Ontology based configuration of virtual systems in a computer cloud

Andrejs Ābele

Department of Management Information Technologies
Riga Technical university, Faculty of Computer Science and Information Technology
1/3 Meža iela
Latvia, Riga LV-1048
andrejs@iti.rtu.lv

Abstract: Multiple combinations of hardware and network components can be selected to design an Information Technology (IT) infrastructure that satisfies requirements. This paper proposes depreciation a framework which helps to create an efficient virtual infrastructure and keep it up-to-date. The framework monitors changes in configuration of the virtual infrastructure and uses gathered data for generating infrastructure optimization recommendations. The ontology, which classifies virtual machines is the key part of the proposed framework. The structure of this ontology is outlined in the paper and an application example is presented.

1 Introduction

Investment in Information Technology (IT) infrastructure including hardware, networks and systems software are expensive and have high level of depreciation. Design and optimum sizing of IT systems is the result of balancing several conflicting requirements, including technical performance and cost, organization impact, and user acceptance [AF00][AF01]. Theoretical research often focuses on two design and sizing problems. The first problem is distribution of the overall computing load of a system onto multiple machines in order to minimize hardware costs. The second problem is where to locate machines that need to exchange information in order to minimize network costs [AFT02]. Virtualization [V07] enables resources sharing and scalability thus reducing the overall cost of IT infrastructure. The cloud increases scalability even further by enabling quick response to workload changes. It also eliminates dependency from the physical location of computing resources.

There are three types of services computer clouds provide. First one is Software as a service (SaaS). With SaaS, a single application is delivered to thousands of users from vendor's servers. Customers do not pay for owning the software rather for using it

[M05]. Second one, Platform as a Service (PaaS), is a complete platform, including application development, interface development, database development, storage, testing, and so on, delivered through a remotely hosted platform to subscribers. Based on the traditional time-sharing model, modern platform-as-a service providers provide the ability to create enterprise-class applications for use locally or on demand for a subscription price or for free [L04]. Third one, Infrastructure as a service (IaaS), delivers basic storage and compute capabilities as standardized services over the network. Servers, storage systems, switches, routers, and other systems are pooled and made available to handle workloads that range from application components to high-performance computing applications [R06].

A single cloud provides services to many users who deploy their virtual infrastructure in the cloud. Each user aims to optimize performance (e.g., speed, reliability) of her virtualized IT infrastructure and to utilize cloud's resources in the most efficient way (i.e., to reduce usage charges). Additionally, the virtual infrastructure in the cloud is updated frequently to take advantages of newly available technologies and releases. The cloud service provider has an opportunity to capture these data. The data can be provided back to users as suggestions for optimizing their virtual infrastructure.

The objective of this paper is to elaborate a framework for accumulating and systemizing clouds usage data and using these data in optimization of virtual infrastructure. The framework uses an ontology web language to classify virtual machines and appliances (similarly as products are classified in [YMWZ09]), an agent which collects system performance statistics and fuzzy logic to cluster collected statistics and update the ontology based on the results. The paper reports research in progress, and at its current stage the emphasis is on overall framework and outline of the ontology.

2 Framework

The main motivation behind the framework is that in a cloud virtual machines do not have a long life span. If a VM gets out-dated then the hole machine gets replaced with a newer version based on a predefined configuration pattern.[S08] Based on frequent VM changing frameworks optimizations method becomes meaningful. If VM would stay the same for long time, there wouldn't be possible to update ontology based on VM performance and because of that there wouldn't be created any new virtual machines combinations.

The framework (Figure 1) includes eight main parts: 1) Specifications of requirements, 2) VM Classifier, 3) VM Component repository, 4) OWL (Web Ontology Language) Register, 5) VM template and VM appliances template repository, 6) VM classifier based on system performances, 7) System performance statistics collecting agent 8) Cloud.

2.1 Specifications of requirements

To create virtual machine Templates or virtual appliance Templates, it is necessary to specify parameters of virtual machines. Specification of requirements consists of four possible choices:

- Chose VM template from repository – This means, that a template of a virtual machine is chosen from the repository. The main idea is to collect a large collection of virtual machine templates, so users could find best solution, without the need to create own template. It will be possible to modify existing templates, but the idea is, that user should be able to find a ready template which fulfils their needs.

- Chose VM appliance template from repository – As VM appliance here is meant combination of virtual machines that fulfils a certain task or combinations of tasks. You could say that VM appliance is a template of a systems infrastructure. As in previous point the idea is to collect so many different templates, as possible, so an infrastructure creation would be made as easy as possible.

- Combine VM manually – here, a possibility is given to combine a virtual machine, using components from VM component repository, which fulfils specific requirements. Of course there are some basic requirements that need to be fulfilled, so a VM could be created. For example, it must have an operating system, processor, RAM, etc.

- Combine VM appliance manually – Combine VM appliance can implement usage of ready VM templates and manually created. In this stage it is only possible to define what VM will be in an appliance and what will they contain, but configuring interaction between them will be possible only in later stages.

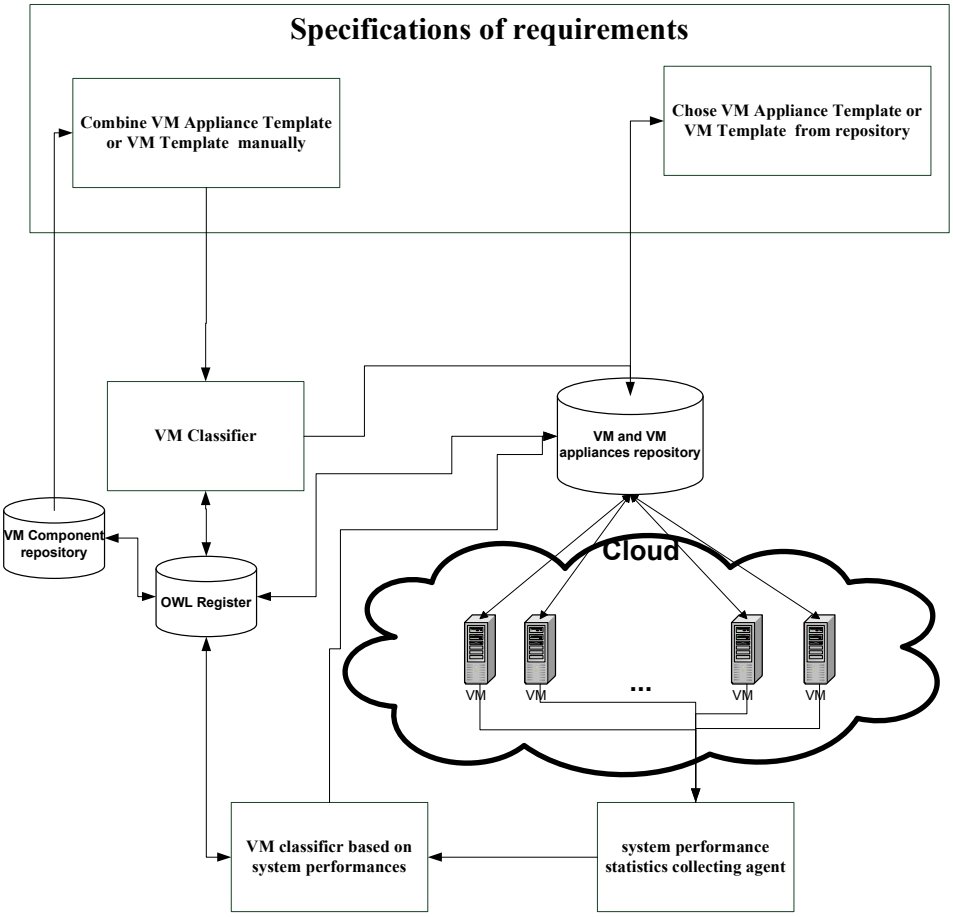

Figure 1: Framework

It is also possible to choose a ready template as a basis for manual configuration of VM, but that is different for VM appliances as VM in a VM appliance templates have been configured to work together, which is impossible in this stage.

2.2 VM Classifier

To compare performance of multiple VM, it is necessary to know which systems are similar and perform similar tasks. Therefore, it is necessary to classify newly created VM's and VM appliances. This classifier classifies VM's based on their architecture. The ontology is used for classification purposes. When VM is classified, its template is added to the VM repository. The ontology is described in the third section of this paper.

2.3 VM Component repository

This repository consists of components which can be installed automatically without human interaction. It contains components which represent hardware, as processor type, amount of RAM, HDD capacity. It also contains some software packages which can be installed automatically. The repository is connected to the OWL register, if the register wouldn't contain all the components, classifier wouldn't be able to classify the VM.

2.4 OWL Register

The OWL register contains the ontology which uses the OWL web ontology language. This ontology contains descriptions of every VM component, every VM machine an appliances. Purpose of this ontology is to help select and overview MV templates. This ontology is elaborated in the third section of this paper.

2.5 VM and VM appliances repository

The VM repository contains VM templates and VM appliances templates. It is interacting with the OWL register because all templates must be registered in the OWL register, otherwise it would be impossible to use those templates. Because the templates are just a description of VM configuration, so it needs actual VM components which it can find through OWL register. It is VM classifiers task to make sure that the VM repository and the OWL register are in sync.

2.6 System performance statistics collecting agent

This daemon collects information about VM performance. It is necessary, so it would be possible to determine which VM configuration is the most effective. Daemon monitors not only individual VM performance, but also VM appliances. From individual VM it collects – CPU data, file system disk usage, attached HDD usage, amount of swap space, processors and associated threads, system load, memory allocation. To determine appliances efficiency daemon collects – network interface activity, network latency, DB activity.

2.7 VM classifier based on system performances

This classifier classifies VM based on their performance on contrary to classifier which vas discus in Section 2.2 of this paper where VM's where classified based on their architecture. This classifier deals with QoS [S07][ZHP10]. As the range of data changes it is hard to maintain static limits for classification, so it is necessary to use Fuzzy logic [ZTJ11] [GGZ03].

2.8 Cloud

Cloud is where all the physical resources are located. It ensures efficient execution of VM by assigning necessary computing resources with limits and according to conditions specified in the hosting agreement. The proposed framework enables consumers to select and to configure their VM in order to maximize performance and to minimize hosting expenses by requesting only as much resources as they need.

3 Ontology

The Ontology is used to classify VM deployed in the cloud. This classification enables identification of VM providing similar services and accumulation of VM performance data. This ontology is developed using OWL DL. All VM components are defined as classes in the ontology (Figure 2). Newly created VM templates are defined as instances. If an instance, which does not belong to any ground level category, is created, then a new category is created. . Only an "is-a" relationship is visualized in the figure. There are also properties like "hasOS", "hasServices", "hasVMtipe", "worksWith". Below are written some logical conclusions which using an OWL reasoner can be deducted from this ontology.

1. Every VM is something
   that hasOSes exactly 1 Operating_system
   and
   that hasServiceses a Services
   and
   that hasVMtipes a VM_Tipe .
2. Every DB_SERVER is a VM
   that hasServiceses a Database
3. Every WEB_SERVER is a VM
   that hasServiceses a Web_server
4. Every LinuxWEBserver is a WEB_SERVER
   that hasOSes nothing but Linuxes .
   Every WEB_SERVER
   that hasOSes nothing but Linuxes is a LinuxWEBserver
5. Every Ubuntu is a Linux.

At the beginning reasoning is described that a VM must have one operating system, must have a service chosen (for now are available – web server, DB server, firewall) and virtualizations type. The second reasoning step describes a database server. A DB server is every VM that has a database service. The third reasoning step is similar to second, just with web server. The fourth reasoning step explains that Linux web server is a web server has operating system only Linux and that web servers that have Linux are Linux web servers. The fifth reasoning step shows that every Ubuntu is a Linux operating system.

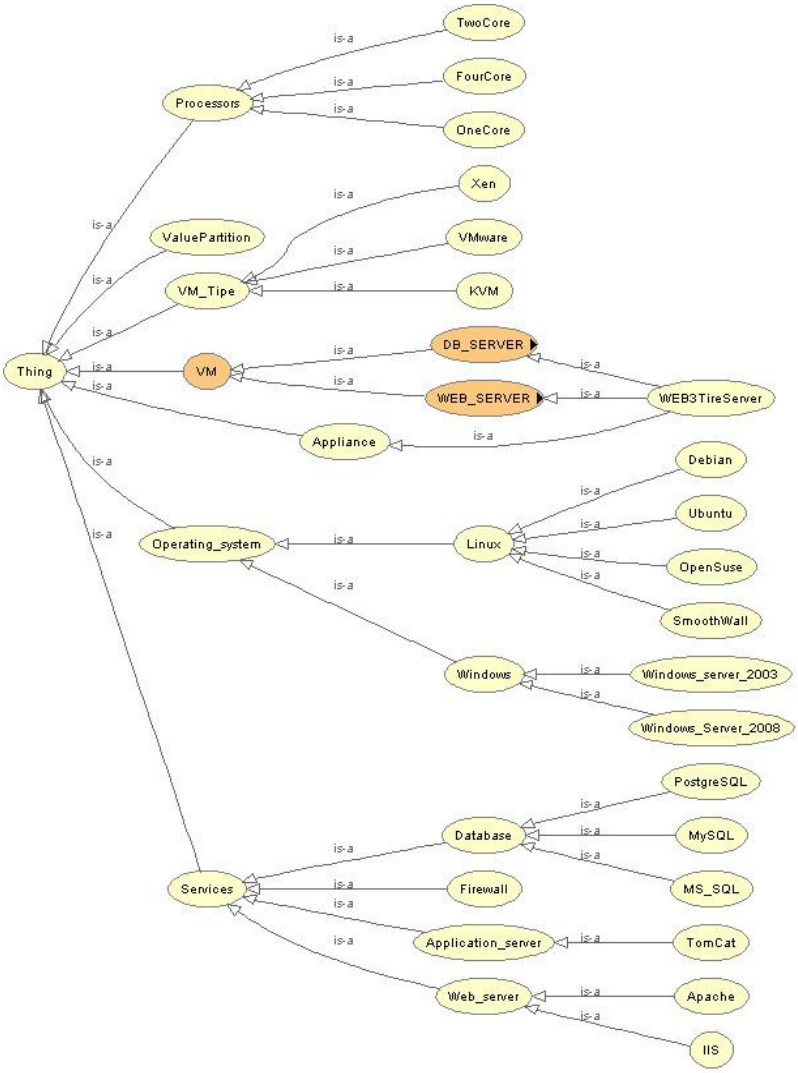

Figure 2: Ontology tree

4 Example

In this section, are demonstrated basic possibilities of the ontology. The example shows how manually configured VM are recognised and classified by the OWL reasoner.

A VM web server is defined (Figure 3. A). It has Ubuntu operating system and Apache web server installed. After activating the OWL reasoner this VM is recognised as LinuxWEBserver (Figure 3. B). Now we will try to create a VM which contains

operating system Ubuntu and has VM type VMware, but reasoner does not categorises it (Figure 4. A), because it doesn't fulfil first condition, which was discussed in 3. section, it doesn't have a service. After a service MySQL is added, the reasoner recognises this VM as LinuxDBserver (Figure 4. B), which is a subcategory of VM.

Figure 3: Web server (A, B)

Figure 4: DB server (A, B)

5 Conclusions and Future Work

This paper presents the framework that will facilitate virtual IT system creations in a computer cloud. This framework allows using usage data to optimize performance of virtual infrastructure. Users are motivated to share data by getting the feedback and provider obtains a competitive advantage to other providers by providing an additional service to customers. Ontology is the key part of the framework; it allows recognizing the VM using reasoners. Because requirements frequently changes, ontology is subject to continuous refinement. Another key part is the performance classifier, which is the subject of further research. And main goal for the future is to create a working prototype where all parts work together.

References

[AF00] Ardabna,D; Francalanci, C:A Cost-Oriented methodology for the design of Web based IT architectures. SAC, Madrid, 2002

[AF01] Ardabna,D; Francalanci, C:A cost-oriented approach for the design of IT architectures. Journal of Information Technology, 2005

[AFT02] Ardabna,D; Francalanci, C; Trubian, M: Joint Optimization of Hardware and Network Costs for Distributed Computer Systems. IEEE Transactions on system, man, and cybernetics-part a: Systems and humans, vol 38, No. 2. 2008

[R06] Reese,G: Cloud Application Architectures: Building Applications and Infrastructure in the Cloud (Theory in Practice (O'Reilly)), O'Reilly Media, 2009

[YMWZ09]Yang, D; Miao, R.; Wu, H; Zhou, Y: Product configuration knowledge modeling using ontology web language. Expert Systems with Applications 36, 2009

[ZTJ11] Zhuang, L; Tao, H; JuanJuan, Z: Solving Multi-objective and fuzzy multi-attributive integrated technique for QoS-Aware Web Service Selection. Networking and Mobile Computing, pp.735–739, 2007

[ZHP10] Zhiyong C.; Haiyang W.; Peng P.: An Approach to Optimal Web Service Composition Based on QoS and User Preferences. International Joint Conference on Artificial Intelligence, Pasadena, 2009

[GGZ03] Guan Y.; Ghose A.; Lu Z.: Using constraint hierarchies to support QoS-guided service composition, International Conference on Web Services (ICWS '06), Chicago, 2006

[L04] Linthicum, D.: Cloud Computing and SOA Convergence in Your Enterprise: A Step-by-Step Guide. Addison-Wesley Professional. 2009

[S07] Stantchev V.: Performance Evaluation of Cloud Computing Offerings. 2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences, Sliema, 2009

[M05] Miller, M.: Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online. Que, 2008

[S08] Sun Microsostems: IntroductIon to Cloud Computing ArChiteCture – Withe Pappre, Sun Microsystems, 2009

[V07] VMware :Understansing Full Virtualization, Paravirtualization, and Hardware Assist – White Papper, VMware, inc., 2007