Dynamic Consolidation Methodology for Optimizing the Energy Consumption in Large Virtualized Service Centers

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

Pierluigi Plebani
Politecnico di Milano
Milano, Italy
plebani@elet.polimi.it

Abstract—**In this paper we approach the high energy consumption problem of large virtualized service centers by proposing a dynamic server consolidation methodology for optimizing the service center IT computing resources usage. The consolidation methodology is based on logically structuring the service center servers hierarchical clusters, consolidation decisions being taken in each cluster using a reinforcement learning based algorithm. The methodology defines two ways of consolidation decisions propagation across the hierarchy: bottom-up propagation for the dynamic power management actions and top-down propagation for the consolidation actions. The consolidation decision time complexity analysis shows that the methodology usage in large service centers improves the decision time with a factor proportional with the ratio between the service center total number of servers and the logical clusters' number of servers.**

Keywords—large service centers, dynamic server consolidation, reinforcement learning, energy consumption, hierarchical clusters.

I. Introduction

OVER the last years the energy efficiency management of service centers has emerged as one of the most critical environmental challenges to be dealt with. A U.S. Environmental Protection Agency report to Congress [1] shows that in just five years, the electricity consumed by service centers and their additional infrastructure will double and the trend is expected to accelerate driven by their shift towards cloud computing. One of the major sources of the service centers huge amount of consumed energy is the inefficient utilization of IT computing resources. The IT computing resources in today's service centers are under-used, usually operating below the optimal loads for energy efficiency. According to [2] in a service center about 30% of servers consume energy without doing any actual work. In large service centers with thousands of servers, the computing resources average utilization ratio is between 5 and 10 percent providing a huge opportunity for organizations to reduce the service center energy consumption. A state of the art technique for providing an optimal energy performance trade-off in service centers is resource consolidation using virtualization.

In this paper the dynamic server consolidation of large service centers is approached by using a hierarchy structure to logically organize the service center in clusters. Each logical cluster is being managed by its own instance of a reinforcement learning based consolidation algorithm presented in one of our previous papers [14]. The a reinforcement learning based consolidation algorithm was proven to be effective for increasing the service centers energy efficiency but its main drawback is the fact that the decision time complexity increases with the number of servers and become unsatisfactory for medium and large service centers. We have showed that reinforcement learning consolidation solution complexity for a large number of servers decrease when the service center is organized according to our hierarchical structure. The consolidation decision time improves with a factor proportional with the ratio between the service center total number of servers and the logical clusters' number of servers.

The rest of the paper is structured as follows: Section II presents the state of the art for dynamic consolidation, Section III presents an overview of the reinforcement learning based consolidation algorithm underlying the consolidation decision time problem for large service centers, Section IV introduces the proposed consolidation methodology based on logically organizing the service center in hierarchical clusters, Section V presents an analysis of the consolidation methodology decision time complexity, while Section VI concludes the paper.

II. Related Work

Resource consolidation (or server consolidation) in service centers aims at combining the virtualized workloads that are executed on different machines (servers) for obtaining an optimal number of computing resources usage [3]. As shown in [4], the greatest challenge for consolidation methods is deciding which workloads should be combined on a common physical server since resource usage, performance and energy consumption are not additive.

A service center may be classified taking into account the number of servers as [5]: (i) small service centers having a number of servers between 101 and 500, (ii) medium service

This work has been done in the context of the EU FP7 GAMES project (http://www.green-datacenters.eu/).

centers having between 501 and 5000 servers and (iii) large service centers with a number of servers usually over 5000. Many state of the art solutions regarding service center servers consolidation approach the energy consumption optimization through resource allocation or consolidation.

A thermal aware workload scheduling and consolidation solution aiming to reduce the power consumption and temperatures in data centers was proposed in [6]. The simulation results show that the algorithm can significantly reduce the energy consumption with some degree of performance loss. In [7] a novel technique for controlling the service centers servers CPU allocation and consolidation based on first order Kalman filter is presented. In [8] the server consolidation problem is approached for small service centers as a constraint satisfaction problem. The authors also propose a heuristic for approaching the server consolidation in large service centers. In [9], the authors propose an algorithm for consolidating virtual machines in large service centers based on a simple gossip protocol. To enable energy efficient consolidation, the inter-relationships between energy consumption, resource utilization, and performance of consolidated workloads must be considered [10]. In [11] the authors reveal that energy performance trade-offs for consolidation and optimal operating points exist. A bio-inspired workload consolidation algorithm for service centers based on defining some autonomous scouting entities is defined in [12]. The entities try to find the suitable server for migrating a virtual machine (worker entity).

Optimal computing resources allocation techniques for server clusters based on reinforcement learning are proposed in [16]. Learning techniques are also used to trade-off between computing resources power consumption and performance during the allocation process [17]. In [13] a consolidation methodology that uses machine learning to deal with uncertain information is discussed. Pervious server behavior data is used to predict and estimate the current power consumption and also to improve the scheduling and consolidation decisions.

The presented state of the art approaches fail to consider the scalability problem when varying the service center dimension (number of servers) and applying different consolidation algorithms.

III. Reinforcement Learning based Dynamic Server Consolidation

In a previous published paper [14] we have approached the problem of dynamic server consolidation in virtualized service centers by proposing the development of an energy aware run-time consolidation algorithm based on reinforcement learning. To make this paper self-contained in Section A we present a short overview of the reinforcement learning based consolidation algorithm. More details can be found in [14]. Also the reinforcement learning consolidation decision time problem statement for large service centers is described in Section B.

A. Consolidation Algorithm Overview

The reinforcement learning consolidation algorithm has three main phases: (i) representing the service center energy related context data in a programmatic manner, (ii) calculating the service center greenness level and (iii) deciding on the consolidation actions that must be executed to bring the service center in an energy efficient state.

1) Context data representation

To represent the energy related context data in a programmatic manner we have defined an ontology based context model: the EACM (Energy Aware Context Model) model [15]. The energy related context data is represented in the EACM model using three main concepts: Context Resources, Context Actions and Context Policies.

Context Resources define the physical or virtual entities that generate and / or process energy related context data. For a service center we have identified three sub-types of Context Resources: Facility Resources, Computing Resources and Application Resources. Facility Resources are physical entities which capture the service center ambient data (sensors) and enforce the design time defined environmental conditions (actuators). Computing Resources are physical entities that consume energy as a result of executing workload. The main Computing Resource of a service center considered in our representation is the server. Application Resources are the software entities executed on the service center computing resources as incoming workload. An activity is modeled through its processor, memory and hard disk computing resources requests.

Context Actions define the set of design time enabled adaptation actions that may be executed at run time to enforce the service center energy efficiency goals. We have identified three sub-types of adaptation actions: Facility Adaptation Actions (e.g. adjust the room temperature or start the air conditioner), IT Computing Adaptation Actions and Application Adaptation Actions (e.g. application redesign for energy efficiency). We have defined two main sub-classes of IT Computing Adaptation Actions: resource consolidation actions (Deploy Activity, Migrate Activity) and dynamic power management actions (Wake-up server, Turn-off server).

Context Policies define the service center energy efficiency goals through a design time established set of Green and Key Performance Indicators (GPIs/KPIs). We have defined three sub-classes of GPIs/KPIs: (1) environmental, imposing restrictions about the service centre ambient conditions (e.g. the temperature in the service center must be under 21°C), (2) IT Computing, describing the energy/performance characteristics of the service centre computing resources (e.g. the server CPU is efficiently used for a load between 60%-80%) and (3) Application, specifying the rules (QoS requests) imposed by the business application for execution (e.g. for optimal execution time the application needs 1Gb of allocated physical memory).

2) Service center greenness level

To calculate the service center greenness level we have defined the concept of service center context situation entropy (E_S) [15]. The entropy is a metric which establishes the service center context situation degree of complying with the design time defined set of GPIs/KPIs. The GPIs/KPIs are represented in SWRL (Semantic Web Rule Language) reasoning rules and automatically evaluated against the EACM model instance ontology implementation. The entropy value of a service center context situation (S) is calculated using the following relation:

$$E_S = \sum_i w_{P_i} \sum_j w_{r_{ij}} * v_{r_{ij}} \qquad (1)$$

where: (i) w_{P_i} is the weight of GPIs/KPIs policy i, and represents the importance of the policy in the service centre context, (ii) $w_{r_{ij}}$ is the weight of the service centre context resource j in the GPIs/KPIs policy i and reflects the context resource importance for that policy and (iii) $v_{r_{ij}}$ is the deviation between the value recorded by the context resource j and the accepted value defined by policy i (if x is the accepted value of the context resource j defined by the GPI/KPI policy i and the actual value recorded by the resource j is r_{ij}, then $v_{r_{ij}} = r_{ij} - x$).

The entropy value is used to trigger the consolidation process as follows: if the current service center context situation entropy value is above a predefined threshold, the service center greenness level is acceptable and consolidation is not required, otherwise the reinforcement learning consolidation process is started.

3) Consolidation actions selection

To decide on the consolidation actions that have to be executed if the service center is not in an energy efficient state a reinforcement learning based solution is used (see Fig. 2 for the algorithm pseudo-code).

The consolidation process starts from the current service center context situation, simulates the execution of all available consolidation (Deploy or Migrate activity) or dynamic power management actions (Turn-on or Turn-off server) based on a reward / penalty approach and builds a decision tree (see Fig. 1).

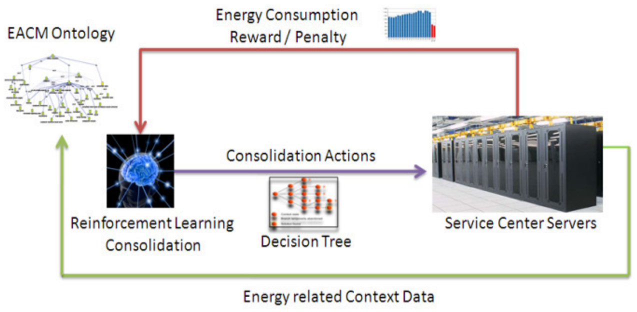

Fig. 1 The reinforcement learning consolidation decision process

A decision tree node stores: (i) the EACM instance describing the service center energy related context situation, (ii) the list of actions that were simulated to generate that EACM instance, (iii) the EACM instance calculated entropy value (relation 1) and (iv) the reward value calculated for the list of actions simulated so far. The reward for executing an action in a certain situation is calculated as follows:

$$R_{S+1} = R_S + \gamma * (E_{S+1} - E_S - ActionCost) \qquad (2)$$

where: (i) R_{S+1} represents the reward for the newly generated (current) tree leaf node *S+1*, (ii) R_S represents the reward of the current leaf node parent, (iii) E_{S+1} and E_S represent the calculated entropy values for the EACM instance stored in the leaf node *S+1* and its parent *S*, while (iv) $ActionCost$ represents an associated design time consolidation and dynamic power management action cost value.

```
Input: pQueue – a priority queue containing the current step reinforcement learning
                         tree leaf nodes sorted by their rewards
      highestRewardNode – the reinforcement learning tree node with the highest reward
Output: TreeNode – the reinforcement learning tree node containing the sequence of
                        consolidation actions that should be executed to bring the service center
                        in a energy efficiency state and its associated reward

TreeNode reinforcementLearningConsolidation (PriorityQueue <TreeNode> pQueue,
                                                                TreeNode highestRewardNode)
begin
  TreeNode currentLeaf = pop (pQueue)
  if (currentLeaf == NULL) then return highestRewardNode
  if (getEntropy(currentLeaf) < $T_E$) then return currentLeaf
  if ((highestRewardNode == NULL)
                  or (getReward(currentLeaf) > getReward (highestRewardNode))) then
    highestRewardNode = currentLeaf
  else
    brokenGPI_KPI_Policies = getBrokenGPI_KPI_Policies (currentLeaf)
    Action = NULL
    foreach policy in brokenGPI_KPI_Policies
      if (getSubject(policy) instanceOf ApplicationActivity) then
        activityInstance = getSubject(policy)
        foreach server in sortServersByDistanceToActivity(currentLeaf, activityInstance)
          if (hasResourcesFor(server, activityInstance) and notRunning(activityInstance)) then
            set Action to (DEPLOY activityInstance on server)
        foreach server in getTurnedOffServers (currentLeaf)
          if (hasResourcesFor(server, activityInstance)) then
            set Action to (TURNON server)
      if (getSubject(policy) instanceOf ComputingResource) then
        serverInstance = getSubject(policy)
        foreach activity in getRunningActivities(serverInstance)
          foreach server in sortServersByDistanceToActivity(currentLeaf, activity)
            if (hasResources(server, activity)) then
              set Action to (MIGRATE activity from serverInstance to server)
        if (getRunningActivities(serverInstance) = = NULL) then
          set Action to (TURNOFF server)
    TreeNode nextLeaf = genNextLeaf (Action)
    addNewLeafNode(pQueue, nextLeaf)
  return reinforcementLearningConsolidation (pQueue, highestRewardNode)
end
```

Fig. 2 The reinforcement learning consolidation algorithm

A tree path between two nodes $Node_0$ and $Node_n$ defines the sequence of actions that executed starting from $Node_0$ service center context situation generates the new service center context situation stored by node $Node_n$. The maximum reward path in the tree represents the sequence of actions that must be executed for consolidating the service center servers.

B. Consolidation Decision Time Problem

The dynamic reinforcement learning process takes consolidation decisions in reasonable time frames (less than 85 seconds for 50 virtual activities to be consolidated) for small service centers (100 servers). When dealing with medium and large service centers (no. servers higher than 500), the consolidation decision time grows exponentially (see Fig. 3).

To evaluate the consolidation process decision time we have simulated a service center with varying numbers of physical servers on which the workload virtualized tasks must be deployed. To ease the estimations, we have considered homogenous service centers with identical server hardware resources configuration (1 CPU with 8 cores x 3000 MHz and 6000 MB Memory). The workload tasks that need to be deployed are also homogenous (a task's hardware request is: 8 CPU Cores with 500 MHz frequency and a 900 MB amount of memory).

Fig. 3 chart shows the results of the consolidation process decision time evaluation. The decision time grows exponentially with the service center number of servers and the workload number of tasks that have to be deployed. Analyzing the results it can be seen that for a service center with around 1000 servers the consolidation decision process time, involving the deployment of 200 tasks, is over 2500 seconds (about 40 minutes). In case of dynamic consolidation which usually involves extremely dynamic workload, this time is unsatisfactory.

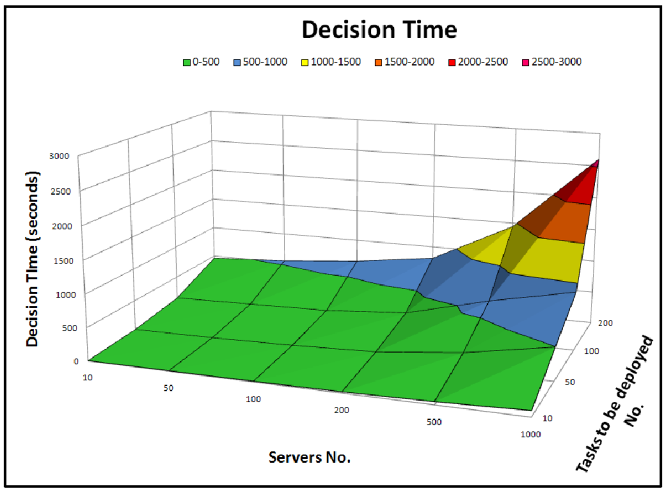

Fig. 3 The reinforcement learning consolidation decision time

Taking into account the above presented time results for medium and large service centers, in this paper we propose a methodology for reducing the decision time by logically organizing the service center servers using a hierarchical clusters structure. Each structure element contains a cluster of service center computational resources (servers or other clusters) managed by its one instance of the reinforcement learning consolidation algorithm.

IV. Methodology for Dynamic Server Consolidation in Large Service Centers

To solve the reinforcement learning consolidation decision time problem in large service centers, we propose a consolidation methodology based on logical clustering the service center in a hierarchical manner and associating to each cluster a specific reinforcement learning algorithm instance (see Fig. 4).

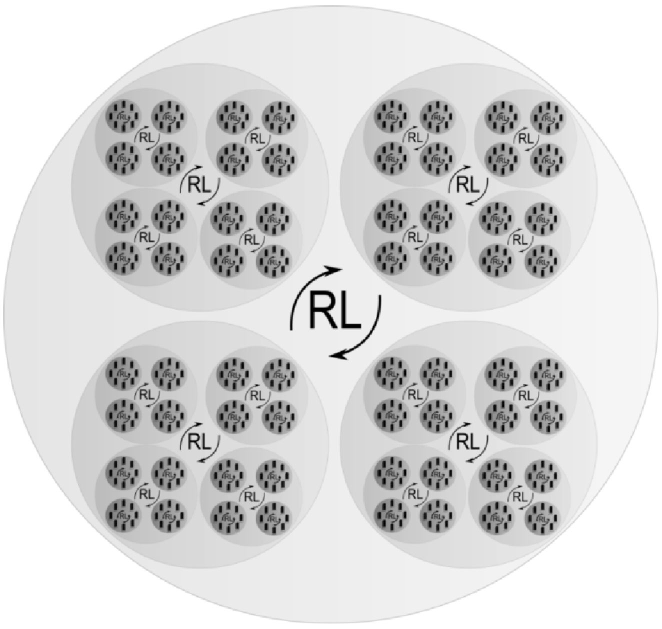

Fig. 4 Logical structuring of the service center servers using hierarchical clusters

A. Service Center Hierarchal Clusters Structure

The bottom layer of the hierarchical structure (level 0) is composed of service center physical servers. The server is the basic service center computational resource atomic granule for which the consolidation actions are considered. On the next hierarchical layer (level 1), the bottom layer servers are grouped into logical clusters, each cluster being managed by its own new instance of the reinforcement learning consolidation algorithm.

Definition 1. A logical cluster is a level 1 cluster if and only if it groups two or more physical servers (level 0 computational resources) that will be managed by the same instance of the reinforcement learning consolidation algorithm.

$$\begin{gathered}(c^{L1}[S_k, RL] is a level 1 cluster) \leftrightarrow \\ (\forall s_k \in S_k, s_k is a server) and \left(2 \leq \left|\left|S_k\right|\right| \leq MAX\right) and \\ (\exists RL(S_k) such that RL manages S_k) \qquad (3)\end{gathered}$$

Level 1 clusters are recursively grouped into higher layer logical clusters on level 2, each obtained cluster being managed by its one reinforcement learning algorithm instance.

Definition 2. A logical cluster is a level 2 cluster if and only if it groups two or more level 1 logical clusters that will be managed by the same instance of the reinforcement learning consolidation algorithm (see Fig. 4).

$$(c^{L2}[(C^{L1}{}_k, RL] isalevel2cluster) \leftrightarrow$$
$$(\forall(c^{L1}{}_k \in C^{L1}{}_k, c^{L1}{}_k isaLevel1Cluster) and$$
$$(2 \leq ||C^{L1}{}_k|| \leq MAX) and$$
$$(\exists\, RL(C^{L1}{}_k) \text{ such that } RL \text{ manages } C^{L1}{}_k) \tag{4}$$

To generalize, we can state that a level n cluster (0<n<TopMostLevel - level 0 and the top most level of the hierarchical structure does not fit in this definition) can be defined as follows:

$$(c^{Ln}[(C^{Ln-1}{}_k, RL] isalevel\text{ n } cluster) \leftrightarrow$$
$$(\forall(c^{Ln-1}{}_k \in C^{Ln-1}{}_k, c^{Ln-1}{}_k isLevel\text{ n} - 1Cluster) and$$
$$(2 \leq ||C^{Ln-1}{}_k|| \leq MAX) and$$
$$(\exists\, RL(C^{Ln-1}{}_k) \text{ such that } RL \text{ manages } C^{Ln-1}{}_k) \tag{5}$$

A logical cluster is a level n cluster if and only if it groups level n-1 clusters that will be managed by the same instance of the reinforcement learning consolidation algorithm.

Definition 3. A logical cluster is the top most cluster of the hierarchical structure (also called meta cluster) if and only if it logically groups all the clusters defined on the layer below it. On the topmost level of the hierarchy it must exists a single meta cluster.

B. Consolidation Decision Propagation in the Hierarchy

The reinforcement learning consolidation algorithm instances decisions are propagated across the service center logical hierarchical structure in two manners: (i) top-down, for decisions implying the execution of consolidation actions (deploy or migrate activity) and (ii) bottom-up for decisions implying the execution of dynamic power management actions (turn-on and turn-off server).

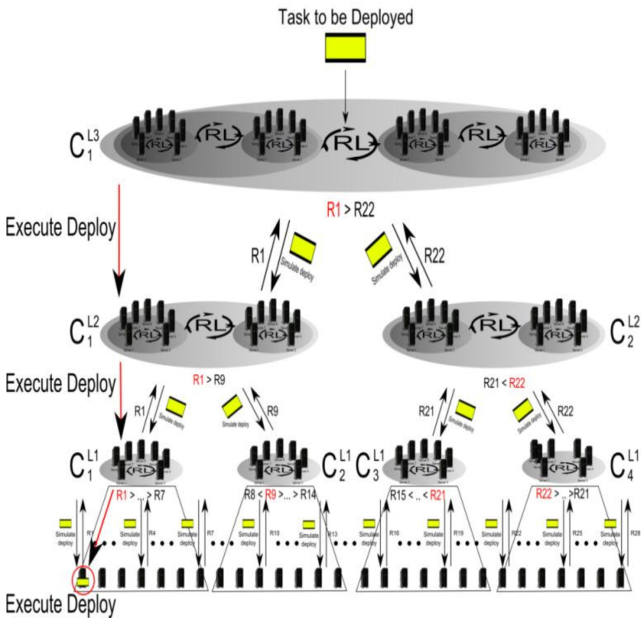

Fig. 5 The deploy action propagation example

The *deploy* activity decision is taken only by the meta cluster which receives the workload that the service center must execute (see Fig. 5). The decision and its associated activity is propagated to all the reinforcement learning algorithm instances controlling the inferior layer logical clusters. Each algorithm instance will simulate the activity deployment on the resources that it controls, calculates the associated reward and propagates the decision to the logical cluster algorithm instances below it. This propagation process continues recursively until the bottom layer is reached. The activity will be deployed on the server which is the leaf of the hierarchical structure path with the maximum reward.

The *migrate* activity decision (from one cluster to another) can only be taken by the reinforcement learning algorithm controlling both logical clusters (see Fig. 6). The migrate decision is also propagated in top-down manner as follows (see Fig. 6): (i) in the hierarchical structure sub-tree having as root the source logical cluster, a *destroy* activity action is propagated and (ii) in the hierarchical structure sub-tree having as root the destination logical cluster a *deploy* activity action is propagated. The destroy activity propagation is similar with the pattern described for deploying a task.

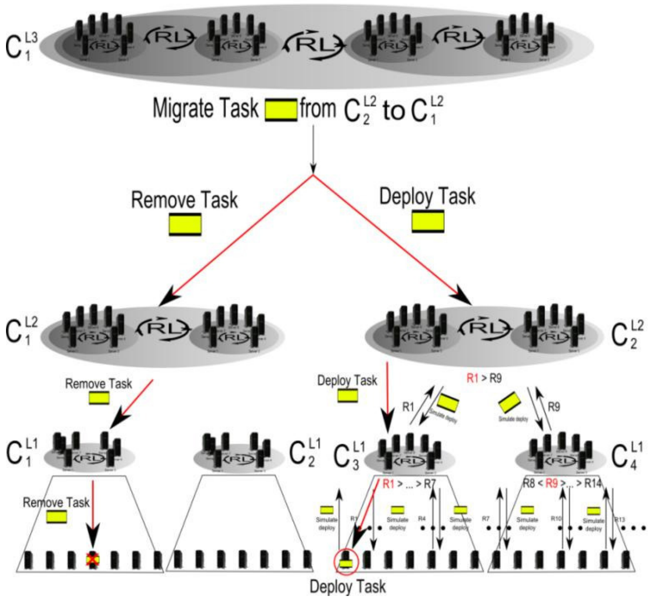

Fig. 6 The migrate activity action propagation example

The *turn-on* and *off* server / cluster actions are propagated across the hierarchical structure in a bottom-up manner. The decision is taken locally by the reinforcement learning consolidation algorithm that controls the computing resources that are turned-on or off. The decision is then signaled to the reinforcement learning algorithm structure controlling the upper level logical cluster which in turn investigates the possibility of turning-on / off the entire cluster containing the inferior level resources which were turned-on or off.

V. Consolidation Time Complexity Analysis

In this chapter the consolidation decision time complexity is being estimated for a large service center with N servers ($N > 5000$) that needs to accommodate M virtualized activities in an energy efficient manner. Two different cases are considered: (i) the service center has no logical organization and (ii) the service center is logical organized using the proposed hierarchical clusters structure described in Section IV.

In *the first case*, the reinforcement learning algorithm considers all the service center servers and virtual tasks when a consolidation decision needs to be taken. As it can be noticed from Section III, the reinforcement learning consolidation algorithm constructs a decision tree and searches for the best sequence of consolidation actions to be taken, in a certain situation, using a depth first search algorithm. The algorithm complexity is usually expressed as B^D where B represents the decision tree branching factor while D is the depth factor. The learning tree branching factor and the depth factor are equal to the number consolidation / dynamic power management actions that the reinforcement learning algorithm can consider at each step.

In the worst case scenario this number is given by the sum of (see relation 6): (i) the number of possible "turn-on server" actions (in the worst case scenario is equal with the number of turned-off servers), (ii) the number of possible "turn-off server" actions (in the worst case scenario is equal with the number of turned-on servers), (iii) the number of possible "deploy activity" actions (the number of un-deployed tasks multiplied with the number of available servers for the worst case scenario), and (iv) the number of possible "migrate activity" actions (the number of already deployed activities multiplied with the number of up and running servers in the worst case scenario).

$$\begin{aligned} B = D = {} & nrTurnedOffServers + \\ & nrTurnedOnServers + \\ & nrUndeployedActivities * nrTurnedOnServers + \\ & nrDeployedActivities * nrTurnedOnServers \end{aligned} \quad (6)$$

By grouping and factoring relation 6 elements, the branching and depth factors can be also calculated using the following relation

$$B = D = N + M * nrTurnedOnServers \leq N + M * N \quad (7)$$

where N is the service center total number of servers while M is the total number of service center virtualized activities considered for consolidation.

Therefore the consolidation time decision complexity for a service center with N servers that needs to accommodate M virtualized tasks in the worst case scenario is:

$$\left((N + M * N)^{(N+M*N)}\right) = O\left((M * N)^{(M*N)}\right) \quad (8)$$

In *the second case*, when the service center is logically organized by using the proposed hierarchical cluster structure, there will be multiple reinforcement learning consolidation algorithms that are executed on logical clusters with a smaller number of computational resources. For simplicity, we consider that the hierarchical structure logical clusters are uniformly created with the same number of computational resources c and the service center total number of servers N can be expressed as c^{kmax} (where $kmax$ is to hierarchy total number of layers). In this case at each hierarchical structure level, there will be a number of N/c^k clusters where N is the service center total number of servers and k is the level number. Using the hierarchical structure meta cluster definition which states that on the top-most level of the hierarchy a single cluster may exist, we can compute the maximum number of hierarchical levels as:

$$\frac{N}{c^{kmax}} = 1 \rightarrow kmax = [log_c N] \quad (9)$$

The reinforcement learning algorithm complexity for a cluster is $O(Mc)^{(Mc)}$ where c is the number of computational resources from a cluster and M is the number of activities considered in the consolidation decisions. But since the consolidation decisions taken by a reinforcement learning algorithm instance are propagated in the hierarchical structure sub-tree under it, we can state that *M=1* for all the hierarchical reinforcement learning algorithm instances except the one taking the actual consolidation decision. The overall consolidation decision time complexity is:

$$O(M * c)^{(M*c)} + (M - 1) * kmax * O(c^c) = O(M * c)^{(M*c)} \quad (10)$$

Considering relations 8 and 10, we can state that using the proposed methodology the consolidation decision time complexity remains exponential but grows with a much slower rate:

$$O\left((M * N)^{(M*N)}\right) > O(M * c)^{(M*c)} \; because \; c \ll N \quad (11)$$

The consolidation decision time when the service center logical hierarchical structuring is used, significantly improves when the ratio between the number of computation resources from clusters (c) and the service center total number of servers (N) decreases. If the difference between c and N is small there are few logical clusters created and the algorithm complexity remains the same:

$$\lim_{c \to N} O(M * c)^{(M*c)} = O\left((M * N)^{(M*N)}\right) \quad (12)$$

VI. Conclusions

In this paper a server consolidation methodology for large service centers based on logically clustering the service center in a hierarchical manner is proposed. Each logical cluster is being managed by its own instance of a reinforcement learning based consolidation algorithm.

By analyzing the consolidation solution complexity, it can be seen that using the proposed methodology the consolidation decision time complexity remain exponential but its growing rate is much slower. Our methodology consolidation decision time rate of improvement is proportional with the ration between the service center total number servers and the logical clusters number of servers.

For future work we intend to implement and test the proposed methodology for a simulated large service center with the goal of assessing its energy efficiency. We will take into account the overhead and energy penalty induced by powering on/off or migrating a task in the methodology clusters and we will calculate the Deployed Hardware Utilization Ratio (DH-UR) for the testing service center.

References

[1] U. S. Environmental Protection Agency, ENERGY STAR Program, *Report to Congress on Server and Data Center Energy Efficiency,* Public Law 109-431, 2007.

[2] M. Uddin, 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] N. E. Jerger, D. Vantrease and M. Lipasti, “An Evaluation of Server Consolidation Workloads for Multi-Core Designs”, *Proceedings of the IEEE International Symposium on Workload Characterization,* 2007.

[4] A. Verma, G. Dasgupta, T. Kumar Nayak, et al., “Server workload analysis for power minimization using consolidation”, In *Proceedings of the USENIX Annual technical conference,* 2009.

[5] Material Stock in German Data Centres, available at http://www.uba-green-it.de, 2010.

[6] L. Wang, G. Laszewskiy, J. Dayaly, et al., “Towards Thermal Aware Workload Scheduling in a Data Center”, In *Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms, and Networks,* pp. 116-122, ISBN: 978-0-7695-3908-9, 2009.

[7] E. Kalyvianaki and T. Charalambous, “On Dynamic Resource Provisioning for Consolidated Servers in Virtualized Data Centers”, *Proceedings of the 8th Int. Workshop on Performability Modeling of Computer and Communication Systems (PMCCS-8),* 2007.

[8] B. Speitkamp and M. Bichler, “A Mathematical Programming Approach for Server Consolidation Problems in Virtualized Data Centers”, *IEEE Transactions on services computing,* Vol. 3 (4), 2010.

[9] M. Marzolla, O. Babaoglu, and F. Panzieri, “Server Consolidation in Clouds through Gossiping”, *Technical Report. University of Bologna, Department of Computer Science,* 2011.

[10] J. Torres, D. Carrera, et al., “Tailoring Resources: The Energy Efficient Consolidation Strategy Goes Beyond Virtualization”, In *Proceedings of the International Conference on Autonomic Computing,* pp. 197 - 198, ISBN: 978-0-7695-3175-5, 2008.

[11] S. Srikantaiah, A. Kansal and F. Zhao, “Energy Aware Consolidation for Cloud Computing”. *Technical Report. Microsoft Research,* 2009.

[12] D. Barbagallo, E. Di Nitto, D. Dubois, and R. Mirandola, “A bio-inspired algorithm for energy optimization in a self-organizing data center”. *In Proceedings of the First international conference on Self-organizing architectures,* SOAR’09, pp. 127–151, 2010.

[13] J. Berral, I.Goiri, R. Nou, et al., “Towards energy-aware scheduling in data centers using machine learning”, In *Proceedings of the Int’l Conf. on Energy-Efficient Computing and Networking,* 2010.

[14] T. Cioara, I. Anghel, I. Salomie, G. Copil, D. Moldovan and B. Pernici, “A context aware self-adapting algorithm for managing the energy efficiency of IT service centres”, *Ubiquitous Computing and Communication Journal,* Special Issue of 9th RoEduNet International Conference, Volume 6 No. 1, ISSN Online 1992-8424 , 2011.

[15] I. Salomie, T. Cioara, I. Anghel, G. Copil, D. Moldovan, and P. Plebani, “An Energy Aware Context Model for Green IT Service Centers”, In *Post-proceedings of the first international workshop on services, energy, & ecosystem (SEE-ICSOC 2010),* Lecture Notes in Computer Science, Volume 6568/2011, 169-180, DOI: 10.1007/978-3-642-19394-1_18, 2011.

[16] G. Tesauro, N. K. Jong, R. Das, M. N. Bennani, “On the use of hybrid reinforcement learning for autonomic resource allocation”, *Cluster Computing* 10(3): 287-299, 2007.

[17] J. O. Kephart, H. Chan, R. Das, D. W. Levine, G. Tesauro, et al., “Coordinating Multiple Autonomic Managers to Achieve Specified Power-Performance Tradeoffs”, ICAC 2007.