Received: February 22, 2019

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Adaptive Autonomic Framework for Optimizing Energy Consumption in the Cloud Data Centers

Sara Diouani[1]* **Hicham Medromi[1]**

[1]*Research Foundation for Development and Innovation in Science and Engineering*
Engineering Research Laboratory (LRI), System Architecture Team (EAS),
National and High School of Electricity and Mechanic (ENSEM), HASSAN II University of Casablanca, Morocco
* Corresponding author's Email: diouanisara19@gmail.com

Abstract: Given the development of the Cloud Computing recently, clients and customers using the Cloud for both individual and business needs have expanded to an uncommon scale. This has normally prompted the expanded deployments of Cloud data centers over the globe. As a result, Cloud data centers are seen to be monstrous energy consumers and natural polluters. They require an extraordinary measure of regular energy which has made an effect on the energy supply and natural conditions of the environment. This is the reason why the vulnerability of persistent energy supply, later on, is being referred to. In this way, there is a need of an energy-aware cloud-based system which automatically and efficiently manages and optimize cloud computing data center resources by considering energy consumption as an essential Quality of Service (QoS) parameter. This paper, focus on the energy utilization of the data centers and how this can be limited so as to make the cloud computing greener. Thus, a new autonomic resource optimization manager has been proposed to avail the most optimum level of resources with reduced server energy consumption. The proposed framework has been verified theoretically and tested experimentally. The experimental analysis has demonstrated that the effectiveness of the proposed solution is greater than the state-of-the-art methods in terms of the achieved results related to reducing energy consumption and response time in cloud computing data centers.

Keywords: Cloud computing, Data center, Green cloud, Energy efficiency, Resource efficiency, Optimization, Autonomic management, MAPE-K, SLA, QoS.

1. Introduction

Energy effectiveness has turned into an increasingly important worry in the cloud computing data centers because of the issues related to energy consumption, including capital costs, working costs expenses, and ecological negative effects. Data centers at the center of Internet-scale applications expend about 1.3% of the overall power supply and this part is anticipated to be 8% by 2020 [1]. Google, for instance, expended 1.9 BkWh in 2010 which represent 0.8% of world data centers [2]. Also, in November 2008, carbon outflows from data centers were 0.6% of the worldwide total and are anticipated to be 2.6% by 2020, which is more than the total carbon emission of Germany [3].

This energy loss is engendered due to suboptimal use of facilities and equipment, and according to these statistics, reducing the energy consumption of data centers and making them work in energy-aware ways is a central aim in research related to data center management. So as to appropriate resources scheduling, the mapping of cloud workloads is obligatory to enhance QoS parameters like time, cost, energy consumption and so on. This is why many IT experts have built up multiple technologies aiming to reduce energy wastage and which still requires the design and implementation of an efficient energy-aware resource management system.

In fact, there is a wide scope of research efforts examining systems and approaches to reduce energy consumption. Comprehensively, research in the field

related to energy data center optimization could be classified as follows [4]:

1) Server level energy management: Taking preferred standpoint of several power/performance states characterized in components, such as CPU and memory. 2) Cluster level management: Using optimization and control techniques to reduce the number of required compute nodes in an executed application. 3) Virtualization: Reducing the quantity of active physical servers by multiplexing them as virtual machines (VM) so as to use less physical servers and exploiting turning off underutilized servers. Another side of virtualization is considering VM migration and consolidation based on thermal output. 4) Scheduling: Job scheduling that can take into consideration energy consumption criteria, for example, the temperature of servers, power costs and CO2 discharge in the case of a geographically distributed data center. 5) Using renewable energy and power sources. Research in each of these classifications is trying to address a section of the energy management of a complex distributed system, i.e., a data center. To date, little research includes all-encompassing methodologies that optimize data center energy consumption based on overall administering procedures.

Also, the complexity of recent data centers has driven specialists and researchers to explore manners by which autonomic strategies can be used for data center management. Autonomic supervisors monitor and manage resources so as to guarantee that the components they manage are self-configuring, self-optimizing, self-healing, and self-protecting (so-called "self-" properties).

Specifically, in this research, we propose an improved version of autonomic management framework for cloud computing data centers with a specific spotlight on making data centers environment more energy- aware and self-optimizing. Our emphasis is on characterizing the establishments of the core concepts, entities, and elements, connections, and algorithms for autonomic management systems which supports a scope of management configurations. The ultimate objective is to develop a management framework that would permit the data center administrator to a) define managed objects and parameters, b) and depend on the system to maintain itself automatically and optimally managed. Different management scenarios are implemented so as to evaluate and simulate the proposed energy-aware framework.

The motivation of this paper is to design an autonomic energy efficient framework for effective scheduling of resources which considers energy consumption as a QoS parameter. The main contributions of this paper are the following: (i) to propose an autonomic resource management approach for execution of heterogeneous workloads by considering the generic property of self-management, using the MAPE-K loop (ii) to reduce the energy consumption and response time (iii) to implement and perform the evaluation in a simulated cloud environment for clustered heterogeneous workloads. Experiments show that our proposed Framework outperforms the state-of-the-art solutions in terms of energy consumption and response time.

The rest of this paper is organized as follows: Section 2 provides a review of related works in the areas of data center energy consumption and autonomic computing. Section 3 gives an outline of the background in the field. Section 4 presents the architecture of our energy-aware autonomic framework by detailing its components. In section 5, the trial assessments of the management framework are outlined using distinctive illustration scenarios which are executed and assessed using our data center simulator to exhibit the effectiveness of the proposed system according to the obtained results. To conclude, the paper in the last section gives an outline of our commitments and their future aspiring expansions.

2. Related works

In light of the targets of our research, related works can be classified into two major classes. First of all, we review some previous research related to approaches for data center energy reduction. Afterward, we give a general view of research efforts in autonomic computing aimed to reduce energy consumption in cloud data centers. In fact, there are few studies which deal with monitoring in the cloud computing environment.

2.1 Review of research efforts in cloud computing data center energy reduction

There are three classifications of energy-efficient optimization techniques for cloud computing:

(1) Infrastructure-based optimization which manages infrastructural changes like making green structures utilizing energy-efficient equipment; data centers use raised floors and brought down ceilings for cooling air dissemination, with the processing hardware sorted out in rows of racks, regularly finished with cold corridors and hot passageways. Server racks may have chiller entryways that

function as radiators to cool down hot air coming out of servers. The cooling of the data center room is regularly done through computer room air conditioning units. In this situation, cool air comes into the data center through raised floor vents. Later plans have racks of computers cooled by fluids that are pumped through the racks, servers and even chips. Yet, these proposed solutions are costly to implement and give a limited reduction in energy consumption [5-7].

(2) The hardware-based optimization includes managing dynamic changes in hardware settings in servers to developing management structures for entire clusters of servers and whole data centers. These solutions involve Dynamic voltage–frequency scaling (DVFS) practices which are utilized on processing components for assisting the dynamic change of their performance to power consumption [8, 9]. However, multiple studies demonstrated that DVFS engender significant performance degradation at high utilization levels and higher response time [10].

Also, data center provisioning algorithms attempt to give the number of servers that guarantee the response time indicated in the Service Level Agreement (SLA), which frequently determines the maximum load. In practice, most of the times, data centers are worked far less than their maximum load, e.g. 30-70% of their maximum load which consumes power. In some research, they consolidate all the workload in a number of servers at a given time [11-13] and power off unused servers or at least make them work in low power mode. And consolidation can be performed at various levels: (a) VM consolidation, (b) server consolidation, and (c) task consolidation. And the consolidation involves the use of migration algorithms. In fact, by moving services to VMs, the CPU utilization of that physical machine (PM) is increased. Then, the number of running PMs can be reduced and this decreases energy consumption [14].

For energy saving, Han et al. [15] have proposed a VM placement algorithm, and an algorithm to find underloaded PMs to switch them to sleep mode. These researchers have applied the mix of these two algorithms in cloud data centers to complete the process of VM consolidation. In addition, VM placement algorithm can manage variable workload to prevent PMs from overloading after VM placement and to minimize the SLA violations.

Actually, most of the proposed methods and strategies focusing on energy efficiency improvement in cloud data centers and especially in Infrastructure as a Service (IaaS) principally focalize on managing computing resources. Those solutions are based on server consolidation techniques [16] and by switching the mode of idle resources to a power saving or to an operating mode.

In fact, in [17], the researchers select VMs to consolidate from the overloaded or underloaded host for migrating them to another suitable host and by considering CPU and RAM as primary energy parameters. And the idle hosts are turned into energy saving mode. Furthermore, researchers in [18], proposed an algorithm to select VMs to be migrated from overloaded hosts by considering CPU, RAM, and Bandwidth. After, the empty hosts are changed to the sleep mode.

Karakoyunlu et al. [19] proposed a method for allocation of resources based on metadata heterogeneity for cloud storage. In this solution, the inactive resources are changed to low energy mode so as to reduce energy consumption.

Researchers in [20] defined two algorithms which are Dynamic Resources Allocation (DRA) method and Energy Saving method. By the DRA algorithm, the waste of the idle resources on the VMs can be diminished. Furthermore, the Energy saving method diminishes the energy consumption of the cloud cluster. More precisely, 39.89% of total energy consumption is decreased (also for memory and VCPUs).

(3) Software-based optimizations include employing job scheduling algorithms in the application level of the data center, which have been broadly utilized for reducing energy consumption. Some researchers used heuristics as a base of a scheduling algorithm to map the task on the heterogeneous system while minimizing energy consumption [5].

Multiple methods for load balancing have been proposed, among them we cite [21]; In this technique, a Fruit fly optimization approach (EFOA-LB) in the cloud is used so as to balances the load among VMs, to reduce energy and response time in the data center, while adopting the Dynamic Threshold value along and the sleeping strategies. The results obtained after simulation uncover that this proposed methodology accomplishes more performance contrasted with some current techniques such as PSO algorithm. These researchers plan to extend the work with other QOS factors such as network traffic in the cloud.

Additionally, in this paper [22], a job scheduling algorithm is presented to assign a job to a VM of the current active hosts itself by considering job classification and preemption. Which restrains the number of hosts utilized in the allocation and reduce the energy consumption in the cloud datacenter. In

this mechanism, each job is characterized into three different types and assigned based on preemption policy with the earliest accessible time of the VM which is related to a host. Along these lines, less number of hosts in the dynamic state is made and the utilization of active host is increased and thus the energy consumption is reduced. Except that this solution focused on job scheduling field more than resource allocation.

In [23], the authors proposed an energy-aware task scheduling algorithm for high-performance computational tasks. This algorithm control dynamic energy consumption by the adoption of DVFS and it concentrate on working in multi-cloud systems and where the data centers are decentralized.

The energy efficiency research study to address the issue of VM placement and optimization in the cloud computing data center has demonstrated that appropriate planning and management of VMs and PMs in the cloud data centers reduce the total energy consumption; Consequently, we concentrated on green computing efforts to save energy at the infrastructure level in data centers and more precisely at server and VM levels.

2.2 Review of research efforts in autonomic computing aimed to reduce energy consumption

Autonomic Computing (AC) refers to the ability of a computing system or application to be self-managing, which means that the system can manage itself and can be adaptable to any changes and adjustments in its environment [24].

In AC, a management module which controls the conduct of a Managed Object (MO) is called an autonomic manager (AM). IBM at first introduced the idea of autonomic management and recommended that an AM consistently goes through a cycle of monitoring, analysis, planning, and execution steps [25]. The idea in AC is that distinctive AMs control diverse resources in a distributed way. This management could be done individually, i.e., each AM is in charge of its own MOs. More generally, in computing systems, it is fundamental that AMs interoperate and which could be heterogeneous sorts of them with various goals. The research introduced in [26] demonstrates the coordination between two autonomous AMs. In this work, the first AM manages SLA administration and resource allocation to reduce SLA violations. The second AM manages minimizing power consumption by turning off unused servers. This work demonstrated that without a connection between the managers there may be a failure to accomplish their objectives.

Also, other recent works such as [27], proposed an autonomic cloud computing solution which offers dynamic allocation and monitoring of resources dependent on VM migration. This system is SLA complaint and automates the user experience by respecting the conditions referenced in the SLA.

3. Background

In this section, we present the principal parameters for energy in cloud computing and we introduce the Autonomic Computing paradigm and maturity computing levels.

3.1 Important managed objects and monitoring parameters for energy in cloud computing

In this part, we aim at finding an initial set of managed objects and its set of possible monitoring parameters for energy in cloud computing. Also, we define the possible related actions to execute. As shown in Table 1, the majority of objects are described. In [28] and [29], we have defined all major energy parameters that influence the efficiency of the cloud data center energy, and specified some major distinct Service Level Agreement constraints of the VM placement.

3.2 The autonomic computing paradigm and maturity levels

Manual management of energy is a complex task, especially when the administrator has to consider the environmental changes and dynamic resources deployment. The Autonomic Computing [30] is a key paradigm which takes into account this dynamicity in an autonomous way.

1) The loop called MAPE-K of the Autonomic Computing paradigm: To incorporate self-optimization in our framework, a further extended Energy-aware Autonomic Resource Scheduling Technique is integrated in our solution, in which IBM's autonomic computing concept has been utilized to plan the resources automatically by optimizing energy consumption where the user using available interface, can easily interact with the system.

Table 1. Illustration of manageable objects and related monitoring parameters and actions

Object	Possible monitoring parameters	Type d'actions possibles
VM	-Throughput -CPU utilization	-Migration -Blocking
Server	-CPU utilization -CPU power state -Server queue job length	-Changing power state -Shutting server down; bringing it up -Invoking admission control
Cluster	-Node utilization -Number of waiting job	-Shutting down nodes -Workload manipulation
Rack	-Number of available nodes -Number of running applications -Number of running system	-Shutting down some nodes -Changing node CPU frequency is applicable
Application	-SLA violations -Number of active allocated servers -Percentage of idle servers -Number of jobs/requests in the application queue	-Make a number of serves idle -Activating a number of servers -Blocking application (stop running and just queueing workload) -Change frequency of allocated servers
Data center	-Current power consumption -Current electricity rate -Current temperature -Data center utilization	-Shutting down racks -Activate racks -Turning on racks

The Autonomic Computing paradigm depends on four fundamental segments distinguishing the MAPE-K loop to ensure the above properties of the framework. Ordinarily, control circles are actualized following MAPE (Monitoring, Analysis, Planning, and Execution) steps. Fig. 1 demonstrates these parts, which are presented as following:

-The M-onitoring of the managed resources and its current performances associated with its current configuration or QoS satisfaction based on SLA, with side effects detection in view of characterized rules;

-The A-nalysis of the produced indications with the intent to distinguish the possible causes of them, in view of stored data in the knowledge base, if changes are required, a demand for change is sent to the plan function;

-The P-lanning of the actions should have been set up with the objective to coordinate the targeted goals; it creates or select methodology /plans to institute on the managed entity;

-The E-xecution of the elaborated plans to change the conduct of the monitored element through the effectors.

The control cycle with its four segments and the Knowledge database allows the autonomic director to be self-manageable. The managed resources can be software or hardware resources including operating systems, wired or wireless network, CPU, database, servers, switches, routers, application modules, Web services or VMs, etc. [24].

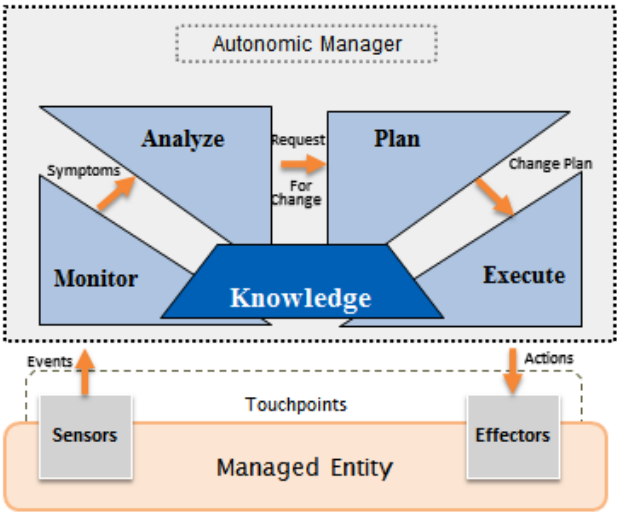

Figure. 1 The MAPE-K cycle of the AC paradigm

2) The maturity levels of the Autonomic Computing paradigm: Implementing an autonomic framework which can incorporate the MAPE-K loop is additionally a complex task since you have to go through five Maturity Levels (MLs) [30] for example, Fig. 2:

Level 1 – Basic ML: the administration and setting of the framework's components are done independently by the administrator. Human abilities are then needed to monitor the system, to examine the observed measurements and metrics and ultimately to execute actions relying upon on the detected anomalies;

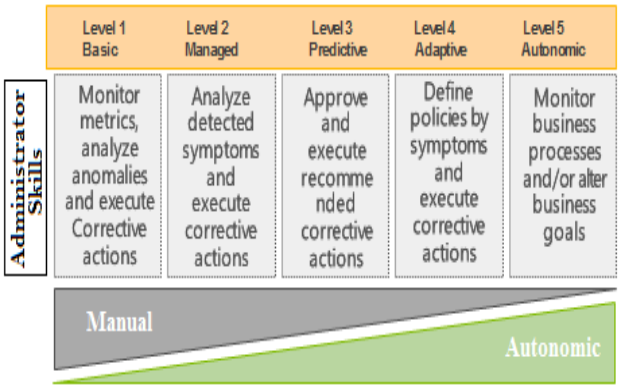

Figure. 2 Maturity levels towards autonomic system

Level 2 – Managed ML: so as to reduce energy, the monitoring tools and technologies can be used to collect metrics and synthesize information from the system. Human attitudes are needed to analyze the collected data and suitable actions;

Level 3 – Predictive ML: examination and analysis abilities are acquainted in the system to analyze the situations and give possible actions. Here the administrator is responsible for giving the final decision as well as the actuation of the actions;

Level 4 – Adaptive ML: the administrator must simply to characterize approaches in light of the correlation between side effects and mechanisms without the need to approve the corrective actions and to activate them. And consequently, the adaptive environment will automatically choose the adequate action in view of the accessible data and the knowledge of what is occurring in the environment system;

Level 5 – Autonomic ML: at the autonomic level, business approaches and objectives govern the autonomic manager with the consideration of applications requirements. The administrator collaborates with the autonomic manager to monitor business processes and modifies the targets if needed. Finally, the system becomes autonomic.

4. Autonomic and adaptive energy-aware resource manager framework

4.1 General presentation

In this part, we describe the design and the structure of our proposed Autonomic Energy-aware Monitoring framework in the cloud computing environment at the infrastructure level (IaaS). We depict its components in more profundity, and how they interact with each other.

The framework design depends on the IBM autonomic manager architecture [25]. It is composed into four fundamental modules which are Monitor, Analyzer, Planner, and Executer. These modules share the same knowledge and learning database and dynamically manage entities using sensors and effectors.

Also, the proposed Framework consists of actors and allocation techniques. These actors are users and IaaS providers. The system efficiently distributes cloud resources. This manager consists of a number of PM, which allocates VM.

For our situation, clients make a demand to allocate a VM in the framework. And when the client chooses to make this demand, the framework will allocate this VM to the corresponding PM, making the general framework as efficient as possible.

Fig. 3, presents the principal functional components of our framework and which are identified with the dynamic service monitoring. The monitoring module is in charge of managing and monitoring at runtime the energy consumption of the cloud data center by collecting data from all the available resources. It has likewise a knowledge database that stores this data which is required by the other components in the framework. Dynamic energy monitoring is actually an instance of an autonomic control cycle: monitoring, analysis, planning, execution whereby frameworks can monitor themselves and keep up an objective behavior.

4.2 Autonomic energy-aware resource manager

In this part, we detail the components of the Autonomic Energy-Aware Resource Manager Framework and how they collaborate with each other. Also, we describe the design and the implementation of this manager.

The monitor: receives as input, the events gathered from managed resource sensors and creates symptoms occurrences or reports as a response. The gathered monitored data include details about the actual resource utilization by the workflow applications and return a set of defined energy utilization states (Normal, warning, etc.).

The analyzer: processes the monitored data and gives the mechanism that correlates and model complex situation. These components allow the autonomic manager to learn about the environment and help predict circumstances. It provides an interface for receiving as input the monitored information as symptom occurrences from the monitoring phase. It checks if a demand for change should be created as a reaction by analyzing the

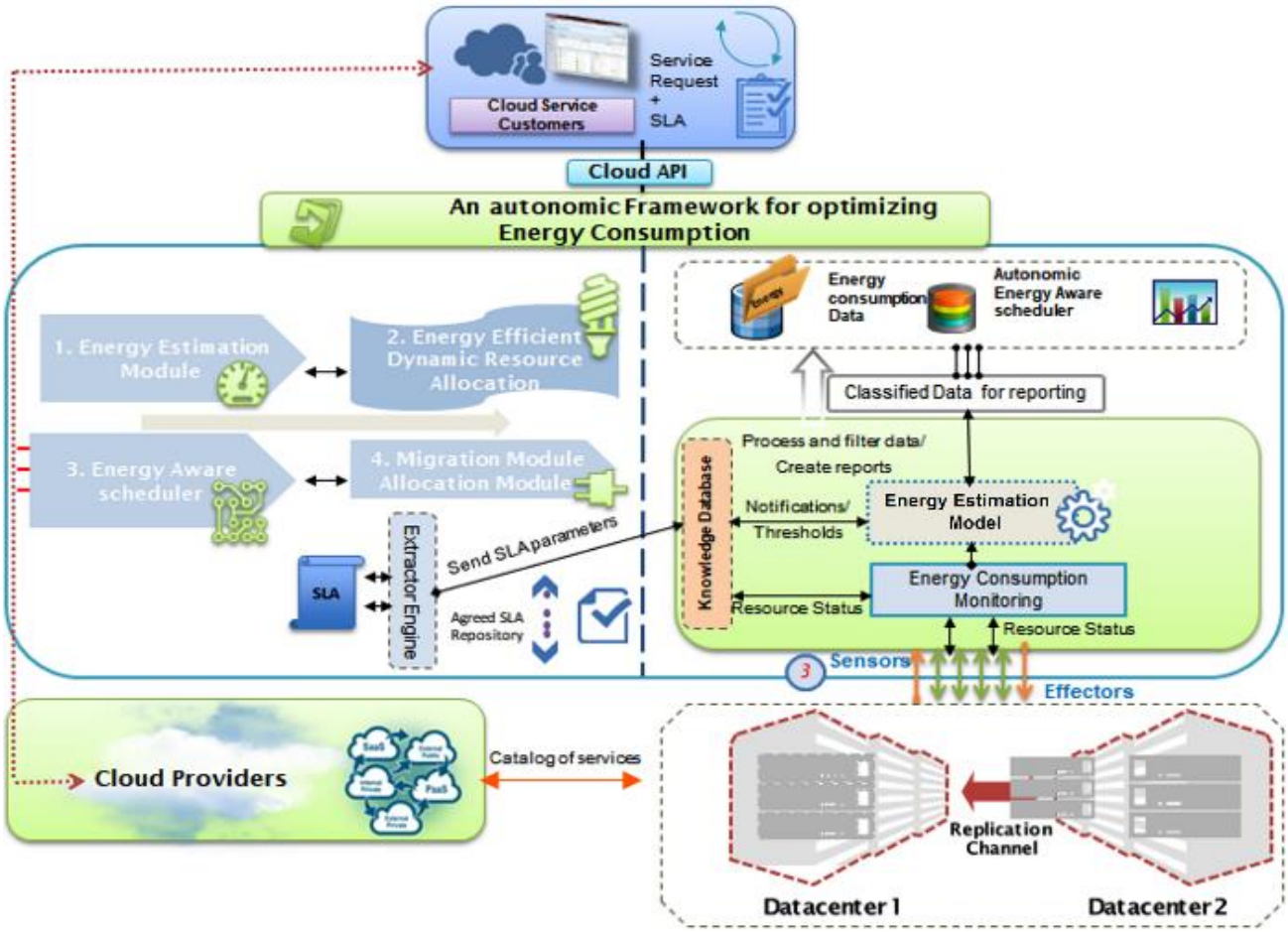

Figure. 3 Adaptive autonomic energy-aware resource manager framework

received information to determine the difference between the exact actual energy consumption and the threshold defined, and then decides on the exact reactive action (such as resource allocation, resource deallocation, and VM migration) to carry out in order to optimize resource utilization and reduce consumed energy.

The planner: provides the mechanisms that develop the action needed to accomplish goals. It exploits policies based on objective and environmental awareness to control its work. It receives as input requests for change and creates actions plans as a reaction.

The executor: By adopting managed effectors, this element takes as input a sequence of actions and performs it considering the dynamic updates.

1) Monitoring Energy Consumption Component: As shown in Fig. 3, the Monitoring module includes three sub-modules which are the Energy consumption monitoring, Energy Estimation model, and Knowledge Database.

Initially, as exhibited in Fig. 4, the sensors get the information about energy consumption of all the systems working under the cloud and update the information in real time, and after, this module is adopted to gather these data for checking persistently the value of resource utilization as shown in Algorithm 1.

After, the collected data will be transferred to the next module for further analysis (analysis component).

Where:

Ru: Set of information about resources utilization in defined time interval t;

ST: Set of Energy utilization states (Normal, warning, etc.).

These monitors may be in various format, depending on the type of usage metrics they are intended to gather and the way in which usage data should be gathered. Thereafter, we present an agent-based implementation format which is assigned to forward the collected data use to a log database for post-processing and reporting aims.

Algorithm 1: Monitoring Energy consumption phase

Input : List of hypervisors H (listH)
Set of information about resources utilization in time interval t : Ru
Set of defined threasholds of resource utilization: Minimal-threashold, Meduim-threashold, High-threashold, Maximal-threashold,
Output: Set of Energy utilization states: ST

Start

Set of available resources $listH = h_1, h_2, ..., h_n$
for *n=1 to n in listH* **do**
  if $(Ru < Minimal - threashold)$ **then**
    Generate alert(ST=ToEmpty);
  end
  else if $(Ru >= Minimal - threashold)$ *and* $(Ru < Meduim - threashold)$ **then**
    Generate alert(ST=Normal);
  else if $(Ru == Meduim - threashold)$ **then**
    Generate alert(ST=Notify);
  else if $(Ru > Meduim - threashold)$ *and* $(Ru < High - threashold)$ **then**
    Generate alert(ST=Warning);
  else if $(Ru >= High - threashold)$ *and* $(Ru =< Maximal - threashold)$ **then**
    Generate alert(ST=Critical);
  else if $(Ru > Maximal - threashold)$ **then**
    Generate alert(ST=Failure);
end
return (ST)

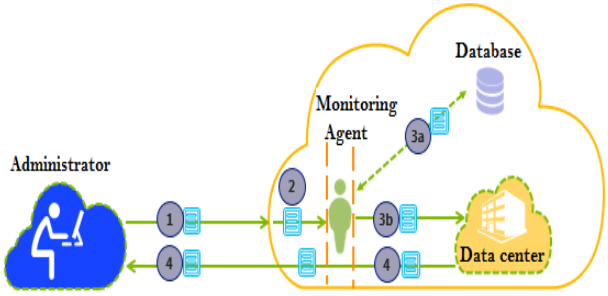

Figure. 4 The cloud administrator sends a request message to a cloud service (1). The monitoring agent intercepts the message (2) to collect relevant usage data before allowing it to continue to the cloud service (3b). While the monitoring agent stores the collected usage data in the log database (3a). The cloud service replies with a response message intercepted by the agent (4), it is sent back to the cloud administrator.

The Monitoring Agent is an event-driven program intermediate that works as a service agent and resides along existing communication paths to transparently monitor data-flows as shown in Fig. 4 where the agent plays the role of a sensor. So as to measure network traffic and message metrics, this type of cloud usage monitor is generally adopted.

The Monitoring Agent could monitor at various levels in the framework Cloud on each Compute Node:

Host OS/Hypervisor. Here the Monitoring Agent could catch all customary working framework measurements at the equipment level, such as percentage CPU time per VM, memory usage, disk I/0, and so forth.

Guest OS. Observing in this level enables the Agent to gather the particular data to one VM in a working framework.

Application Level. Instead of the past two levels, this necessitates the change in the application to give a monitoring interface whereby application- level execution measurements can be gotten.

The captured data are filed in a database that can be sent at runtime to the monitoring engine (ME) and may be utilized at various times. This ME may periodically apply a rule set from the resource information, given by the knowledge database, to verify if the resource usage is being abused, or requires any action.

The Monitoring Engine observes and analyze the runtime execution and performance of cloud services to guarantee that they are satisfying the optimal resource use. This system can pro-actively repair or fail-over cloud services when exception conditions happen, for example, when the MA reports that a resource usage of a hypervisor is bad.

An UML description of the functional

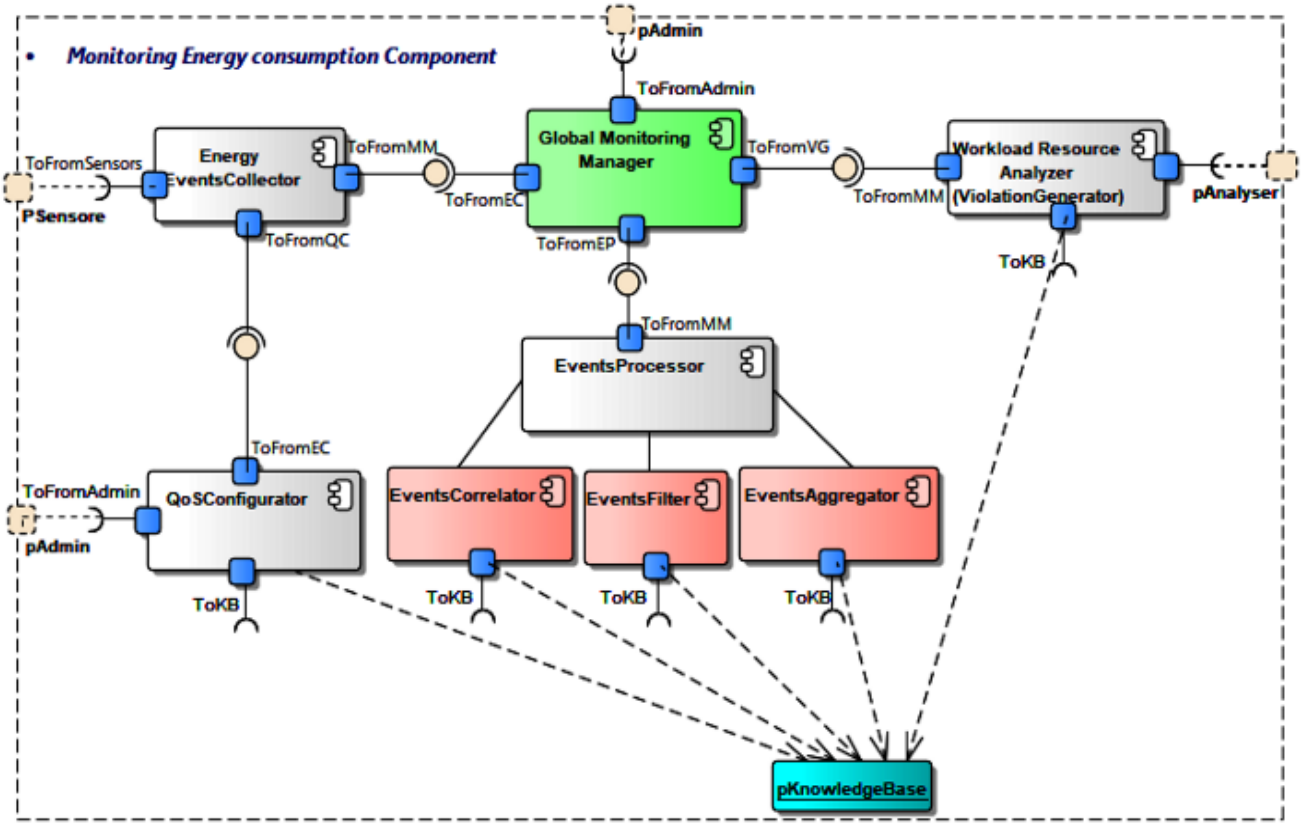

Figure. 5 Functional architecture of the monitoring energy consumption component

architecture of this monitoring engine is presented in Fig. 5, making evident the high-level functional components permitting the targeted maturity level.

The primary components of our Monitoring Energy Consumption Component are: The Global Monitoring Manager ensures the interface with the administrator, also, is in charge of the configuration and deployment of the different parts and the collaboration between them.

As Fig. 6 shows, the global monitoring Manager exposes four external interfaces. A simplified form of the web subsystem gives a web interface for the administrator, enabling them to interact and cooperate with the system (e.g. include new patterns, get alerts, and so on.). This created event patterns expression and EPL statements incoming from the administrative interface (at runtime configurations) are received by the REST interface which is exposed by the receiver service (API). Thus, these rules are transferred to the persistence service which stores them in the Knowledge database. Communication with receiver service and database utilizes the Java Database Connectivity (JDBC) interface overlaid by the Hibernate technology;

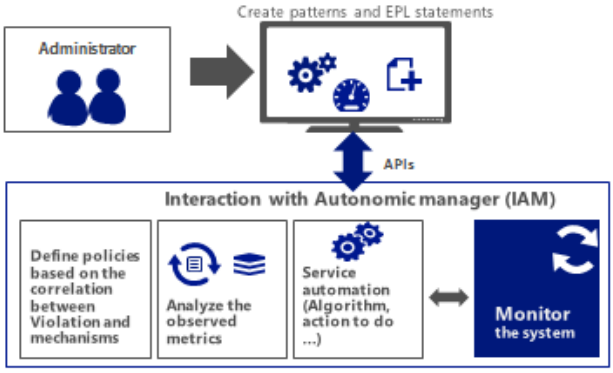

Figure. 6 Defined rules by administrator

The QoSConfigurator mechanism is intended to interact with the knowledge database (KB) for the retrieval of the agreed SLAs between cloud service providers and consumers, also contain the description of the appropriate sensors; After that, it collaborates with the Energy Events Collector for the specification of the metrics to gather depending on the agreed SLA;

The Energy EventsCollector is responsible for the collection of metrics at runtime of cloud services through interaction with the corresponding sensors. The description of this interaction is specified by the QoSConfigurator that asks the KB.

The interaction between Energy EventsCollector and sensors integrated into the managed entity can be done through periodic request/response way or through a listening way with notifications from sensors when a change. After data is initially collected, it can be used at different times. All of the monitored data is archived in a KB that can be queried by the Global Monitoring Manager;

The EventsProcessor it processes arriving events by applying defined patterns stored in the KB; it can be filtering (EventsFilter), correlation (EventsCorrelator) or aggregation (EventsAggregator) events. Additionally, the correlation means (link between the metrics), aggregation (group different level metrics) and filtering (selection of specific patterns) of collected data are needed to identify or predict more complex patterns that can be modal, temporal, etc. such as a sequence of response time values exceeding a defined threshold and following an increasing tendency. As soon as a pattern is detected, it notifies the Global Monitoring Manager.

The Workload Resource Analyzer (ViolationGenerator) component is invoked by the Global MonitoringManager when one or several events match a defined pattern. It then generates adequate violation correlated to non-satisfaction of QoS requirements via the catalog of violation included in the Knowledge Base. In our case research, we consider six sort types of states related to the considered resources utilization:

1) TOEMPTY: the state of the resource utilization is identified as TOEMPTY when the resource utilization is lower than a defined Minimal-threshold (to facilitate the task for the cloud administrator, we consider it equal to the value 10);

2) NORMAL: the state of the resource utilization is identified as NORMAL when the resource utilization is between a defined Minimal-threshold and a defined Medium- threshold (to facilitate the task for the cloud administrator, we consider it equal to the value 10 and 50 respectively);

3) NOTIFY: the state of the resource utilization is identified as NOTIFICATION when the resource utilization is equal to a defined Medium-threshold (to facilitate the task for the cloud administrator, we consider it equal to the value 50);

4) WARNING: the state of the resource utilization is identified as WARNING when the resource utilization is between a defined Medium-threshold and a defined High-threshold (to facilitate the task for the cloud administrator, we consider it equal to the value 50 and 70 respectively);

5) CRITICAL: the state of the resource utilization is identified as CRITICAL when the resource utilization is between a defined High-threshold and a defined Maximum-threshold (to facilitate the task for the cloud administrator, we consider it equal to the value 70 and 95 respectively);

6) FAILURE: the state of the resource utilization is identified as FAILURE when the resource utilization is higher than a defined Maximum-threshold (to facilitate the task for the cloud administrator, we consider it equal to the value 95).

Thereafter, the Monitoring Energy Consumption Component sends then the state of the energy utilization to the analysis component.

2) *Analysis Energy Consumption Component:* The fundamental objective of the Analyze step is to process and analyses the information about energy consumption of cloud system received from the Monitoring Energy Consumption Component and to relate this information in accordance with the knowledge base policies (QoS requirements of workload(s), and so forth.) so as to produce an analytical diagnosis (resources are provisioned, scheduled and executed).

Also, notification messages and alerts are dispatched to the customers/providers to alert about the situation, as shown in Algorithm 2.

In fact, the analysis component consistently checks the status of energy utilization. Three Reaction (3R) strategies are defined based on the received state:

-In the case of a state type failure, it dispatches the notification messages to the customers/providers to alert about the situation (not possible to add other resources. . .) and mark the host as “saturated” and define the action (move to another host).

-In the case of a state type Toempty, it dispatches the notification messages to the customers/providers to alert about the situation (resources must be moved to another host. . .) and mark the host as “empty” and define the action (move the resources to another host).

-In the case of the other states types: the analysis component consistently checks the workloads queued and energy consumption.

The aim is to provide the resources for the execution of heterogeneous cloud workloads by reducing the energy consumption. In other words, the workloads submitted should be executed with minimum energy consumption. Indeed, the workload submitted by the client to resource provider is stored into a crowd of workloads for their execution.

Algorithm 2: Analysis and Plan Energy consumption phase

Input : Set of Energy utilization states: ST
Output: Initiate the actions to be executed (Migration, Allocation, Consolidation, Switch on/off) : ACT

Start

Set of available resources $listH = h_1, h_2, ..., h_n$
Allocate resources to process data based on QoS requirements
for *work processing, calculate resource requirements Rr* **do**
  for *n=1 to n* **do**
    if $(ST == failure)$ **then**
      Generate alert(not possible to add other resources);
      if $(h_n! = saturated)$ **then**
        $(h_n = saturated)$
      end
      Generate alert(current host must be declared as dead node);
      ACT=DeclareHost(h_n,saturated);
      move on to next host to allocate resource;
      ACT=DeclareHost(h_n,saturated);
    end
    else if $(ST == toEmpty)$ **then**
      Generate alert(Resources must be moved to another host and verify if the host is marqued empty);
      ACT=Move this resources to another host;
      if $(h_n! = empty)$ **then**
        $(h_n = empty)$
      end
      Generate alert(The Host must be switched off);
      ACT+=Swith off this Host;
    end
    else if $(ST == Normal)$ *or* $(ST == Notify)$ *or* $(ST == Warning)$ *or* $(ST == Critical)$ **then**
      if $(Rr =< Pr)$ **then**
        Generate alert(Start execution of resources to process data);
        ACT=Allocate this resources in this host;
      end
      else if $(Rr > Pr)$ **then**
        Generate alert(This resources must be allocated in the next host from resource pool if resource is available);
        if $(E_{act}(t) =< E_{th})$ **then**
          ACT=Allocate this resources in the next host with min value of energy consumption of $(E_{act}(t))$;
        end
      end
      for *all resources* (h) *in* $(listH)$, *calculate* $(E_{act}(t))$ **do**
        if $(E_{act}(t) =< E_{th})$ **then**
          Generate alert(Continue the execution of resources to process data);
          ACT=Allocate this resource in this host;
        end
        else if $(E_{act}(t) > E_{th})$ **then**
          Generate alert(This resource must be allocated in next host);
          ACT=Allocate next host for the resource;
        end
      end
    end
  end
  Update-Knowledge-base();
end

The workloads are analyzed based on their QoS requirements and estimated value of energy consumption stocked in the KDB based on their previous statistics of execution. If the value of energy consumption of workloads executes within a defined range (less than the threshold value of energy consumption), then resources will be provisioned, on the other hand, an alert will be generated in order to notify for trigger another

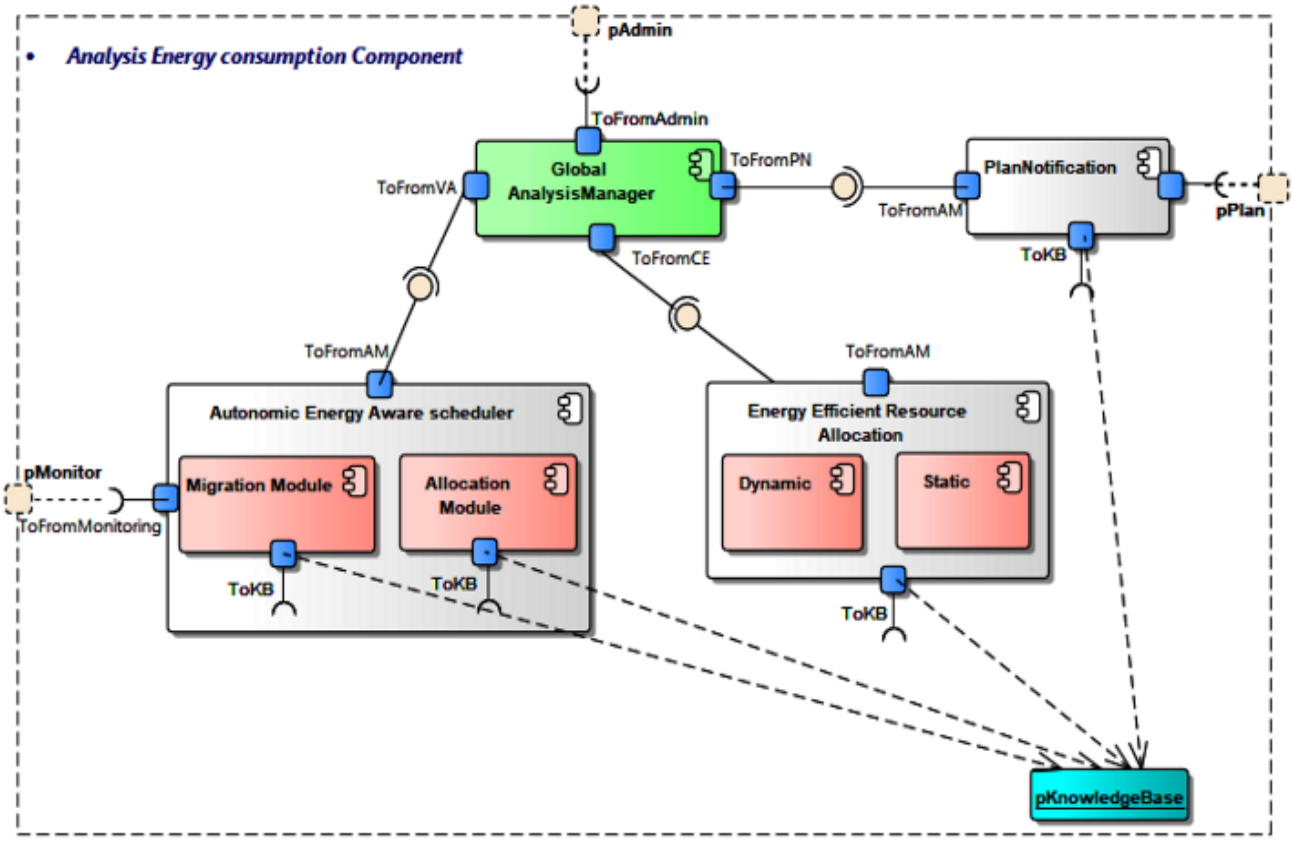

Figure. 7 Functional architecture of the analysis energy consumption component

analyses of the workload after reallocation of resources by the autonomic manager. Also, the system calculates the resource requirements to check whether the resources are sufficient for the execution of workloads is provided or not. If the sufficient resources are given, then the scheduling of resources for workload execution is started otherwise add new resources from the pool of reserved resources. Resources are another time assigned for further execution after finding the minimum value of energy consumption, as shown in Algorithm 2. The analysis component collects the information of available resources from the KDB which contains details of all the resources available in the resource pool and reserve resource pool. And based on cloud consumer details, it assigns resources and executes heterogeneous cloud workloads.

The Autonomic Energy Aware Scheduler engine processes the received resource utilization states. It provides an interface for receiving the monitored information from the monitoring phase. It analyzes and classifies resource utilization states depending on the priority and criticality (normal, warning, etc.). This mechanism presents the backbone of the Analyze component.

Also, the Autonomic Energy Aware Scheduler engine contains two subcomponents: The Migration Module, as illustrated in Fig. 7, and Allocation Module, where:

-Rr: represents the resource requirements;

-Pr: represents the resource provided;

-ACT: represents the actions initiated to be executed (Migration, Allocation, Consolidation, Switch on/off).

The Energy Efficient Resource Allocation is invoked by the Global Analysis Manager component. The goal of this engine is to automatically allocate resources either in a static or in a dynamic manner; On the basis of roles, policies on the KDB, resources will be allocated in a static way. And in a dynamic way by adopting our algorithm.

The Plan notification component it is invoked by the Global Analysis Manager to define and structure the set of actions to be executed when a request for change has been produced by the Analysis Energy consumption component in order to provide adequate scalability and QoS level. The actions are chosen in the Analysis phase in correspondence with the knowledge base. The next steps are planning the order and timing of the actions (Plan phase) and

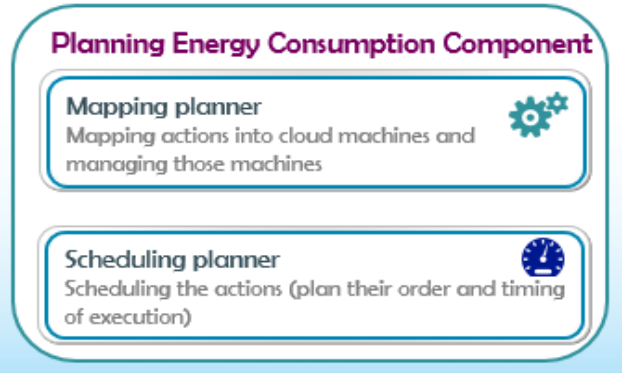

Figure. 8 Planning energy consumption component

finally executing them (Execution phase) with the help of actuators communicating with the self-management interface of the responsible services.

3) Planning Energy Consumption Component: plans the execution of the recommended actions and prevents changing impacts (i.e., allocating and deallocating the same resource reciprocally) in the tasks. This stage is divided into two steps: the mapping planner and the scheduling planner as Fig. 8 shows.

The first planner is in charge of mapping actions into PMs and VMs in data centers cloud computing and managing those machines. The scheduling planner is responsible for planning and arranging the order and timing of the execution of the actions.

4) Executing Energy Consumption Component: The execution phase is the final one. The executor executes the plan considering the dynamic updates. By adopting managed software effectors, this element takes as input a recommended sequence of actions (new workload submission, resource addition, alert generation, etc.) on the computational devices and execute it by using various sort of technologies (Web services, script file, Restful services, and so on.). Also, Effector is used to transferring the new policies, rules, and alerts to other nodes with refreshed data.

5) Knowledge Database: The term Knowledge Management (KM) in our context and following the approach means intelligent usage of measured data, obtained by monitoring, for the decision making process to satisfy application performance goal defined in SLA agreements while optimizing the computational resource usage and thus reducing cloud data center energy consumption. The core of the KM is a knowledge database (KDB) that interacts with these phases in the management process. This KDB integrates various policies (QoS parameter violation, failure, thresholds, algorithms for best cloud resource selection, SLA violation,

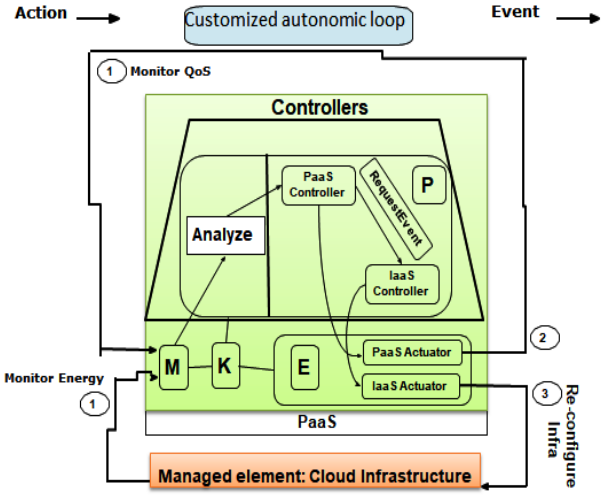

Figure. 9 Customized autonomic loop

etc..). Also contain the catalog of violations that will be used to guide the autonomic behavior (e.g. diagnostic models, the association of issues and specific corrective actions, etc.). The planning and executing steps of the MAPE-K loop will have to be explored in details in our remains future work, particularly when several adaptation mechanisms will have to be considered.

5. Evaluation

The goal of this evaluation is to validate the functional architecture of our Autonomic Energy-aware Monitoring framework by monitoring at runtime the energy consumption of the cloud data center and the proposed mechanism for an adaptive remedy rectification, to self-optimize a Cloud infrastructure to maximize power efficiency and performance while maintaining predictable and reliable behavior.

This experimental validation is based on the implementation of the autonomic energy consumption saving engine. We use a set of sensors to measure the real energy consumption and the set of agents to follow CPU, memory, disk usage per VM and per server as Fig. 9 shows.

Fundamentally, every compute node possess a software agent installed next to the hypervisor. This software is a customized OpenStack Ceilometer compute agent. It is a metrics collector, installed on each compute node of the datacenter [31]. This component polls metering data and instances statistics from the compute node through libvirt. Additionally, it monitors the host system, gathering various metrics on the node (e.g. CPU utilization, RAM usage, disk use). The compute agent forwards all the measurements to the central agent,

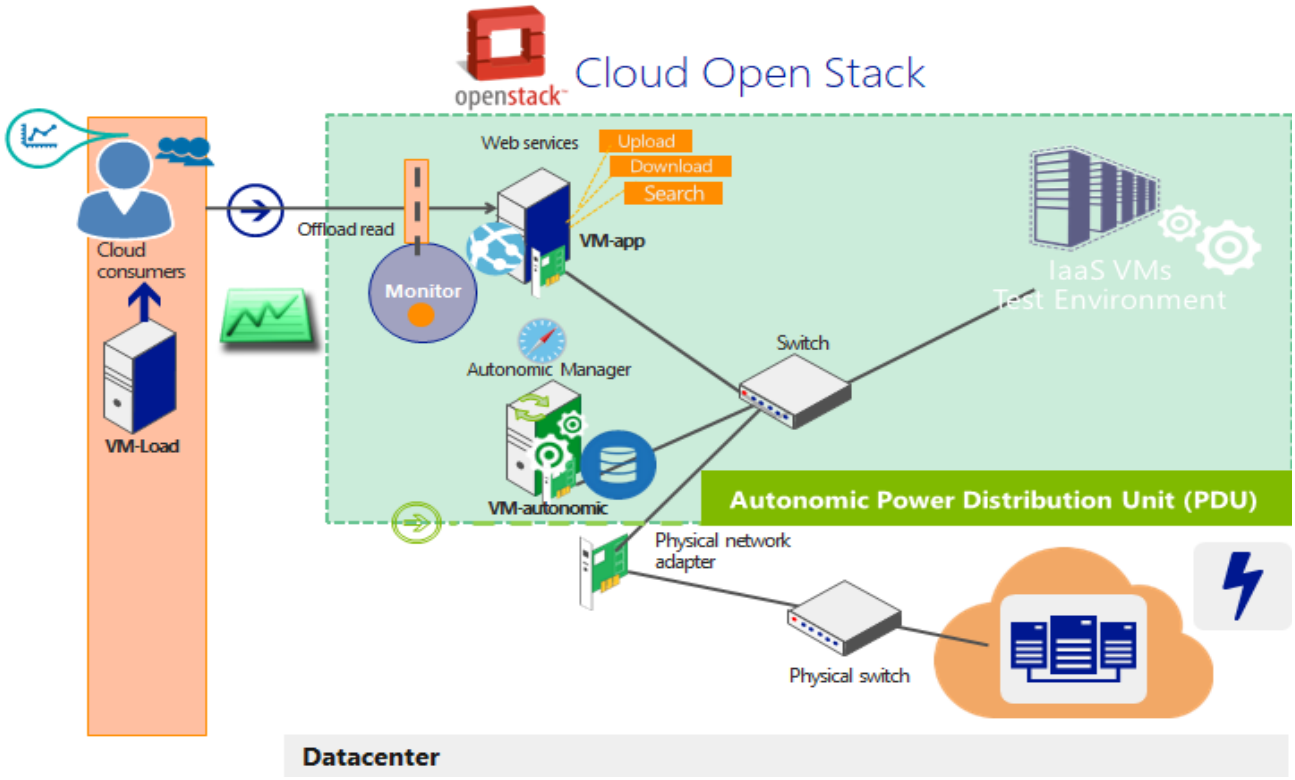

Figure. 10 Testbed

Table 2. Resource capacities of physical and virtual machines

Instance Type	System	CPU	Memory	Storage	Network
Host	Linux/Ubuntu	12-core 6-CPU 3.6 GHz Intel Xeon E5420 QC	64 GO	2 TO	Infiniband 20G Ethernet
Virtual machine	Linux/Ubuntu	4 VCPU 3.6 GHz Intel Xeon X3440	10 GO	500 GO	Infiniband 20G Ethernet
Cluster	Linux/Ubuntu	2-core 1-CPU 3.6 GHz Intel Xeon X5025	16 GO	700 GO	Infiniband 20G Ethernet

responsible for aggregating them. Continuously, this software agent feeds our autonomic energy consumption saving engine with the collected data.

These sensors allow us to associate an energy consumption value for a given physical server utilization (in terms of CPU utilization, RAM usage, disk I/O, network activity).

The dynamic detection of energy consumption related to the response time (Rt) and workload (wkd), the autonomic activation of our proposed mechanism, and then the observation of the induced benefits of this action. We first describe the experimental setup we have used in evaluating our approach, then we depict the test scenario of saving energy consumption, and we briefly describe the cloud services we have used in the experiments. At last, we talk about the results got from the execution of this test scenario.

5.1 Experimental testbed

In order to validate our proposed framework, we need to perform its experimental testing in a cloud environment as realistically as possible. In this section, we describe the testbed used in the experiment, as it is depicted in Fig. 10, we use Cloud OpenStack.

OpenStack is a distributed collection of open source cloud computing software projects that enterprises or cloud providers can adopt for creating, managing, and deploying infrastructure cloud services [32]. Table 2 shows the resource capacities of the physical and virtual machines being used in the proposed Cloud OpenStack testbed.

1) Infrastructure: We installed the KVM hypervisor on the server and created a large ubuntu-

Table 3. Energy savings

Approach	Energy consumed	Energy Saved (%)	Number of migrations	Response Time
AECS-F	461.13 Wh	50%	10	average(122 S)
ESM	579.37 Wh	9.24%	1	
DRA	518.53 Wh	21.83%	3	

4.04.5-server VM with a flavor occupying total server CPU and RAM available. Moreover, we installed libvirt-bin to ease the deployment and management of the VMs on the server. Then, we stress the virtual machine CPU for a range of given utilization level (from 0 to 100, with a step of 10). Note that, for some workloads, the CPU load starts at 10% due to the intensive disk write or memory filling.

During the evaluation, we consider four kinds of VMs: The first one, called vm-autonomic-controllers, hosts the autonomic manager of the system providing interfaces to monitor the energy consumption of the cloud data center and act on the system. The second one, named vm-load, is used for load injection purposes. It contains stress-ng [33] that is used to generate some variable load (first in terms of CPU and then, coupled with RAM and disk operations) and to analyze overall performance under different types of workload. The third one, named vm-app contains our developed bookstore web application, where an Apache server responding to the incoming HTTP requests is installed. The last one, called vm-KB is an instance for the database which back-up all transactions.

2) Prototype: This paragraph aims at presenting the prototype including autonomic manager of the system which provides interfaces to monitor energy consumption about the system state, analyze them and act on the system at runtime (as shown in Fig. 10). In order to evaluate system behavior under high load, we used a customized stress-ng [33] by introducing bean shell code adapted to our scenario for load injection to generate concurrent/ simultaneous requests towards our implemented VMs to simulate a load variation and to analyze overall performance under different types of workload, and report the measurements values.

5.2 Evaluation scenario

In our experiment, we compare our optimization autonomic energy consumption Framework with two greedy approaches Dynamic Resource Allocation (DRA) [34, 35] method and Energy Saving Method (ESM) [36] based on response time and energy consumption. Also, we simulated the proposed solution with four overloaded hosts detection algorithms and VM selection policy and which were described in [37].

These overloaded host algorithms are median absolute deviation (MAD), and interquartile range (IQR) with maximum correlation (MC), minimum migration time (MMT), and minimum utilization (MU) of VM selection policy.

How to effectively save and use the idled resources on low loading VMs, and save energy consumption on servers are the two issues we have to face and solve. There are two cases with/without our proposed Autonomic Energy Consumption Saving Framework (AECS-F).

Furthermore, every 4 minutes were considered as an hour, thus we calculated the energy consumption of 24 hours, impacted by each controller, which is presented in Table 3. Each experiment was run several times and we found the energy consumption difference between each run was 1& 2 watts. We have studied a scenario that consists of a single peak of workload: 5 threads are started and maintained for 15 min, whereupon we simulate a peak of workload passing from 5 to 20 threads. The observation interval and evaluation window were fixed in 4 seconds and 30 minutes, respectively. The threads sent measurements periodically. Such a high frequency of requests enabled evaluation of system behavior during peak load times.

5.3 Results and discussion

This section presents the results obtained from these experiments. The first part of the analysis for the simulation results brings into play the version of the approach that optimizes the two metrics named response time and energy. Fig. 11 shows energy consumption and response time results obtained for each simulation.

Overall we can notice that ESM gets results intermediaries with respect to AECS-F and DRA, in particular regarding consumption energy.

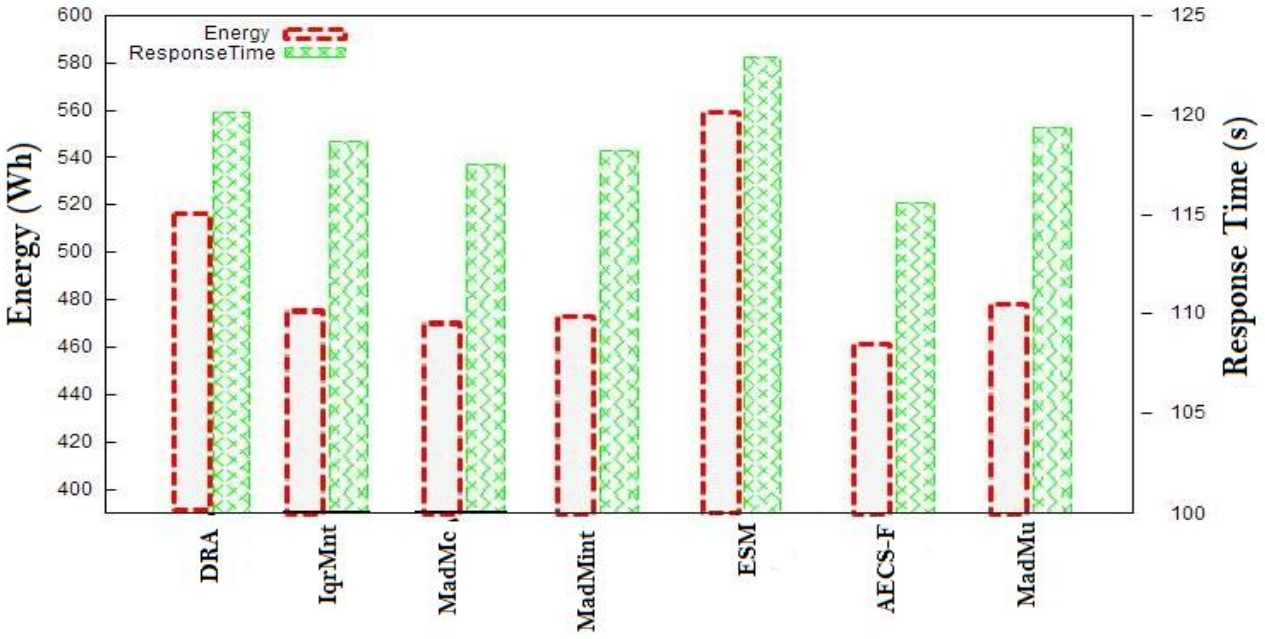

Figure. 11 Final value of energy consumption and response time during simulations of the different approaches

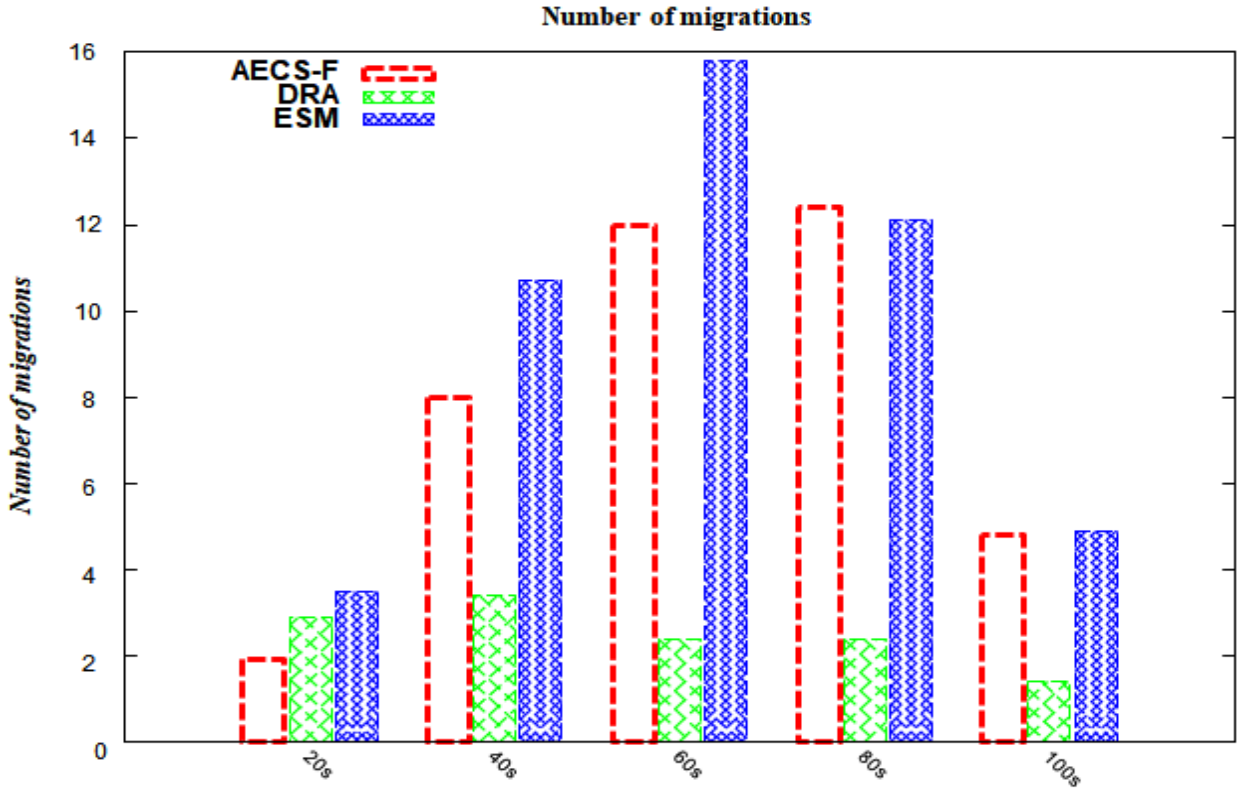

Figure. 12 Comparison of the number of migrations at each re-allocation moment generated by the three approaches

Indeed, the AECS-F is very effective in terms of VM consolidation minimizes very well the number of PMs used as described in Fig. 12. This allows him to obtain a very good energy consumption value of 461.13 Wh. Its response time result is also pretty good with a result of 116.12 s. Concerning these two metrics, the AECS-F allows for obtaining good results compared to other algorithms. ESM displays a substantially equivalent response time in 122.56 s but suffers from its inevitable tendency to use a large number of PMs which gives it the worst energy result with 579.37 Wh. DRA simulation with reallocations retains a fairly good response time of 120.54 s and intermediate power

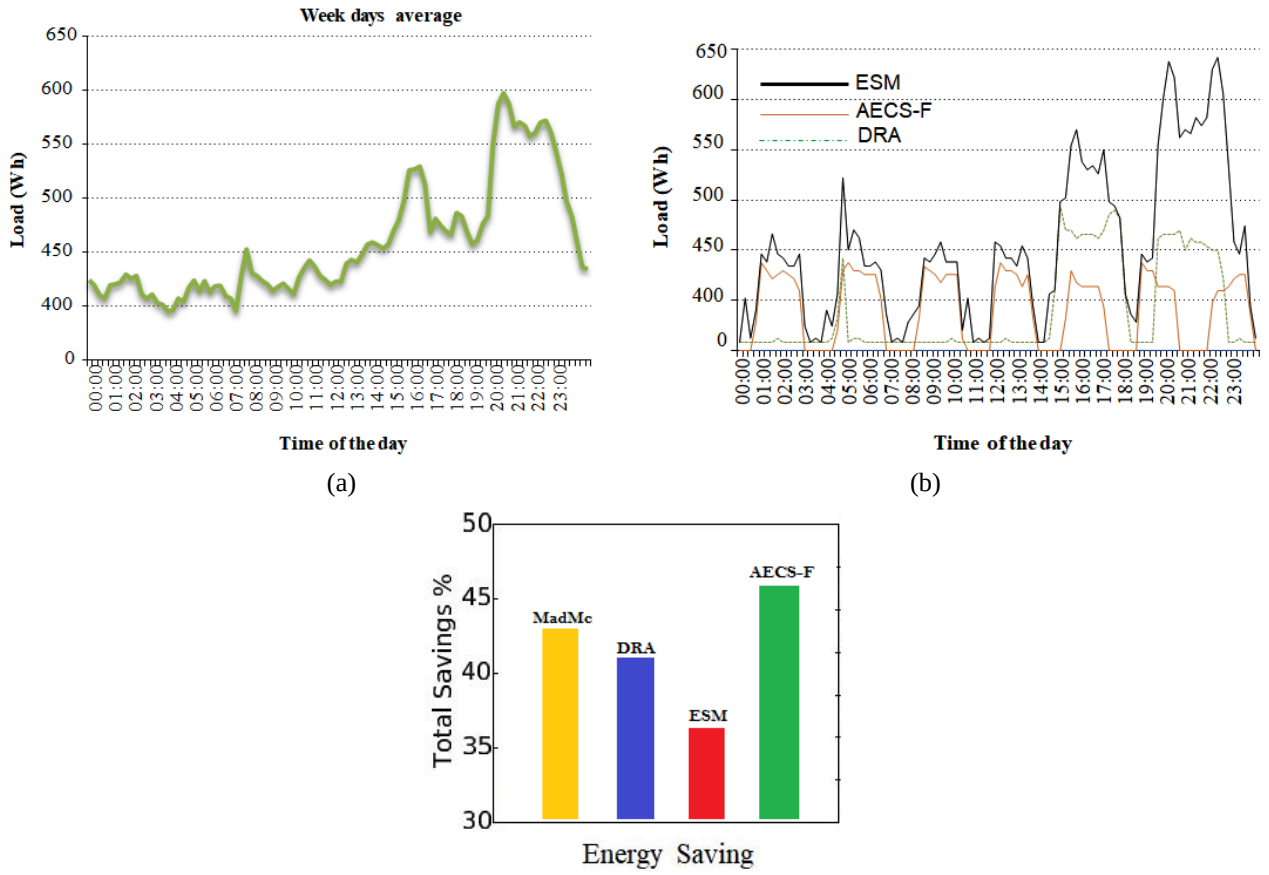

Figure. 13 The results obtained from the experiments with/without implementations: (a) average consumption on weekdays, (b) measured consumption on one day, and (c) Energy saving

consumption of 518.53 Wh. Thus, this result is worse than that of our AECS-F approach in terms of energy consumption.

According to the results obtained concerning energy consumption and response time after the simulation, we can see that the four algorithms IqrMmt, MadMc, MadMmt and MadMu have approximate results of 470 Wh, so we will focus on the following comparison of the algorithms on MadMc which represents the lowest values with ESM, DRA and our AECS-F solution.

All the measurements related to the energy consumption were sent to the Backend which was hosted in a Cloud environment built with VMs in a remote data center. In addition, we implemented a web-based Front end that provides the user with statistics over the collected sensor data.

Examples of charts reported are shown in Figs. 13 (a) and (b). Specifically, Fig. 13 (a) shows the average energy consumption during days of a week during simulations of the three approaches over a four-week period and Fig. 13 (b) drills down these approaches consumption for a more detailed analysis. After these comments, it appears that the optimization of this metric time response to a lot of impact on other metrics, see Fig. 13 (c): in a positive way for robustness and dynamism, but in a very negative way for energy. Such an optimization configuration is therefore only conceivable for a very low-load system, which will still limit the total consumption of the data center.

In summary, the results of executing the above scenarios evaluated some key features of our proposed autonomic SLA monitoring scheme, which are: its ability to monitor energy consumption in an efficient way and the ability to save energy.

6. Conclusion and perspectives

This paper has suggested the AECS-F as a new adaptive autonomic resource optimization manager framework to avail the most optimum level of resources with reduced server energy consumption. This manager considers major energy parameters and major possible constraints of VMs distribution in PMs. The primary target is to guarantee energy

efficiency by keeping energy-performance trade-off in concern while respecting the defined SLA. First, we a) defined managed objects and parameters, b) designed the system architecture by the adoption of the MAPE-K loop c) developed the scheduler algorithms d) implemented the proposed framework and performed the evaluation in a simulated cloud environment for clustered heterogeneous workloads.

Also, the experimental results indicated that the autonomic framework shows better energy efficiency and reduced response time.

For future work, we intend to enhance the autonomic framework in terms of proposals of different scenarios.

References

[1] J. Koomey, “Growth in data center electricity use 2005 to 2010”, *A report by Analytical Press, completed at the request of The New York Times*, Vol. 9, 2011.

[2] Q. Hardy, “Google says it will run entirely on renewable energy in 2017”, *The New York Times*, Vol. 6, 2016.

[3] W. Forrest, J. M. Kaplan, and N. Kindler, “Data centers: How to cut carbon emissions and costs”, *McKinsey on Business Technology*, Vol. 14, No. 6, pp. 4-13, 2008.

[4] F. Norouzi, “Coordinated autonomic managers for energy efficient date centers”, *Electronic Thesis and Dissertation Repository*, 3761, 2016.

[5] N. Khattar, J. Sidhu, and J. Singh, "Toward energy-efficient cloud computing: a survey of dynamic power management and heuristics-based optimization techniques", *The Journal of Supercomputing*, Vol. 1, No. 61, 2019.

[6] K. Lampka, B. Forsberg, and V. Spiliopoulos, "Keep it cool and in time: With runtime monitoring to thermal-aware execution speeds for deadline constrained systems", *Journal of Parallel and Distributed Computing*, Vol. 95, pp. 79-91, 2016.

[7] W. Zhang, Y. Wen, Y.W. Wong, K.C. Toh, and C.H. Chen, "Towards joint optimization over ICT and cooling systems in data centre: A survey", *IEEE Communications Surveys & Tutorials*, Vol. 18, No. 3, pp. 1596-1616, 2016.

[8] J. Liu and J. Guo, "Energy efficient scheduling of real-time tasks on multi-core processors with voltage islands", *Future Generation Computer Systems*, Vol. 56, pp. 202-210, 2016.

[9] Z. Lai, K. T. Lam, C. L. Wang, and J. Su, "Latency-aware DVFS for efficient power state transitions on many-core architectures", *The Journal of Supercomputing*, Vol. 71, No. 7, pp. 2720-2747, 2015.

[10] K. Choi, R. Soma, and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 1, pp. 18-28, 2005.

[11] H. Hallawi, J. Mehnen, and H. He, "Multi-Capacity Combinatorial Ordering GA in Application to Cloud resources allocation and efficient virtual machines consolidation", *Future Generation Computer Systems*, Vol. 69, pp. 1-10, 2017.

[12] F. Teng, L. Yu, T. Li, D. Deng, and F. Magoulès, "Energy efficiency of VM consolidation in IaaS clouds", *The Journal of Supercomputing*, Vol. 73, No. 2, pp. 782-809, 2017.

[13] E. Arianyan, H. Taheri, and S. Sharifian, "Novel heuristics for consolidation of virtual machines in cloud data centers using multi-criteria resource management solutions", *J. Supercomput.*, Vol. 72, No. 2, pp. 688-717, 2016.

[14] Y. Dong, L. Zhou, Y. Jin, and Y. Wen, “Improving energy efficiency for mobile media cloud via virtual machine consolidation”, *Mobile Networks and Applications*, Vol. 20, No. 3, pp. 370-379, 2015.

[15] G. Han, W. Que, G. Jia, and L. Shu, "An efficient virtual machine consolidation scheme for multimedia cloud computing", *Sensors*, Vol. 16, No. 2, pp. 246, 2016.

[16] V. R. Reguri, S. Kogatam, and M. Moh, "Energy Efficient Traffic-Aware Virtual Machine Migration in Green Cloud Data Centers", In: *Proc. of IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, pp. 268-273, 2016.

[17] R. Yadav, W. Zhang, H. Chen, and T. Guo, "MuMs: Energy-Aware VM Selection Scheme for Cloud Data Center", In: *Proc. of the 28th International Workshop on Database and Expert Systems Applications*, pp. 132-136, 2017.

[18] M. A. Khoshkholghi, M. N. Derahman, A. Abdullah, S. Subramaniam, and M. Othman, "Energy-Efficient Algorithms for Dynamic Virtual Machine Consolidation in Cloud Data

Centers", *IEEE Access*, Vol. 5, pp. 10709-10722, 2017.

[19] C. Karakoyunlu and J. A. Chandy, "Exploiting user metadata for energy-aware node allocation in a cloud storage system", *J. Comput. Syst. Sci.*, Vol. 82, No. 2, pp. 282-309, 2016.

[20] C. Chen, P. Sun, C. Yang, J. Liu, S. Chen, and Z. Wan, "Implementation of a Cloud Energy Saving System with Virtual Machine Dynamic Resource Allocation Method Based on OpenStack", In: *Proc. of the Seventh International Symposium on Parallel Architectures Algorithms and Programming*, pp.190-196, 2015.

[21] M. LawanyaShri, S. Subha, and B. Balusamy, "Energy-Aware Fruitfly Optimisation Algorithm for Load Balancing in Cloud Computing Environments", *International Journal of Intelligent Engineering and Systems*, Vol. 10, No. 1, pp. 75-85, 2017.

[22] S. Loganathan, R. D. Saravanan, and S. Mukherjee, "Energy Aware Resource Management and Job Scheduling in Cloud Datacenter", *International Journal of Intelligent Engineering and Systems*, Vol. 10, No. 4, pp. 175-184, 2017.

[23] A. Alsughayyir and T. Erlebach, "Energy aware scheduling of hpc tasks in decentralised cloud systems", In: *Proc. of the 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pp.617-621, 2016.

[24] A. Maarouf, Y. Mifrah, A. Marzouk, and A. Haqiq, "An autonomic sla monitoring framework managed by trusted third party in the cloud computing", *International Journal of Cloud Applications and Computing*, Vol. 8, No. 2, pp. 66–95, 2018.

[25] R. Kettimuthu, Z. Liu, I. T. Foster, P. H. Beckman, A. Sim, K. Wu, and A. N. Choudhary, "Towards Autonomic Science Infrastructure: Architecture, Limitations, and Open Issues", In: *AI-Science@ HPDC*, pp.2:1-2:9, 2018.

[26] K. Indira and M. Thangavel, "Green cloud computing", *Cloud Computing Technologies for Green Enterprises*, IGI Global, pp.114–136, 2018.

[27] R. Tomar, A. Khanna, A. Bansal, and V. Fore, "An architectural view towards autonomic cloud computing", *Data Engineering and Intelligent Computing*, pp. 573–582, 2018.

[28] S. Diouani and H. Medromi, "Towards an optimized energy consumption of resources in cloud data centers", In: *Proc. of the International Symposium on Ubiquitous Networking*, pp. 179-185, 2018.

[29] S. Diouani and H. Medromi, "Green cloud computing: Efficient energy-aware and dynamic resources management in data centers", *International Journal of Advanced Computer Science and Applications*, Vol. 9, No. 7, pp. 124-127, 2018.

[30] R. Sterritt, M. Parashar, H. Tianfield, and R. Unland, "A concise introduction to autonomic computing", *Advanced Engineering Informatics*, Vol. 19, No. 3, pp. 181-187, 2005.

[31] V. Cima, B. Grazioli, S. Murphy, and T. M. Bohnert, "Adding energy efficiency to openstack", *Sustainable Internet and ICT for Sustainability*, pp.1-8, 2015.

[32] T. Pflanzner, R. Tornyai, B. Gibizer, A. Schmidt, and A. Kertesz, "Performance analysis of an openstack private cloud", In: *Proc. of the 6th International Conference on Cloud Computing and Services Science*, pp.282-289, 2016.

[33] "stress-ng," http://kernel.ubuntu.com/cking/stress-ng/, 2018.

[34] A. Wolke, B. Tsend-Ayush, C. Pfeiffer, and M. Bichler, "More than bin packing: Dynamic resource allocation strategies in cloud data centers", *Information Systems*, Vol.52, pp.83-95, 2015.

[35] M. B. Nagpure, P. Dahiwale, and P. Marbate, "An efficient dynamic resource allocation strategy for vm environment in cloud," In: *Proc. of the International Conference on Pervasive Computing*, pp.1-5, 2015.

[36] C. T. Yang, J. C. Liu, K. L. Huang, and F. C. Jiang, "A method for managing green power of a virtual machine cluster in cloud", *Future Generation Computer Systems*, Vol. 37, pp. 26-36, 2014.

[37] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers", *Concurrency and Computation: Practice and Experience*, Vol. 24, No. 13, pp. 1397-1420, 2012.