environmental challenges to be dealt with. Since computing demand and electricity price is continuously rising, reducing energy consumption of IT systems and data centers is expected to become a priority for the industry. The worldwide data centers CO_2 emissions are already equivalent to about half of the total airlines' CO_2 emissions, and are expected to increase in the next years [1]. Data centre electricity consumption accounts for almost 2% of the world electricity production and their overall carbon emissions [2]. Nowadays, the data centers are moving towards the service-center concept by adopting of a service-based model. In such a service center, the software is accessed as-a-service and the computational capacity is provided on demand to the clients. Based on these premises, Green Computing emerges as a new discipline and practice aiming at designing and using computing resources in an environmentally-aware way.

The GAMES (Green Active Management of Energy in IT Service Centers) [3] 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. The GAMES vision is to create a new generation of Green and Energy-aware IT service centers by designing energy efficient service centers one side and by managing at run-time the service center energy efficiency.

This paper addresses the problem of run-time managing the service center energy efficiency by using a context aware self-adapting algorithm for adjusting the energy consumption to the incoming workload. The context consists of the following categories of energy / performance data: (i) data related to business applications to be executed on the service center IT computing resources, (ii) data related to the workload and energy consumption of the service center IT computing resources and (iii) ambiental data related to the service center environment. The self-adapting algorithm implements the four phases of a MAPE control loop: **M**onitoring, **A**nalyzing, **P**lanning and **E**xecution phases. In the monitoring phase, the context data (service center energy performance data) is collected and represented in a programmatic manner by using our RAP (Resources, Actions, Policies) context model [4]. In the analyzing phase, the current service center energy performance context instance is evaluated to identify whether the predefined GPI and KPI indicators are fulfilled. In the planning phase a reinforcement learning approach is used to generate the best sequence of actions to be taken in the execution phase to bring the service center as close as possible to a GPI and KPI compliant state.

The proposed adaptation algorithm decides on two types of adaptation actions: consolidation actions and DPM (Dynamic Power Management) actions. The consolidation actions are executed for balancing the service center workload in an energy efficient manner. DPM actions identify the service center over-provisioned resources targeting to set them into low power states.

In the Software as a Service area, energy consumption has not been considered yet. Different QoS-aware techniques based on integer programming, genetic algorithms or heuristics have been proposed in the last years [7], [8], [9] with an energy-aware extension in [6]. A systematic analysis of cost issues in the design of hardware and network systems is presented in [10] while a cost-oriented design of data centers in which applications are

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

allocated with a fixed number of tiers for satisfying response time requirements is proposed in [11].

To reduce energy consumption, autonomic techniques have been developed to manage workload fluctuations [12] and to determine the optimum trade-offs between performance and energy costs, mainly by switching servers on and off. It has been shown that by implementing Dynamic Voltage Scaling (DVS) mechanisms (i.e., reduction of CPU frequency), significant energy reduction can be obtained [13], [14]. Self-* algorithms for ad-hoc networks, in which the connections between nodes are added / removed for energy efficiency purposes are proposed in [15], [16]. An agent based solution for power management in complex service centers comes from the IBM researchers R. Das and J. O. Kephart [17]. The authors use virtualization and load-balancing techniques for efficiently and energy aware management of service center's resources. The development of energy-aware computing systems is proposed in [5]. In [18], a self-adaptive solution for optimizing and balancing performance and energy consumption for RAID storage is proposed. This technique groups the available data into hot (frequently accessed data) and cold data. Hot data is kept on an active disk group while the cold data is kept in an inactive disk group which spins at low speed in order to save energy. Energy–aware provisioning and load dispatching algorithms that take into consideration the energy / performance models and user experience are designed for servers that host a large number of long-lived TCP connections [19]. An architecture which supports the design and programming of self-optimizing algorithms for data locality and access in distributed systems is considered in [20]. Based on application's run-time behavior, the data access and distribution is re-modeled with respect to a set of predefined energy performance objectives.

The rest of the paper is organized as follows: in Section II, the conceptual architecture and objectives of the GAMES project are briefly presented; Section III details our self-adapting algorithm; Section IV shows a case study and results while Section V concludes the paper and highlights the future work.

II. THE GAMES PROJECT

The GAMES project aims at developing an innovative framework of methodologies, algorithms, techniques, metrics, tools and services for energy-aware IT Service Centers (Green Service Centers). The main expected outcome will be to increase energy efficiency of IT Service Centers and reduce their global carbon footprint. The GAMES Green IT Service Center will use self-adaptation and context-awareness as a basis for energy-efficiency.

For run-time adaptation, the GAMES project combines context aware and autonomic computing techniques with control theory specific methods. To adjust and adapt the service center to the energy efficiency goals established in the design phase, two control loops are proposed: a local control loop associated to each service center server and a global control loop associated to the whole service center (see Fig. 1).

The local control loop is used to locally optimize the service center server specific energy consumption, without considering the entire service center state. The energy consumption optimization process of a single server is achieved by using DPM techniques to put the over-provisioned resources to low-power states. The resources remain in low-power state until there is a predictive workload that requires scaling-up the server computing capacity. The local loop should balance the server workload and its resource usage in an energy efficient way.

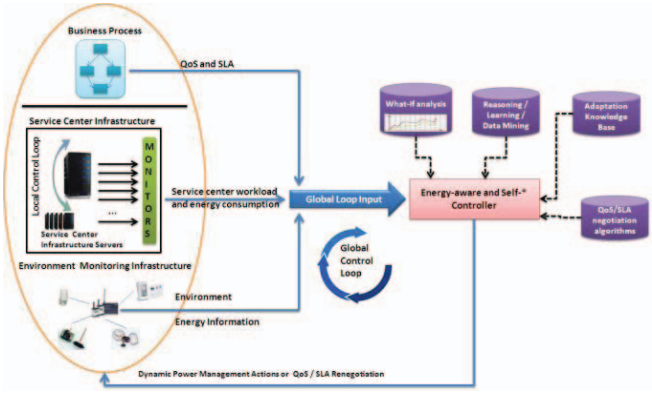


Figure 1. Run-time adaptation conceptual architecture

The global loop controller (Energy-aware and Self-* Controller in Fig. 1) uses the whole service center energy performance data for taking run-time adaptation decisions to enforce the predefined service center GPI and KPI indicators. The global loop proactively detects the service center over provisioning resources and identifies the most appropriate power and performance management strategies to adapt to the incoming workload. Usually, in a service center the computing resources are over-provisioned to be able to handle the peak load. The global loop exploits this feature to minimize the energy consumption by sensing the workload changes and using consolidation techniques to dynamically adjust the allocation of service center computing resources.

III. THE SELF-ADAPTING ALGORITHM

This section presents a general purpose self-adapting algorithm for context aware systems. This algorithm will be specialized and used to adapt the service center energy consumption to the incoming workload (see Section IV). The specialized version of the self-adapting algorithm implements the global control loop, of the Energy-Aware and Self-* Adaptivity Controller.

To define and formalize the self-adapting property of a context aware system we have used three main situation calculus concepts: *situations*, *conditions* and *action plans*. The *situation* concept also referred as system situation or

context situation, represents the state of a context aware system at a moment of time. The *conditions* guide and control the system execution. The *action plans* are executed by the context aware system to enforce the set of conditions for a specific situation. Using these concepts, we define the self-adapting property of a context aware system as a function, which associates an action plan *a* to each system context situation *s* that fails to fulfill the prescribed set of conditions.

```
1.  input:
          RAPmodel - the context model sets R,A,P
          Kb={(si,ai)} the set of previosly encountered
               context situations and their asociated actions
2.        TE - the entropy threashold
3.  output: a new adapted system state
4.
5.  procedure selfAdaptingAlgorithm(RAPmodel, Kb, TE)
6.  begin_MAPE
7.    //Monitoring Phase
8.     ctxData = gatherContextData();
9.     s = createContextSituation(RAPmodel, ctxData)
10.  //Analisys Phase
11.    ES = evaluateContextSituationEntropy(s, P)
12.  //Planning/Deciding Phase
13.    action = Ø
14.    if (ES > TE) then
15.      if ( (s,a) in Kb )
16.        action = a
17.      else
18.        a' = determineActionsSequence(P, ES, TE)
19.        action = a'
20.        Kb = Kb + (s, a')
21.      endif
22.    endif
23.  //Execution Phase
24.    executeAction(actionSeq)
25. end_MAPE
```

Figure 2. Context-aware self-adapting algorithm

Fig. 2 shows the proposed self-adapting algorithm which is based on a MAPE based control loop. The Monitoring Phase collects raw data about the system's self and execution environment which is used to represent the current context situation in a programmatic manner (lines 7-9). The Analysis Phase evaluates the context situation to detect whether the predefined set of conditions are fulfilled or not (lines 10-11). The Planning Phase decides and selects the appropriate adaptation action plan to be executed (lines 12-23) while the Execution Phase runs the selected action plan (lines 23-24).

The following sub-sections detail the first three phases of the self-adapting algorithm (the execution phase is trivial and is not further detailed).

A. The Monitoring Phase

The goal of the self-adapting algorithm monitoring phase is to collect the system's self and execution environment raw data and to represent the system context situation in a programmatic manner.

To represent the system's context situation in a programmatic manner we have used our RAP context model [4]. The RAP model defines two equivalent context data representations: (i) a set based representation, used to determine the system execution environment changes and (ii) an ontology based representation, used to infer context related knowledge by means of reasoning and learning algorithms. In the set based approach, the context information is modeled as a triple, C = <R, A, P >, where R is the set of context resources that provide raw data about the system's self and execution environment, A is the set of actions which are run to enforce the system execution guiding rules and P is a set of policies which define the rules that guide the system execution. In the ontology-based representation, the relationships between the context model sets are modeled in a context ontology core. A RAP context model instance represents a context situation and contains the values of the RAP context model sets elements in a moment of time (a snapshot of the system self and execution environment).

B. The Analysis Phase

The analysis phase targets the evaluation of the system context situations to detect those situations in which the rules that guide the system execution are broken. The set of guiding rules or conditions are described as RAP policies using our policy representation/evaluation model proposed in [22]. The policies are represented in XML and automatically converted into SWRL (Semantic Web Rule Language) [23] rules. The SWRL rules are injected into the RAP context model instance ontology for their evaluation. The SWRL rules reason about RAP context model instance ontology individuals in terms of ontology classes and properties.

For measuring the degree of fulfilling the set of policies we have defined the concept of context situation entropy (E_S) and its associated threshold (T_E). System context situation entropy measures the level of system's self and execution environment disorder by evaluating the degree of fulfilling the policies. If the context situation entropy is below the predefined threshold T_E, then all the policies are fulfilled and no adaptation is required. Otherwise, the system must execute adaptation actions to control its self and execution environment and to keep the system entropy below T_E (for further details see [21])

C. The Planning Phase

This phase deals with deciding and planning the actions that need to be taken in order to enforce a broken policy. The phase starts with identifying the previously encountered similar context situations in which the same policies were broken. If a similar situation is found, the same action plan is selected and executed. Otherwise, a new sequence of adaptation actions must be generated.

The sequence of adaptation actions is determined by using a reinforcement learning approach. The learning process determines all possible system context situations by simulating the execution of all available actions for each system context situation and building a decision tree. A

reward/penalty approach, calculated based on system's context situation entropy is used to find the best sequence of actions that the system may execute in a given context situation. The best sequence of adaptation actions is identified as the decision tree path with the minimum entropy penalty.

IV. Specializing the Self-Adapting Algorithm for Optimizing the Energy Consumption in a Service Center

In the following sub-sections each of the self-adapting algorithm MAPE phases are shown, highlighting their specialization and usage for optimizing the energy consumption in a service center. For each MAPE phase, the involved hardware/software resources and their interaction are presented and discussed.

A. The Monitoring Phase

The goal of the self-adapting algorithm monitoring phase is to collect the service center energy/performance related data (section A1) and to update/represent the service center context, also referred as the GAMES context (section A2). The GAMES context consists of the following categories of energy/performance data: (i) business data related to the GAMES-enabled application which is executed on the service center IT computing resources, (ii) current workload and energy consumption data of the service center IT computing resources and (iii) ambiental data related to the service center environment.

1) The Service Center Monitoring Infrastructure

The next two sub-sections present the monitoring infrastructures used to gather the current workload / energy consumption data and ambiental data. The business context data, is gathered from the GAMES-enabled application activities non-functional requirements and their KPI indicators, therefore no monitoring infrastructure is needed (see Table 1).

In our case study, we have considered a service center with the following IT resources: (i) five servers on which the GAMES-enabled application activities are executed and (ii) a set of sensors and facilities interconnected through a sensor network which control the service center environment.

TABLE I. KPIs of the GAMES-enabled Application Activities

Act. No	Activity KPI		
	CPU	Memory	Storage
1	3000 MHz	2048 MB	200 MB
2	2 x 1500 MHz	512 MB	400 MB
3	2 x 2000 MHz	1024 MB	256 MB

a) Monitoring Infrastructure for Service Center IT Computing Resources

To accurately capture the GAMES context workload and energy consumption data of the service center IT computing resources, we have adapted and used the monitoring architecture recommended by The Standard Performance Evaluation Corporation [24] which defines the following monitoring components (see Fig. 3): (i) *System Under Test (SUT)* - the service center IT computing resources whose workload and energy consumption data is collected, (ii) *Software Driver* - a software program stressing the SUT components, (iii) *Monitor* - a third-party software component that registers the workload and energy consumption data of various SUT resources and (iv) *Measurement Server* - the physical system on which the monitor and driver resides.

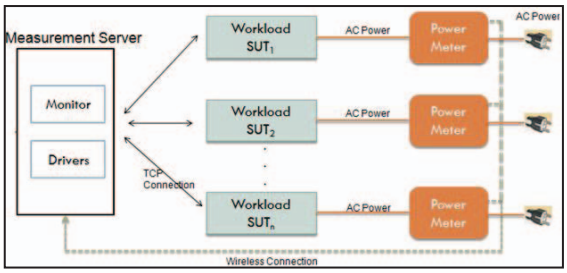


Figure 3. Monitoring architecture

Each monitoring components is mapped onto the service center hardware and software resources. *System Under Test (SUT)* is mapped on the service center server. For our test case we have defined five SUT service center physical servers. The configuration of each SUT is shown in Table 2. *Software Driver* is represented by a set of java based applications stressing different server resources (see Table 1 for these application requirements). We have defined two categories of applications: (i) individual activities - stressing only one server resource (such as HDD, CPU or memory) and (ii) combined activities - stressing combinations of server resources. The *Monitor* is represented by the Nagios monitoring framework [25]. Nagios monitor defines two major components: (i) Nagios software server, running on the Measurement Sever and collecting data about the SUTs and (ii) Nagios software daemons, running on each monitored SUT and sending data to the Nagios software server.

TABLE II. The Case Study Service Center SUT's

Name	CPU	Memory	Storage	Power source
SUT1	1 x 3000 MHz	2048 DDR2	250 GB @ 7200 rpm	300 W
SUT2	2 x 3000 MHz	2048 DDR2	250 GB @ 7200 rpm	365 W
SUT3	4 x 2000 MHz	4096 DDR3	146 GB @ 10000 rpm	480 W
SUT4	4 x 2260 MHz	6144 DDR3	146 GB @ 15000 rpm	675 W
SUT5	8 x 2000 MHz	8192 DDR3	300 GB @ 15000 rpm	2 x 1100 W

b) Monitoring Infrastructure for Service Center Ambiental Environment

The GAMES context ambiental data is collected through the Wi-microSystem wireless of Infusion Systems [26]. Also, set of facilities and actuators are used to control the service center ambiental characteristics and to enact the adaptation actions.

TABLE III. THE SERVICE CENTER AMBIENT MONITORING INFRASTRUCTURE COMPONENTS

Infrastructure Component	Type	Range of values / Available actions
Temperature	Sensor	[-20, 40] ℃ / -
Humidity	Sensor	[0, 100] % / -
Light	Sensor	{ON, OFF} / -
Air Conditioning Unit	Actuator	- / {Decrease by 5℃, Increase by 2 ℃}
Humidity Controller	Actuator	- / {Increase by 3 %,Decrease by 3 %}
Light Controller	Actuator	- / {Turn ON, Turn OFF}

The service center ambiental environment monitoring infrastructure components (sensors and actuators) and their range of values / available actions are shown in Table 3. The sensor collected data is encoded as MIDI messages which are real-time wirelessly transmitted, through Bluetooth waves, to the Measurement Server for analysis. The Bluetooth receiver, located on the Measurement Server, is mapped as a Virtual Serial Port (VSP). The MIDI message information is extracted using the Microsoft Windows API multimedia operations and published through web services (see Fig. 4).

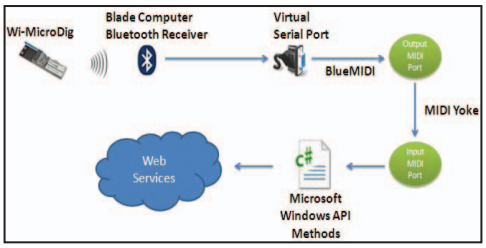


Figure 4. Data flow of service center ambiental information

For simulation purposes, we have provided two manipulation mechanisms for sensor values: a Random Disturbing Mechanism (RDM) and a Direct Manipulation Mechanism (DMM). RDM assigns values to the service center sensors using a predefined pattern (or random values) in order to generate complex situations in which adaptation is needed (i.e. the temperature and humidity in the service center are above a predefined limit). DMM uses an Extensible 3D (X3D) implementation, for representing the service center sensors as X3D objects (see Fig. 5). A click event performed on an object generates a change in the object state and implicitly, a new simulated sensor value. The DMM approach, which is further used as the main simulation tool, allows us to accurately reconstruct real scenarios and closely control their evolution.

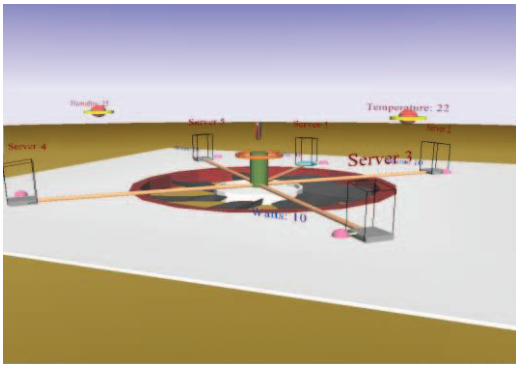


Figure 5. Service center X3D representation

2) Context Data Representation

For representing the GAMES context data in a programmatic manner we have adapted the RAP context model elements as presented below.

A *RAP context resource* is a physical or virtual entity which captures or generates the GAMES context data. The GAMES context data is collected through three different types of context resources, one for each category of context data: (i) The activities of the GAMES-enabled applications are resources for the business data, (ii) the service center IT computing resources, such as service center servers, provide the current workload and energy consumption data and (iii) the sensor and facilities, such as cooling/heating, lighting, humidity, are the resources for the ambiental data.

A *context action* interacts with the context resources in order to enforce the GPI and KPI indicators. We have identified three types of adaptation actions, one for each category of resources: (i) business adaptation actions (such as QoS / SLA renegotiation and/or KPI modification) applied on the GAMES-enabled application activities, (ii) service center IT computing resources adaptation actions (such as consolidation actions or DPM actions) applied on the service center computing resources and (iii) sensor and facilities adaptation actions (such as turning on the cooling device) applied on the service center sensor and facilities.

The *context policies* represent the predefined GPI and KPI indicators that must be enforced at run-time in the service center. GPI and KPI indicators are modeled by using three types of policies: (i) environmental policies imposing restrictions about the service center ambiental conditions; (ii) service center IT computing resources energy consumption policies and (iii) business policies describing the QoS / SLA related rules imposed for the execution of the GAMES-enabled application activities.

As mentioned before (see Section III A), the RAP model defines two equivalent context data representations: a set based representation and an ontology based representation.

In the ontology-based context representation, the service center specific elements are modeled as sub trees of the RAP core ontology by using *is-a* type relations (see Fig. 6).

Figure 6. GAMES context ontology representation

The RAP set based representation is used to evaluate the service center energy / performance changes while the ontology based representation is used to determine if the GPI and KPI indicators are fulfilled at runtime.

B. The Analysis Phase

In this phase the GAMES context ontological representation is analyzed against the predefined set of policies to detect the context situations in which the GPI and KPI indicators are not fulfiled. Table 4 shows the defined set of policies for the case study.

TABLE IV. SERVICE CENTER ENERGY/PERFORMANCE AND ENVIROMENTAL POLICIES

Environmental Policy	Accepted values
Temperature And Humidity	$Temperature \in (18°C, 23°C)$ $Humidity \in (20\ \%, 30\ \%)$
Light	$Light = \begin{cases} OFF, Room\ empty \\ ON, Room\ not\ empty \end{cases}$
Energy/Performance Policy	**Accepted values**
Workload values for SUT1	$CPU \in (2400\ MHz, 2600\ MHz)$ $MEM \in (1200\ Mb, 1600\ Mb)$ $HDD \in (180\ Gb, 230\ Gb)$
Workload values for SUT2	$CPU1 \in (2400\ MHz, 2600\ MHz)$ $CPU2 \in (1500\ MHz, 2000\ MHz)$ $MEM \in (1800\ Mb, 2000\ Mb)$ $HDD \in (180\ Gb, 230\ Gb)$

The rest of this section presents the steps of the analysis process used to determine if the energy / performance policy for SUT1 is fulfilled in the current GAMES context situation. The analysis process has the following steps:

Step1. The SUT1 energy / performance policy is represented in XML according to our policy model detailed in [22] (see Fig. 7).

```
<Reference Name="SUTPolicy1">
    <Restrictions>
       <PropertyRestriction Property="Server" Value = "Server1" Operator="ServerName"/>
       <PropertyRestriction Property="associated-component" Operator="Cpu">
          <PropertyRestriction Property="used" Operator=">" Value="2400"/>
          <PropertyRestriction Property="used" Operator="<" Value="2600" />
       </PropertyRestriction>
       <PropertyRestriction Property="associated-component" Operator="Memory">
          <PropertyRestriction Property="used" Operator=">" Value="1200"/>
          <PropertyRestriction Property="used" Operator="<" Value="1600" />
       </PropertyRestriction>
       <PropertyRestriction Property="associated-component" Operator="Storage">
          <PropertyRestriction Property="used" Operator=">" Value="180"/>
          <PropertyRestriction Property="used" Operator="<" Value="230" />
       </PropertyRestriction>
    </Restrictions>
</Reference>
```

Figure 7. SUT1 energy / performance policy XML representation

Step2. The XML of the energy / performance policy for SUT1 is converted into *SWRL rules* using our XML to SWRL conversion model presented in [22] (see Fig. 8 for the SWRL representation).

```
Server(?ServerRef)
^ name(?ServerRef, "Server1")
^ associatedComponent(?ServerRef, ?CpuRef)
^ CPU(?CpuRef)
^ associatedCore(?CpuRef, ?CoreRef)
^ used(?CoreRef, ?UsedCore)
^ swrlb:greaterThan(?UsedCore, 2400)
^ swrlb:lessThan(?UsedCore, 2600)
^ associatedComponent(?ServerRef, ?MemoryRef)
^ Memory(?MemoryRef)
^ used(?MemoryRef, ?UsedMemory)
^ swrlb:greaterThan(?UsedMemory, 1200)
^ swrlb:lessThan(?UsedMemory, 1600)
^ associatedComponent(?ServerRef, ?StorageRef)
^ Storage(?StorageRef)
^ used(?StorageRef, ?UsedStorage)
^ swrlb:greaterThan(?UsedStorage, 180)
^ swrlb:lessThan(?UsedStorage, 220)
-> respected(EnergyPolicy_1, true)
```

Figure 8. SUT1 energy / performance policy SWRL representation

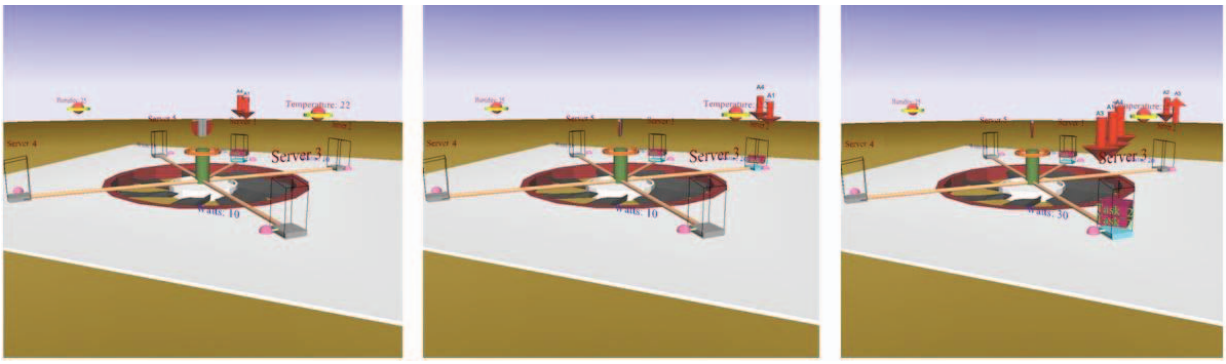
Step3. The SUT1 energy / performance policy SWRL representation is injected in the GAMES context situation ontology representation.

Step4. The policy is evaluated using the Pellet [27] reasoner.

Step5. The evaluation result is used to calculate the entropy of the current GAMES context situation. In our test case the entropy threshold T_E is set to 1 and the KPI and GPI related policies can be evaluated to true or false (1 / 0). The context entropy for the current GAMES context situation is obtained by evaluating the degree of fulfilling of all KPI and GPI context policies (see relation 1 for the entropy evaluation formula).

$$E=E_S=\sum \text{eval(KPI)} + \sum \text{eval(GPI)} \quad (1)$$

Step6. If the obtained entropy is higher than the entropy threshold, the planning phase is fired; otherwise nothing happens because no adaptation is needed.



Step 1 *Step 2* *Step 3*

Figure 9. Planning phase scenario tracing

C. The Planning Phase

The phase goal is to decide and select the actions that need to be taken in order to enforce a broken context policy. The phase starts with identifying the previously encountered similar context situations in which the same policies had been broken. If a similar situation is found, the same adaptation action is taken. Otherwise, the sequence of actions is determined through a reinforcement learning algorithm, by using a policy iteration technique. The reinforcement learning process considers all possible GAMES context situations and builds a decision tree by using a *what-if* approach. The execution of all available actions is simulated and rewards / penalties are used to find the best sequence of actions.

In this case study we have defined the following generic adaptation actions: "A1: Deploy Activity on Server", "A2: Put Server in Hibernate State", "A3: Move Activity from Server1 to Server2", "A4: Put Server in Active State".

For reducing the overhead, the algorithm keeps track of the visited nodes while expanding the search space in a greedy manner. During each phase, the algorithm is searching for the maximum rewarded node from the existing leafs and expands the searching tree. The expansion goes on until the best path is found. The reward function at each node is calculated using the formula:

$$f_{i+1} = f_i + \gamma(E_i - E_{i+1} + \frac{MaxActionCost}{CurrentActionCost}) \qquad (2)$$

In formula (2) E_i is the entropy of the current context state, E_{i+1} is the entropy value of a future context state obtained by executing a simple adaptation action, γ represents the future reward's discount factor, *MaxActionCost* refers to the highest cost of executing a simple adaptation action (the most expensive action) determined by using the *what-if* analysis, *CurrentActionCost* is the cost of executing the current simple adaptation action.

To test the algorithm we have considered a scenario in which the GAMES-enabled application activities arrive in the order presented in Table 1. In the initial service center state, all its SUTs are considered as being hibernating.

The planning phase tracing for this scenario consists of the following steps (see Fig. 9):

Step1. Activity 1 is received. The KPI and GPI context policies are evaluated. The KPI requirements of activity 1 are broken in the server center initial state because all the SUT's are hibernating. The reinforcement learning algorithm starts evaluating the options, and finds the following sequence of adaptation actions: "A4: Put SUT1 in Active State", "A1: Deploy Activity 1 on SUT1".

Step2. Activity 2 is received. The KPIs of activity 2 are broken because SUT1 reached its load threshold and all other SUT's are hibernating. The following sequence of adaptation actions is determined: "A4: Put SUT2 in Active State" and "A1: Deploy Activity 2 on SUT2".

Step3. Activity 3 is received. There are no available resources to deploy it on SUT2. The following sequence of adaptation actions is determined (see Fig. 10 for the search tree): "A4: Put SUT3 in Active State", "A1: Deploy Activity 3 on SUT3", "A3: Move Activity 2 from SUT2 to SUT3", "A2: Put SUT2 in Hibernate State".

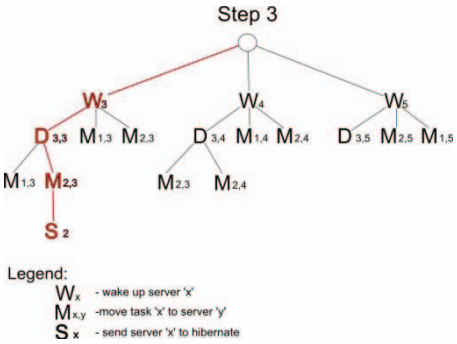


Figure 10. Planning phase "what-if" search tree for step 3

After finding the best sequence of adaptation actions for each GAMES context situation, the association (GAMES

context situation, adaptation actions sequence) is stored for further reference.

V. CONCLUSIONS

This paper proposes a context aware self-adapting algorithm for balancing the performance and energy consumption of a service center. The self-adapting algorithm automatically detects / analysis the workload and performance changes in the service center then decides on how it should adapt in order to minimize the energy consumption. The obtained results are promising, showing that the algorithm is capable to manage at run-time the service center energy efficiency by dynamically taking consolidation and dynamic power management actions.

For further development we intend to evaluate the energy savings by using the self-adapting algorithm.

ACKNOWLEDGMENT

The work has been done in the context of the EU FP7 GAMES project [3].

REFERENCES

[1] V. Metha, "A Holistic Solution to the IT Energy Crisis", {Online}, Available at http://greenercomputing.com, 2007.

[2] J.M.Kaplan, W.Forrest, N.Kindler, "Revoluzioning Data Center Energy Efficiency", McKinsey&Company, Tech. Rep., 2008.

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

[4] T. Cioara, I. Anghel, I. Salomie "A Generic Context Model Enhanced with Self-configuring Features", Journal of Digital Information Management, Volume 7/3, pp.159-165, ISSN 0972-7272, 2009.

[5] L. A. Barroso, U. Hölzle, "The Case for Energy-Proportional Computing", IEEE Computer, vol. 40, no 12, pp. 33-37, 2007.

[6] Alexandre Mello Ferreira, Kyriakos Kritikos, Barbara Pernici, "Energy-Aware Design of Service-Based Applications", ICSOC/ServiceWave, pp. 99-114, Stockholm, 2009.

[7] D. Ardagna, M. Comuzzi, E. Mussi, P. Plebani, B. Pernici, "PAWS: a framework for processes with adaptive Web services", IEEE Software, 24(6), pp. 39-46, 2007.

[8] G. Canfora, M. Penta, R. Esposito, M. L. Villani, "QoS-Aware Replanning of Composite Web Services", Proc. of Int. Conference on Web Services (ICWS'05), IEEE, Orlando, pp. 121–129, 2005.

[9] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end QoS constraints", ACM Trans. Web, 1(1), pp. 1–26, 2007.

[10] D. Ardagna, C. Francalanci, M. Trubian, "Joint Optimization Of Hardware And Network Costs For Distributed Computer Systems", IEEE Trans. on Systems, Man, and Cybernetics, Vol. (2), 2008.

[11] W. Lin, Z. Liu, C.H. Xia, L. Zhang, "Optimal Capacity Allocation for Multi-Tiered Web Systems with End-to-End Delay Guarantees", Perform. Eval., vol. 62, no. 1-4, pp. 400-416, Oct. 2005

[12] J. S. Chase, D. C. Anderson, "Managing Energy and Server Resources in Hosting Centers", In ACM Symposium on Operating Systems principles, vol. 35, no. 5, 2001.

[13] M. Tanelli, D. Ardagna, M. Lovera, L. Zhang, "Model Identification for Energy-Aware Management of Web Service Systems", Proceedings of the 6th International Conference on Service-Oriented Computing, pp.599-606, 2008, Australia.

[14] D. Kusic, N. Kandasamy, "Risk-aware limited lookahead control for dynamic resource provisioning in enterprise computing systems", Cluster Computing, 10(4), pp. 395-408, 2007.

[15] S. Ganeriwal, A. Kansal and M. B. Srivastava, "Self Aware Actuation for Fault Repair in Sensor Networks", Proc. of the IEEE International Conference on Robotics and Automation, pp. 5244- 5249, 2004.

[16] J. Saia, A. Trehan, "Picking up the Pieces: Self-Healing in reconfigurable networks", In Proc. of IEEE International Parallel and Distributed Processing Symposium, pp. 1 - 12, 2008.

[17] R. Das, J. O. Kephart, C. Lefurgy, G. Tesauro, David W. Levine and Hoi Chan, "Autonomic multi-agent management of power and performance in data centers", In Proc. AAMAS, pp. 107-114, 2008.

[18] Q. Zhu and Y. Zhou, "Chameleon: A Self-Adaptive Energy-Efficient Performance-Aware RAID", IBM Austin Conference on Energy-Efficient Design (ACEED), Texas, 2005.

[19] G. Chen, W. He, "Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services", the 5th USENIX Symp. on Net. Syst. Design and Impl., pp. 337-350, 2008.

[20] R. Buchty, J. Tao, W. Karl, "Automatic Data Locality Optimization Through Self-optimization", LNCS 4124, pp. 187–201, Springer-Verlag Berlin Heidelberg, 2006.

[21] T. Cioara, I. Anghel, I. Salomie, "A Reinforcement Learning based Self-healing Algorithm for Managing Context Adaptation", 1st Int. W'shop on Comm., Collab. and Social Net. in Perv. Comp., 2010.

[22] T. Cioara, I. Anghel, I. Salomie "A Policy-based Context Aware Self-Management Model", 11th Int. Symp. on Symbolic and Numeric Alg. for Scientific Comp., SYNASC 2009, Timisoara, Romania, pp.333-341, ISBN: 978-0-7695-3964-5.

[23] A. B. Bener, V. Ozadali, "Semantic matchmaker with precondition and effect matching using SWRL", Expert Systems with Applications Journal, Volume 36 , Issue 5, pp. 9371-9377, July 2009.

[24] The Standard Performance Evaluation Corporation, www.spec.org.

[25] D. Josephsen, "Building a Monitoring Infrastructure with Nagios", ISBN:0132236931, Prentice Hall PTR, 2007.

[26] Infusion Systems Ltd, http://www.infusionsystems.com.

[27] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, "Pellet: A practical OWL-DL reasoner, Web Semantics: Science", Services and Agents on the World Wide Web, Vol. 5 (2), pp. 51-53, June 2007.