An Architectural Framework for Enforcing Energy Management Policies in Cloud

Marwah M. Alansari
School of Computer Science
University of Birmingham
Birmingham,UK
Email: mma809@cs.bham.ac.uk

Behzad Bordbar
School of Computer Science
University of Birmingham
Birmingham, UK
Email: B.Bordbar@cs.bham.ac.uk

***Abstract*—Management of energy consumption in Cloud has recently received considerable attention. Most existing research focuses on designing algorithms for dynamically managing the running of virtual machines in Cloud, such as placement and migration algorithms. Despite the use such on-line algorithms being essential, another equally important dimension that is related to High-level policies and guidelines that are set by Cloud Mangers or Administrators for management energy consumption. Such policies often stem from business, legal and financial requirements. Currently, most implementations of High-level policies such as Management Energy Consumption Policies are done manually via the use of low-level programming languages and APIs for accessing Cloud interfaces. Since High-level policies can change frequently, the manual implementation for such policies increases the cost and the time of the development and maintainability. Thus, there is a clear need for a methodical way of executing High-level policies automatically in Cloud. In this paper, we propose a generic architectural framework for enforcing High-level policies particularly Management Energy Consumption Policy in Cloud via using a Business Rule Engine. The generic architecture is implemented to execute Energy Management Business Rules to fire management actions in OpenNebula cloud environment.**

Keywords*—*High-level Policies; Business Rule Engine, Business Rule, Generic Architecture ; Management Energy Consumption; Cloud Platform; Cloud Management System

I. INTRODUCTION

Management of energy consumption in Cloud has recently received considerable attention [1] [2] [3] [4]. Belgzanov and Buyya [1] presents a method using fixed and dynamic thresholds to deal with efficient consolidation of virtual machines. Li et. al [2] present multi-objective algorithms research on development of the algorithms and techniques for placement and rearrangement of virtual machines in order to reduce energy consumption [5] [6] [7]. Borgetto et.al apply a rule-based approach for reconfiguring and migrating virtual machines between physical hosts [8]. Most existing research focuses on the algorithms and dynamic management techniques for dealing with the infrastructure that is running the Cloud data centres.

Another equally important dimension regarding the management of the energy in Cloud relates to High-level policies and guidelines that are set by Cloud Mangers or Administrators. Such policies often stem from business, legal and financial requirements. An example of High-level policies is Energy Consumption Management Policy for the migration of virtual machines amongst physical hosts in the Cloud-platform [8]. Another example is migrating services between countries to benefit from the cheap energy at off-peak time. These policies are set by non-technical experts. There is a gap between Low-level techniques used at data-centres for the management of energy consumption and High-level policies that are set by managers. Indeed, the High-level policies should be converted and implemented via Low-level techniques, which can be executed in Cloud.

Currently, business managers decide on the policies, which they then communicate with the IT teams who implement them. This manual process increases the cost and causes delay, especially when such policy changes are frequent. This paper will present a generic architectural framework for enhancing the Cloud manager so that such High-level policies can be enforced automatically. To do so, we make use of a Business Rule Systems [9] which executes the policies through interaction with the Cloud manager. The suggested approach has been implemented with the help of Drools Business Rule Engine, which is integrated into the Cloud management system running OpenNebula.

The structure of the paper is as follows: Section II consists of the preliminaries which are related to Cloud management system , a description about Business Rules and an overview of energy consumption strategies in Cloud. In Section III, we present the running example that we used throughout the paper. We explained the study problem in Section IV. In Section V, we present our proposed solution and more details relating to the solution. Furthermore, we integrate our architectural framework using Drools and OpenNebula in Section VI. In addition, we outline a sample of expressing policies as rules that would run via the generic architecture in Section VII. Finally, we discuss related work in Section VIII.

II. PRELIMINARIES

A. Cloud Management Systems and OpenNebula

Cloud data centres naturally tend to deliver a vast amount of services to a large number of users. Thus, manual management for such an environment is difficult. There are a number of Cloud management systems such as Eucalyptus [10], OpenNebula [11] [12], and oVirt [13]. OpenNubula is a centralised management system which is used to deliver three common Cloud IaaS deployment models. Those models are; Private, Public and Hybrid Cloud. OpenNbeula supports

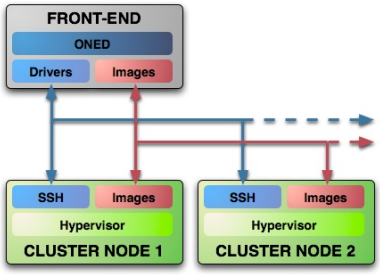

Fig. 1. The Deployment Architecture of OpenNebula Cloud Management System

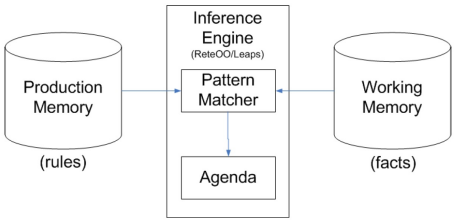

Fig. 2. The Architecture of OO-RETE Engine in Drools

the management of the heterogeneous cloud environment by offering several drivers. Those drivers can be connected to three different hypervisors which are; Xen [14], KVM [15], and Vmware [16].

Fig.1 represents the system architecture of a small cloud that uses OpenNebula. OpenNebula consists of a global manager component, which is called ONED Management Daemon; this is deployed on the Front-End node. ONED Daemon controls a set of worker physical nodes, which are known as Cluster Nodes. Each Cluster Nodes runs a hypervisor that deploys virtual machines images. The communication drivers in the ONED Daemon implements a Secure Shell(SSH) network protocol for securing the data transmission amongst Cluster Nodes. In addition, ONED Daemon uses image repository to make the virtual machine images accessible when being used with any suitable technology such as [11] [12]. The management is done by generating all interactions and actions to all of the virtualised environments in both in-house Cluster Nodes, and external Public Cloud Images. Examples of interaction actions are; placement of a new virtual machine or, migrating or stopping the execution of virtual machines. The architecture of OpenNubula can scaled by organising the deployment architecture into a hierarchical structure [12].

B. Rule-Based System and Business Rule Engine

Ian Graham defines a business rule as "*a compact, atomic, well-formed, declarative statement about an aspect of a business that can be expressed in terms that can be directly related to the business and its collaborators, using simple unambiguous language*" [17]. There are different types of business rules, which are; Assertion, Action, Procedure, and Constraints rules. Assertions rule have the form of *X is Something* [17]. For Example, Energy Consumption for Host1 is 500 Watts per hour. While Action rule has the pattern of: *If X happens then do action Y* [17]. An example that demonstrates an Action rule is if the Cloud Physical Host is idle and the time is night then Switch-Off Cloud Host. Procedure rules are forms of rules that provide instructions or plans to do something [17]. For instance, Monitoring Energy Consumption in the Cloud Cluster, then getting the Cloud Cluster Information, then sending a request to the sensor to communicate with the Cloud Manger, then updating the Energy Consumption Data. The final type of business rule is a Constraint rule, which consists of some comparative statements [17]. An example that can represent a constraint rule is When Energy Consumption in the Cloud Cluster is high and Lower-loaded machines are above 50%, then Migrate the Virtual Machines and Switch-Off the Lower-loaded hosts.

The Business Rules Management System (BRM) is a system that simplifies the development and maintenance of business policy. BRM stores various business policies as a set of sub-rules in a repository. BRM can allow policy to be formed in chain-able collections of rules and provides a higher level of graphical language to express the execution of rules as processes such as using BPMN [18]. Therefore, policies implemented using BRM can be developed and expressed in simplified form via using declarative language such as Drools Declarative Language. Furthermore, BPM can execute developed policies using Business Rule Engine such as Drools [17] [18].

Drools is an implementation used with the enhanced version of RETE-Algorithm for supporting Object Oriented Pattern [18]. RETE-Algorithm was introduced by Charles Forgy; it is one of the efficient implementation rules inference engines. The Algorithm represents rules as an acyclic graph to form a RETE-network and provides a pattern matching process [19] [20].

Fig.2 illustrates the implementation architecture of RETE-Algorithm, which is used in Drools. Drools Rule Engine requires the inclusion of two main inputs, which are; Rule Base and Working Memory. Rule Base is a long-term memory in which rules are stored such as Drools Rule File. Working Memory is a type of short-term memory which, contains Facts that need to be evaluated by the inference engine. Facts are object models or the instances that contain attributes that illustrate a domain data for an application. Agenda is the place where a rule that has become an active is stored for later in order to fire satisfied rules. The agenda uses resolving conflicting methodology for ordering the execution of active rules [19] [18].

C. Strategies for Management Energy in Cloud Platform

The introduction of Virtualisation technologies has contributed to energy saving in the Cloud environment. Virtualisation allows the running of multiple instances on a single host, which can increase the physical host resource utilisation level. As result of virtualisation, the number of running physical hosts in data centres can be reduced which may lead to energy being saved [5]. Therefore, Resource Consumption has become a measuring metric, which can be used as constraints for lunching Energy Management Policies [5].

To the best of our knowledge, there are two common tech-niques for saving energy at the management level in a virtualised environment: Dynamic Switch ON/OFF [5] [6] and

Dynamic Vertical Scaling [7]. Dynamic ON/OFF is applied with the objective of reducing the number of running hosts in virtualised environment. Applying the Dynamic ON/OFF method process requires the use of a virtual machines migration operation [6] which is a process of transferring virtualised services from one host to another. The migration process can be done either off-line (suspend/resume) or on-line (live-migration). The former method terminates the running virtual machine and then moves the virtual machine to the intended destination, after that, the virtual machine execution is resumed. In On-line migration, virtual machines are transferred to the destination without stopping the execution [5] [6] [8].

There are various implementations for Dynamics Switch ON/OFF approach. One of the implementations is suggested in [1] [8]. The suggestion includes defining fixed or dynamic thresholds for host resources, which can be used as a measurement of host consumption level such as CPU thresholds. The thresholds approach can be combined with various local search algorithms [2] [3] [21] [8] to find proper destinations for migration services and switch-OFF any hosts that are receiving lower workloads. One of the disadvantages of using the Dynamic ON/OFF approach is the continuous demand for services migration which, can affect the overall performance of the Cloud platform [5]. As a result, the process of implementation energy consumption policies, through modifying the underlining system, is both costly and time consuming.

The Dynamic Vertical Scaling mechanism is another method that can be considered as an energy management strategy [7]. Dynamic Vertical Scaling is referred to as Dynamic CPU Voltage Scaling in [7]. Vertical Scaling can be referred to as the method that can automatically change the virtual resources requirements for all running virtual machines, based on changing the resource demand of the virtual machine. For example, let us say that the maximum requirement for running a virtual machine X on host Y is to use 2 vCPU and 1GB vMemory during peak time. On the other hand, the minimum requirement is to use 1VCPU during off-peak time.

The implementation of the Dynamic Vertical Scaling approach does not require the use of a migration process [5] [8]. There is a need to define some thresholds for resource consumption levels, which can be considered as triggers for applying the strategy. In addition, there is also a requirement to define the duration time for using the minimum requirement. In [8], the approach is implemented by using a rule-based approach which, uses a rule scheme for reconfiguration. The scheme uses the current resources usage as well as predicted resources usage values as constraints which gives rise to the need to set configurations of the virtual machines to use another policy mode. The Dynamic Vertical Scaling method also can introduce some performance overheads due to the lack of using a mechanism to control the execution of the strategy [5]. Thus, Dynamic Vertical Scaling is similar to the Dynamic ON/OFF; it needs to be managed properly.

III. A RUNNING EXAMPLE

In this section, we shall outline the running example that is used throughout the paper in order to help to explain our approach.

A. Description of The Example

Company X uses a Cloud platform management system for controlling IaaS Cloud environment. Company X wants to implement a High-level Management Energy Consumption Policy (MECP) which uses both Management Energy strategies mentioned in Section II for controlling energy for running virtual machines images as well as physical hosts during oper-ating time. The deployment architecture of the Cloud platform of Company X is a typical Cloud-based architecture that consists of a master node, a number of physical hosts as well as virtual machines images (See Section II).

Assumptions: we defined a number of assumptions for simplifying the implementation MECP related to Company X. Firstly, we assume that the measurement of resource consumption metric for each physical host and running virtual machines will be based on both the average percentage of CPU usage and Memory usage. Secondly, Company X's Cloud platform environment uses the same virtualisation software. The objective is to reduce the complexity of virtual machines migration operation and to avoid software incompatibility is-sues. Therefore, Company X has a homogenous Cloud platform. The final assumption is that there are no rules to restrict the migration operation of virtual machines. This is to simplify our explanation and does not affect the generality of our method.

B. The Measurement Metrics and Thresholds Defined For The Example

There is a collection of measurement metrics that are used as constraints for MECP policies. We assume that the defined metrics represent an extension of fixed utilisation thresholds proposed in [1]. Since the Cloud platform is divided into clusters, MECP policies are applied to each cluster in Cloud platform. We assume that Company X has a number of monitors for collecting and measuring defined metrics which are related to MECP at a specific time. Therefore, the main measurement metrics are specified in as the following way:

1) **Resource Consumption** is the value of the CPU-usage and Memory-usage for a single host in the Cloud cluster.
2) **Cluster Resource Consumption** is the total value of host resource consumption for all physical hosts in the Cloud cluster.
3) **Cloud Cluster Energy Consumption** is the total Power Consumption [7] of all physical hosts in a single cloud cluster.

Each one of those metrics is specified with thresholds that indicate when the defined metrics are considered is at High, Normal or Low level.

C. A Sample of Management Energy Consumption Policy

MECP, which Company X wants to implement, consists of a number of scenarios, events and constraints. There are some rules that can be executed during off-peak time to apply both Dynamic ON/OFF and Vertical Scaling which, are mentioned in Section II. We explain the implementation of MECP within a single cloud platform cluster for simplicity. Similarly, the

policy can be applied to all clusters in the Cloud platform. We explained the policy in steps written in plain English.

MECP is a management policy that is divided into two sets of rules, which are Monitoring Rules and Management Rules. Monitoring Rules should run at a short fixed period of time μ to collect data from the Cloud Cluster. On the other hand, Management Rules should be executed based on the type of data that have been collected. The following is a sample of Monitoring and Management Rules that MECP might be included which are executed during Off-peak time.

1) **Monitoring Rule 1:** When the time is off-peak and an Cloud Cluster is receiving an average workloads of less than 50 %, then Monitor Average Cluster Energy Consumption, Number of Over-loaded Hosts, Number of Under-loaded Hosts, Number of Normal-loaded Hosts, Average Resource Consumption happens every 20 second for an hour and also Report every 10 minutes.
2) **Monitoring Rule 2:** When the time is off-peak and the Cloud Cluster is receiving an average workloads of more than or equal 50%, then Monitor Number of Over-loaded Hosts, Number of Under-loaded Hosts, Number of Normal-loaded Hosts, Average Resource Consumption happens every 10 Seconds for an half-hour and also Report every 15 minutes.
3) **Management Rule 1:** When the Cluster Energy Consumption value is less than 1000 wph and Lower-loaded Hosts are more than 50%, then Migrate Virtual Machines to Normal-loaded Hosts and Switch-off Lower-loaded that do not run any Virtual Machines.
4) **Management Rule 2:** When the Cluster Energy Consumption value is less than 1000 wph and Over-loaded machines are more than or equal to 50% and Normal-loaded machines are between 30% and 40%, then Migrate if possible Small-size Virtual Machines to Over-loaded Hosts and Large-size Virtual Machines to Normal-loaded machines. Then,if possible Switch-off Lower-loaded virtual machines.
5) **Management Rule 3:** When the Cluster Energy Consumption value is between 1000 and 2000$wprh$ and the Normal-loaded Hosts are more than or equal to 50% and the Lower-loaded hosts between are 10% and 20%, Migrate all virtual machine running on Lower-loaded hosts to Normal-loaded hosts. Then, Switch-off Lower-Loaded hosts.
6) **Management Rule 4:** When the Energy Consumption is between 1000 and 2000wph and Over-loaded machines are more than 50% and Lower-loaded are less than10%. Then, Change resource configurations for all normalsized virtual machines in Over-loaded hosts to use basic-configuration.

IV. DESCRIPTION OF THE PROBLEM

High-level policies or guidelines such as management and monitoring policies represented in Section III, are defined by Cloud managers or non-technical users. Such policies are expressed according to a defined set of business or financial requirements. High-level policies can change regularly. This case implies that any High-level policy can be modified or extended. The continuous changing in policies is based on the nature of the technical environment and changes in regulation and business requirements that High-level policies might be applied to.

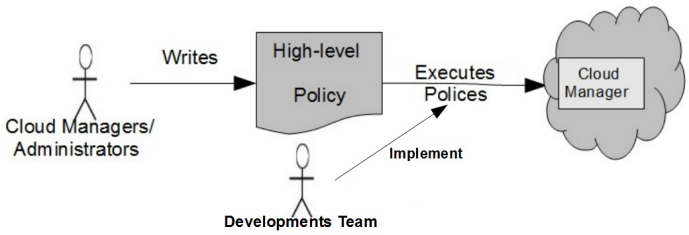

Fig. 3. The Gap Between High-level Polices and Their Execution

In Fig.3, it shows that, Cloud Mangers or Administrators specify a number of High-level policies during the design time. Those policies are written as a set of rules specified in natural language form. An example of such policies is MECP, which are shown as Rules 1-6. As presented in Fig.3, any High-level policies must be converted and implemented so that they can be executed in Cloud.

One possible solution is to do the conversion to be done manually by implementing policies using Low-level programming methods. The development team uses the High-level policy description and Low-level provided APIs that are dealing with the Cloud environment. Then, they implement the policy via designing a system that behaves in the following cycle Monitoring- Decision Making-Acting [8] processes. For example, the specified Management Rules and Monitoring Rules can be implemented using Object-Oriented Strategy Pattern [22] in which each rule can be encoded as a strategy. Then, a controller software component can be used for selecting a specified strategy and its execution time according to the description defined in High-level policy. The controller can interact with a number of software components for monitoring and providing actions.

Yet, the Cloud environment nature is dynamic and might include different types of events that require the High-level policy to be extended to form a new set of rules. An example is that MECP might be modified to include a new set of rules to control the behaviour of the Cloud Cluster during the deployment a new virtual machines in Cloud. Therefore, if the management system is designed using low-level programming methods or patterns, the software development and maintainability time and cost may increase. As a result, employing a Cloud system with a method that can automatically execute High-level policies in a less complex manner, is essential.

We found that there is a gap between the High-level policies specification and the conversion of such policies to be implemented in the Cloud environment. The appearance of this situation is as consequence of the following issues:

1) There is a no a generic architectural framework for executing High-level policies such as MECP in Cloud.
2) The lack of defining a methodology for specifying how to use lower-level provided APIs such as monitoring APIs and Actions APIs for executing High-level policies.

3) The absence of having a declarative language where High-level policies can be written in order to reduce the development and modification process of such policies.

V. THE GENERIC ARCHITECTURAL FRAMEWORK FOR ENFORCING HIGH-LEVEL POLICIES

The previously mentioned problem can be addressed by creating a generic architectural framework that is based on using Business Rules Engine as an Expert System [23] for executing management policy. The Business Rule Engine is integrated to work with the Cloud Manager. The communication between Business Rule Engine and Cloud Manger is done by two different components, which are; Sensor and Actuator. The Sensor is directly interlinked with Cloud APIs that are responsible for requesting measurement metrics that can be used for applying management policy. One the other hand, the Actuator uses the management actions APIs that directly launch the required actions, which are received from the Business Rule Engine. The functionality of the Business Rule Engine is as a Decision making Component that executes management policy such as MECP in Section III. The run-time execution behaviour for the Business Rule Engine will be presented in Section IV.

Fig.4 illustrates the generic architecture framework for executing management policy automatically. From Fig.4, the architecture consists of the following components: Business Rule Engine, Sensor, Actuator, and Decision Making or Supportive Component, which can be an optional component. The Cloud Manager directly interacts with the Sensor and the Actuator for either requesting measurement data or invoking one or more management action(s). The interaction is done by the Cloud Manger; it uses the provided remote APIs. Business Rule Engine executes Monitoring Rule1 to request Sensor to report every 10 minutes about the averages of Cluster Energy Consumption, Overloaded, Normal-loaded and Under-loaded hosts from the Cloud Manager. The final component can be considered as an optional component, which is used during the execution of Management Rules. The objective is to enhance the Business Rule Engine with a supportive component that can be used to support the Business Rule Engine with computation for setting values or performing local-search for a suitable solution. For example, producing migration schema for allocating migrated virtual machines to available hosts.

A. The Run-time Execution States for Monitoring and Management Policies

A number of run-time behavioural states are needed in order to define how the Business Rule Engine can interact with the Cloud Manger and how it can execute both Monitoring Rules as well as Management Rules. The states can specify when Monitoring Rules are launched to execute the Sensor Component and also when Management Rules, such as MECP, are executing Actuator. The states are Request-Update-Execute-Invoke see (Fig.5). According to a defined time period for executing Monitoring Rules, the execution process of the Business Rule Engine starts when events are received that specify the time and overall request from the Business Rule

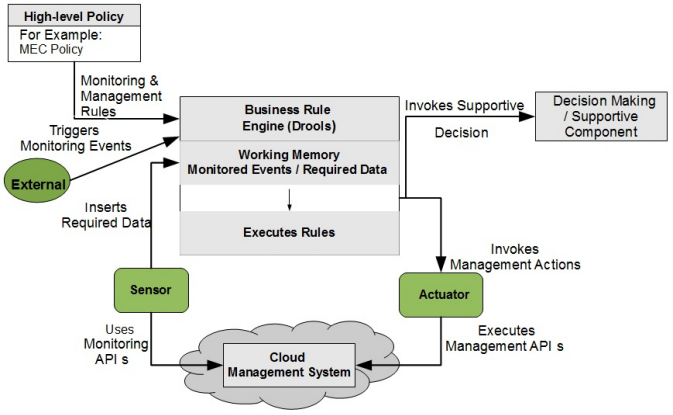

Fig. 4. The Generic Architectural Framework For Executing High-level Policies

Engine. During the Request state, the Business Rule Engine starts a stateful session that inserts received events such as time and overall requests, as facts into the Working Memory. Then, the Business Rule Engine uses predefined Monitoring Rules to determine which rules are satisfied. Let us say that the current time is off-peak time and the Cloud Cluster is receiving less than 1000 requests per second. Therefore, Monitoring Rule1 will be executed which ends up sending Monitoring action, Reporting action and the duration for Monitoring to the Sensor. The Sensor uses the received information and starts the data collection process. According to Execution Monitoring Rule1, the data that is required are Cluster Energy Consumption, Normal-loaded, Over-loaded and Lower-loaded Hosts and Average of Resource Consumption at each host in the Cloud Cluster. At the end of the Monitoring Cycle, the Sensor reports the collected values to the Business Rule Engine. After retrieving the required data, the Business Rule Engine state changes to Update state. At the Update state, a new stateful session is started with the process of inserting the retrieved data as facts in the Business Rule Engine When the updated process is finished, the state is changed to Execution state.

At the Execution state, the predefined Management Rules such as Management Rules (see Section III) will be executed. Therefore, the rule engine uses inserted data as constraints

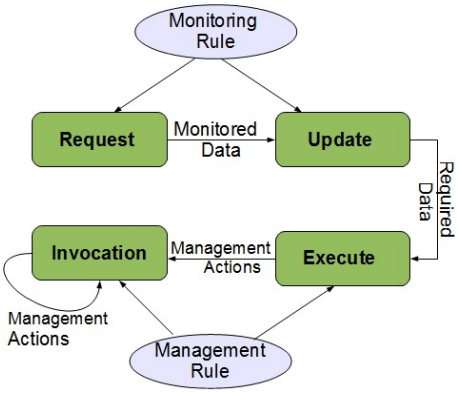

Fig. 5. The Run-time Execution States for Business Rule Engine

in order to define which set of Management Rules are satisfied. For instance, if the average value of Cluster Energy Consumption is < 1000 and the number of Lower-loaded Hosts are $> 50\%$ then Management Rule1 will be executed. Then, the Business Rule Engine invokes the Actuator with two actions, which are; Migration Virtual Machines and Switching-off actions. The Business Rule Engine state is transferred to Invocation state.

During the Invocation state, a sequential execution for each specified action starts which involves interaction with both Actuator and Supportive components that implement some required computation. During this state, the Cloud Manger remotely receive management actions that request migration virtual machines to other hosts and Switching-off Lower-loaded hosts.

VI. THE GENERIC ARCHITECTURAL FRAMEWORK IMPLEMENTATION

For implementing the proposed framework, we used Open-Nebula management system and integrated the Business Rule engine to interact with the ONED demon remotely. The integration is done based on the interaction with Open-nebula RPC interfaces. The Rule Engine is implemented using Drools Business Rule Engine. Drools is chosen for the following purposes: simplicity for representing the domain knowledge using JAVA classes. Rules such as Monitoring Rules can be written in the form of Business Rules (see Section II). In addition, Drools can support order execution of rules via grouping rules as rule-group flow. the Sensor is implemented as a software component that periodically uses Open-Nebula monitoring APIs for collecting data required by Monitoring Rules such as; the hosts' resources consumption and virtual machines consumptions. The Sensor uses an XML parser to parse retrieved data from the RPC message and sends data in XML-formate to the Business Rule Engine. On the other hand, the Actuator is implemented as a software component that deals with management actions APIs, which are triggered when management messages are launched from the Business Rule Engine. The Actuator also uses parser to parse the management messages received from the Business Rule engine and encapsulates the messages as RPC format. Optionally, the Drools Planner is used for the implementation of an additional optional Decision Making component in order to produce a migration schema using Simulating Annealing search-based algorithm.

VII. THE GENERAL RULES EXPRESSION SUITED THE PROPOSED ARCHITECTURE

The proposed architectural framework in section VI requires rules to be written in a certain structure. Since the framework uses Monitoring and Management rules, we conclude that the expression of Monitoring Rules is similar to Business Actions Rules (See Section II) whereas the representation of Management Rules is as same as Constraints Business Rules(See Section II). We are going to present a general expression for writing Business Monitoring Rules as well as Business Management Rules as UML-activity diagrams for clarifying the structure of Rules and for easily mapping High-level specified Rules to Declarative Rule Language, such as Drools Language.

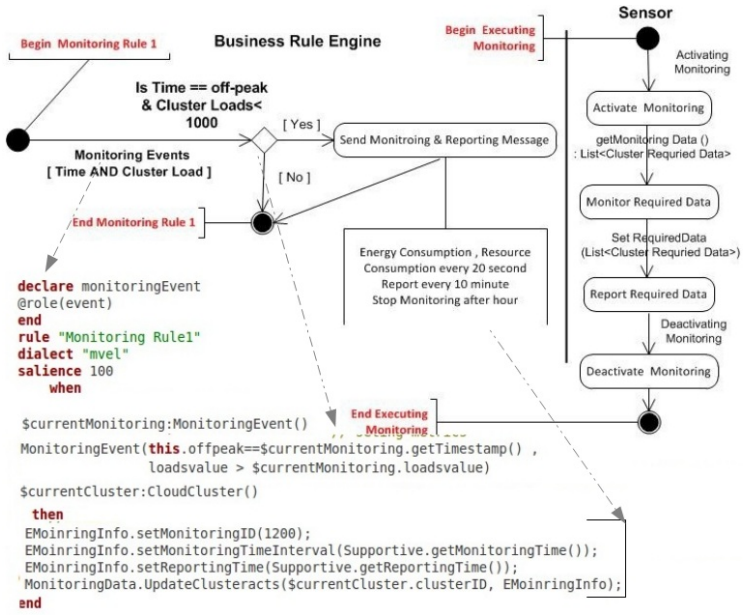

Fig. 6. UML-Activity Diagram and Drools Language For The Structure of Monitoring Rule1

A. The General Expression of Monitoring Rules

Monitoring Rule is a type of Business Action Rule. The typical structure of Business Action Rule is as Event-Condition-Action rule [17]. Thus, the rule structure should include definitions for event messages, which will be inserted as a fact in the Business Rule Engine. Events facts are used in rule condition statements; they state which metrics that are required for monitoring. For example, an Off-time peak message and a load message can be considered as events in Monitoring Rule. We refer to events as Monitoring Events.

Fig.6 shows a UML-activity diagram followed by Rule Language for expressing a Monitoring Rule. We demonstrate Monitoring Rule1 mentioned in Section III as an example. The diagram starts with a representation for Monitoring Events, which are an Off-time peak message and a load message. A branch which maps a rule condition statement to check the time and loads values. The assertive arrow leads to the rule action part which contains an invocation message to tell which state what to monitor, how long monitoring will last and when to report the values to the Business Rule Engine.

B. The General Expression of Management Rules

Management Rule is a kind of Constraints Business Rule, which its structure is Constraints Condition-Consequence [17]. Management Rule condition statement uses the reported monitored data, which are denoted as Required Data, and constrains values. An example of the Required Data are Cluster Energy Consumption, loads and Resources Consumption values. Depicted in Fig.7 is a UML-activity diagram and Drools Language Representation for Management Rule1. The rule starts by us-ing the Required Data, which are; Cluster Energy Consumption measured in Watts and the percentage of lower-loaded hosts in the Cloud Cluster. The assertive arrow leads the rule consequence part, which consists of two sequential invocation Management Actions, which are; Migration VM

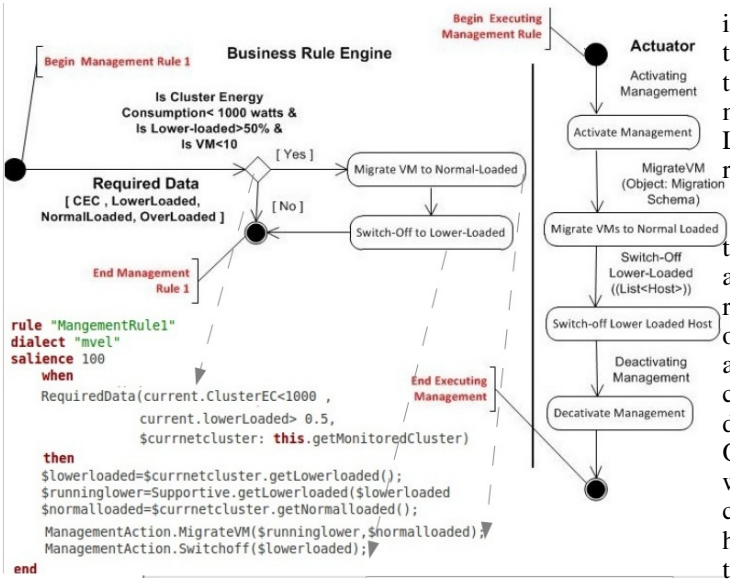

Fig. 7. UML-Activity Diagram and Drools Language For The Structure of Management Rule1

and Switching-off Lower-loaded hosts. These messages are sent to the Actuator (see Fig.7).

VIII. RELATED WORK

Rules can be described in a simplified manner by using modelling languages to describe Business Rules combined with a well-defined transformation methodology. There are a number of modelling languages for describing Business Rules. Some of these languages are Semantics of Business Vocabulary and Business Rules (SBVR) [24], Simple Rule Mark-up Language (SRML) [25], UML-based Rule Modelling Language (URML) [26], and Business Process Modelling Notation (BPMN) [24].

SBVR is a language that attempts to provide a definition of a standardised rule modelling vocabulary [24]. SBVR presents a vocabulary that is intended to become a standard upon which many grammars can be based for specifying rules. SRML is also a descriptive language which can represent rule models but with limited vocabulary [25]. On the other hand, URML [26] [27] is a graphical representation for Rules, which supports modelling domain vocabularies (i.e., ontologies) and rules types derivation. In URML, rules are represented as Circles with identifiers, a Condition arrow and a conditioned model element. One beneficial of URML is that rules can be translated to an Event-Condition-Action rule structure [26]. In [9], there is an attempt to use BPMN, which is a graphical modelling proposed by OMG [24]. BPMN is as a collection of graphical representation that can be used to describe a business process. BPMN is used in [9] to provide a high-level graphical description for simple rules pattern. The objective is to simplify the expression of business rules used in business application. The transformation of graphical representation for rules in BPMN is accomplished by generating a methodology for mapping to Drools Rule Language [9].

All the presented languages can be used to provide a High-level description for the Business Rules that we used in the generic architecture. Therefore, we are going to study the possibility of using SBVR and BPMN for modelling the Monitoring and Management Rules and to provide a methodology for transforming the rules to Rule Declarative Language. The objective is to provide a method for simplifying rules expression by non-expert users.

There are various architectures that have been proposed that based on rules for controlling actions in Cloud. In [28], an architecture for reconfigurable cloud-applications during run-time in the cloud is presented. The architecture is based on customised rule engine which enables the execution of a set of rules to govern the application behaviour. Application providers can update application behavioural policies during run time, such as adapting new load conditions. The OVF description domain model language for virtualisation, which is composed of vocabulary description for VirtualMachine, HardwareComponent, Service, VirtualDataCenter, etc, has been used for representing domain knowledge that is to be used by the engine. Furthermore, Semantic Web Rule Language(SWRL)is used to enable an easy definition of High-level policies for defining application behaviour on top of the static. It is argued that the performance of the architecture performance is based on the performance of the rule engine [28]. The architecture does not provide a description of the types of policy or how rules can be expressed in the policy. The usage of OVF- domain description language can increase the probability of policies among various cloud based-application. This architecture is different from the architecture presented in the paper since the generic architecture uses Business Rule Engine that controls the monitoring side and the management side. In addition, we provide a classification for rules that High-level policy might include.

Another approach, which has some similarities to our approach but is concerned with low-level implementations, is presented in [8]. The rules are used either for reconfiguring virtual machines resources or for migrating virtual machines to another host. The adaptation is based on the use of configuration rules scheme that acts based on comparing resources utilization level and with predicted thresholds values. In the approach, there is also suggestion to use algorithms such as First-fit, RoundRobin and Monto-Carlo for reallocating migrated virtual machines [8]. The similarity with our approach is found in the use of rule-based approach for reconfiguring and migration. However, our approach is concerned with executing High-level policies defined by the Cloud Managers, which can be written in plain English. Therefore, we used the Business Rule System and presented an architecture for integrating this engine with Cloud Management system. The objective is to investigate how High-level policy, such as Management Energy Consumption policy can be written as Business Rules and be executed in the Cloud.

In [29] the Business Broker service-oriented architectural approach, which is based on web services technologies, is for deployment of various rule engines. The approach provides a service layer interface for accessing and executing business rules from various knowledge bases. Furthermore, the Business Rule Brokering Layer allows heterogeneous rules engines to be encapsulated and be used. Various rule engines can be plugged via using the adapter pattern [30]. The rule-based knowledge is Web services which can be accessed remotely [29]. Using

Business Brokering architecture can increase scalability for the proposed generic architecture. Therefore, there is feasibility for extending the generic architecture framework to be organised in multi-level hierarchical layers.

IX. Conclusion

We proposed an architectural framework to enforce High-level policies in Cloud. The architecture is based on using Business Rule Engine to control the execution of High-level policies, such are Management Energy Consumption Policy, a Sensor for dealing with monitoring APIs, an Actuator for lunching Management Actions APIs and a Supportive Component to add extra computation or more decision making features. For simplifying the expression of High-level policy and implementing it in the generic architecture, we divided High-level Policy into two sets of rules; Monitoring and Management Rules. We correlate the expression for both groups as Business Rules in which their structures are demonstrated in UML-activity diagrams. The architecture is implemented using Drools Business Rule Engine integrated with OpneNebula cloud management system.

References

[1] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*. ACM, 2010.

[2] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "Enacloud: An energy-saving application live placement approach for cloud computing environments," in *IEEE International Conference on Cloud Computing*. Ieee, 2009, pp. 17–24. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5284078

[3] J. Xu and J. a. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*. Ieee, 2010, pp. 179–188. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5724828

[4] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, and L. Yuan, "Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers," in *IEEE International Conference on Services Computing*. Ieee, 2010, pp. 514–521. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5557272

[5] K. Ye, D. Huang, X. Jiang, H. Chen, and S. Wu, "Virtual machine based energy-efficient data center architecture for cloud computing: A performance perspective," in *IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*. Ieee, 2010, pp. 171–178.

[6] L. Lefèvre and A.-C. Orgerie, "Designing and evaluating an energy efficient cloud," *The Journal of Supercomputing*, vol. 51, no. 3, pp. 352–373, 2010. [Online]. Available: http://www.springerlink.com/index/10.1007/s11227-010-0414-2

[7] X. Fan, W.-d. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM International Symposium on Computer Architecture*. San Diego: ACM Press, 2007.

[8] D. Borgetto, M. Maurer, G. Da-Costa, J.-M. Pierson, and I. Brandic, "Energy-efficient and sla-aware management of iaas clouds," in *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, ser. e-Energy '12. New York, NY, USA: ACM, 2012, pp. 25:1–25:10. [Online]. Available: http://doi.acm.org/10.1145/2208828.2208853

[9] D. Di Bona, G. Lo Re, G. Aiello, A. Tamburo, and M. Alessi, "A Methodology for Graphical Modeling of Business Rules," *2011 UKSim 5th European Symposium on Computer Modeling and Simulation*, pp. 102–106, Nov. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6131196

[10] EucalyptusCommunity, "Eucalyptus cloud platform," 2012. [Online]. Available: http://www.eucalyptus.com

[11] OpenNebulaCommunity, "Opennebula: The open source toolkit for cloud computing," 2012. [Online]. Available: http://www.opennebula.org

[12] G. Toraldo, *OpenNebula 3 Cloud Computing*. Birmingham B3: PACKT Publishing, 2012.

[13] ovirtCommunity, "ovirt," 2012. [Online]. Available: http://ovirt.org

[14] XenCommunity, "Xen," 2012. [Online]. Available: http://www.xen.org

[15] KVMCommunity, "Kvm," 2012. [Online]. Available: http://www.linux-kvm.org

[16] VMWareCommunity, "Vmware," 2012. [Online]. Available: http://www.vmware.com

[17] I. Graham, *Business Rules Management and Service Oriented Architecture: A Pattern Language*. Wiley, 2007. [Online]. Available: http://books.google.co.uk/books?id=_InzFf5XpeQC

[18] JBossCommunity, "Drools tools reference guide," pp. 1–24, 2011. [Online]. Available: http://www.jboss.org/drools/documentation

[19] C. L. Forgy, "Rete : A fast algorithm for the many patternimany object pattern match problem," *Artificial Intelligence*, vol. 19, pp. 17–37, 1982.

[20] D. Sottara, P. Mello, and M. Proctor, "A configurable rete-oo engine for reasoning with different types of imperfect information," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 11, pp. 1535–1548, Nov. 2010.

[21] C. Hyser, B. Mckee, R. Gardner, and B. J. Watson, "Autonomic virtual machine placement in the data center," HP Laboratories, Tech. Rep., 2008. [Online]. Available: http://www.hpl.hp.com/techreports/2007/HPL-2007-189.html

[22] E. Freeman, E. Robson, B. Bates, and K. Sierra, *Head First Design Patterns*, ser. Head First Series. O'Reilly Media, Incorporated, 2004. [Online]. Available: http://books.google.co.uk/books?id=LjJcCnNf92kC

[23] J. Durkin, *Expert systems: design and development*. Macmillan, 1994. [Online]. Available: http://books.google.co.uk/books?id=9-BQAAAAMAAJ

[24] O. OMG, "Semantics of business vocabulary and business rules specification sbvbr," 2008. [Online]. Available: http://www.omg.org/spec/SBVR/1.0/PDF/

[25] H. Boley and S. Tabet, "Simple rule markup language srml," 2012. [Online]. Available: http://ruleml.org/

[26] REWERSE, "Uml-based rule modeling language," 2012. [Online]. Available: http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=URML

[27] S. Lukichev, A. Giurca, G. Wagner, D. Gasevic, and M. Ribaric, "Using uml-based rules for web services modeling," in *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, april 2007, pp. 290 –297.

[28] L. M. Vaquero, D. Morán, F. Galán, and J. M. Alcaraz-Calero, "Towards runtime reconfiguration of application control policies in the cloud," *Journal of Network and Systems Management*, vol. 20, no. 4, pp. 489–512, Aug. 2012. [Online]. Available: http://www.springerlink.com/index/10.1007/s10922-012-9251-3

[29] F. Rosenberg and S. Dustdar, "Design and implementation of a service-oriented business rules broker," in *Seventh IEEE International Conference on E-Commerce Technology Workshops*. Ieee, 2005, pp. 55–63. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1521010

[30] T. J. Shalloway, Alan, *Design Patterns: Elements of Reusable Object-Oriented Software with Applying Uml and Patterns:An Introduction to Object-Oriented Analysis and Design and the Unified Process*. Addison Wesley, 2003. [Online]. Available: http://books.google.co.uk/books?id=_XDFAAAACAAJ