Self-Managed Services Conceptual Model in Trustworthy Clouds' Infrastructure

Imad M. Abbadi
Oxford University Computing Laboratory
Wolfson Building, Parks Road,
Oxford OX1 3QD, UK.
imad.abbadi@comlab.ox.ac.uk

Abstract

Current clouds infrastructure do not provide the full potential of automated self-managed services. Cloud infrastructure management are supported by clouds' internal employees and contractors (e.g. enterprise architects, system and security administrators). Such manual management process that require human intervention is not adequate considering the cloud promising future as an Internet scale critical infrastructure. This paper is concerned about exploring and analyzing automated self-managed services for cloud's virtual resources. We propose a conceptual model of self-managed services interdependencies and identify static and dynamic factors affecting their automated actions in the context of cloud computing. Next, we identify the challenges involved in providing secure and reliable self-managed services. We have just started the work in this area as part of EU funded Trusted cloud (TCloud) project[1].

1 Introduction

A cloud is a new buzzword in computing terms, which has various definitions, for example, *'Cloud is an elastic execution environment of resources involving multiple stakeholders and providing a metered service and multiple granularities for specified level of quality'*[4]; another definition, *'Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.'*[5]. Cloud support three main deployment types Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [5]. IaaS provides the most flexible type for cloud users who prefer to have the greatest control over their resources, while SaaS provides the most restrictive type for cloud users where cloud providers have full control over the virtual resources. In other words cloud computing provides a full outsourcing support for the SaaS, a partial outsourcing support for PaaS (more specifically it provides the virtual environment and software tools for users helping them to develop and deploy their applications), and a

[1]http://www.tclouds-project.eu/

minimal outsourcing support for IaaS (more specifically cloud provider mainly manages the infrastructure components running the virtual machines). In this paper we are mainly focusing on IaaS cloud type. Cloud users for IaaS would typically be organizations.

The two main characteristics in potential cloud critical infrastructure, which differentiate it from current enterprise infrastructure are pay-per-use payment model and automated self-managed services [4]. In this paper we are mainly focusing on self-managed services for infrastructure virtual resources. This provides cloud infrastructure with exceptional capabilities and new features. For example, scale per use, hiding the complexity of infrastructure, automated higher reliability, availability, scalability, dependability, and resilience. These should result in cost reduction in terms of infrastructure maintenance.

The technologies behind current cloud infrastructure are not new, as they have been used in enterprise infrastructure for many years [7]. Cloud computing current understanding become popular with Amazon EC2 in 2006 [1], and its infrastructure is built up of technologies and processes based on in-house solutions. Although, current cloud infrastructure has been there for long, but we are still far away from achieving cloud potential features for several reasons which we discuss the ones related to self-managed services in this paper [4].

Cloud computing originate from industry (commercial requirements and needs) and has recently moved to research because of its promising potential as an Internet-scale computing infrastructure [2, 4]. The lack of academic research that formally analyze current cloud infrastructure results in confusion in realizing cloud potential features, as in the case of overestimating some cloud features (e.g. using immediate and unlimited keywords when describing some self-managed services). The lack of such resources also results in underestimating the challenges involved for providing automated clouds' infrastructure management. For example, some people interpret NIST definition for "resources rapidly provisioned and released" as if cloud should provide unconditional immediate and unlimited services; e.g. immediate and unlimited scalability. This is not a visible requirement considering nowadays technologies. There will always be a limitation in hardware resources. There are also many other factors that have not been considered for such a strong claim, e.g. should cloud provide unlimited resources in case of application software bugs, should resources be available immediately upon application request without user prior agreement, whats about financial control measures, etc. NIST definition does not mean immediate and unlimited; our understanding of "rapid provision and release" is controlled by boundaries and pre-agreed Service Level Agreement (SLA). For example, scalability should always be agreed between cloud user and provider in advance in upper/lower bound limits and defined in a SLA. If the organization wants to increase/decrease either limit, then they would need to update their SLA to reflect that (using automated APIs which simplifies the process). Also in case of increasing the upper limit the service provider needs to check if his internal infrastructure can cover the additional resources. When decreasing the lower limit, the service provider must ensure that customers are not overcharged for unused resources. These protect both cloud provider (e.g. have an expectation of overall resources upper limits) and cloud user (e.g. does not pay for resources used by software bugs resulting in illegitimate demands of virtual resources).

1.1 Objectives

The main objective of this paper is to define and explore clouds' self-managed services for virtual resources. In this we provide a conceptual model of self-managed services, we identified the factors that affects management services decisions, and then we discuss management services interdependency in cloud computing context. Based on this we discuss the challenges involved in providing secure and reliable self-managed services, which we have just start working on as part of TCloud project.

1.2 Organization of the Paper

This paper is organized as follows. Section 2 defines self-managed services, identify the main functions required to provide self-managed services, and the data these functions require to provide automated services. It then provides a conceptual model representing the interaction and relationship between these functions in cloud's multi-layered heterogeneous infrastructure. Section 3 discusses the challenges involved in providing self-managed services. Finally we conclude the paper in Section 4.

2 Self-Managed Services

In this section we briefly define self-managed services, identify the factors that affects managed services decisions, and provide a conceptual model for functions required to support self-managed services.

2.1 Definition

One of the main cloud potential features is the provision of automated self-managed services. Self-managed services are about providing cloud infrastructure with exceptional capabilities enabling it automatically (i.e. without human interventions) manage the infrastructure virtual resources and take appropriate actions on emergencies. Self-managed services are not about autonomic computing [3]. Autonomic computing is concerned about providing self-management for physical resources (e.g. physical servers) and it does not change dynamically based on changes in end-users requirements; however, self-managed services are about providing self-management for virtual resources, which run on top of physical resources and are based on many static and dynamic factors including end-user requirements and infrastructure properties. In other words self-managed services could run on top of autonomic computing but the opposite is not true (it is outside the scope of this paper to discuss this in further details).

For IaaS cloud type, self-managed services are concerned about supporting clouds' virtual resources availability, reliability, scalability, resilience, and adaptability. We now provide common definition of these services from literature and then we discuss how they fit in cloud computing context. To the best of our knowledge our paper is the first to discuss these services interdependencies in cloud computing context.

Reliability is a statistical number that is difficult and time consuming to measure. Reliability in general is related to the average time taken for the component to fail (i.e. Mean Time Between Failure (MTBF) or Mean Time To Fail (MTTF)) [11]. By fail we do not mean a planned maintenance window; i.e. if a component is brought down because of a fault then the component reliability will be negatively affected; however, if a component is brought down because of planned maintenance then reliability will not be affected at all.

Unlike individual component reliability, end-to-end *service reliability* is related to the success in which a service functions [8]. End-to-end service reliability is based on the resilience of components architect to support the service. High end-to-end service reliability implies that a service always provides correct results and guarantees no data loss. Higher individual components reliability together with excellent architect and well defined management processes, help in supporting higher resilience. This in turn increases end-to-end service reliability and availability.

Availability of a service represents the relative time a service provides its intended functions. It is based on two main factors: (a.) Mean time between failure (MTBF) and (b.) Mean time to repair (MTTR) [11]. Availability is then calculated as MTBF/(MTBF+MTTR). For example, 99.5% availability within a year means the service can be down within a year for 43.8 hours regardless of the reason (e.g. planned maintenance or unplanned failure). High levels of availability are the result of excellent technical architect, which considers well crafted procedures, redundant components, and high components reliability; i.e. resilient design.

Resilience is the ability of systems to maintain its features (e.g. serviceability and security) despite a number of sub-system and components failures [11]. High resilience can be achieved by providing redundancy together with careful design (eliminating single points of failure) and well planned procedures. Resilient design helps in achieving higher availability and end-to-end service reliability, as its design approach focuses on tolerating and surviving the inevitable failures rather than trying to reduce them. The complexity of cloud infrastructure means a large number of sub-systems have to work perfectly together to keep the operation running. In addition multiple and different groups need to cooperate, exchange critical messages and coordinate amongst themselves when taking a self-managed decision, as explained in section 2.3.

Adaptability is the ability of systems to provide timely and efficient reaction on system changes. Example of such changes include: (1.) increase/decrease in service request affecting overall system load, (2.) size of resources, (3.) security requirements, (3.) environmental conditions (e.g. different types of resources), and (4.) components failure. Adaptability should always consider overall system architect ensuring the main properties of a system are preserved (e.g. security, resilience, availability and reliability).

Scalability is the ability of systems to support dynamic environment by adding and removing resources quickly and efficiently. For example, on peak periods the system should scale resources up, and similarly on off-peak periods the system should release unneeded resources. These should not affect fundamental system properties and should always represent user defined requirements, as defined in SLA and QoS.

Scalability can be of either or combination of two types: horizontal scalability and vertical scalability. Horizontal Scalability is about the amount of instances that would need to be added or removed to a system to satisfy increase or decrease in demand. Vertical Scalability is about increasing or decreasing the size of instances themselves to maintain increase or decrease in demand. Scalability must not affect user-defined security and privacy requirements. For example, adding a Virtual Machines (VM) to a group of VMs must preserve the overall system security; e.g. having a less secure VM enables revealing sensitive content.

2.2 Factors Affecting Management Services

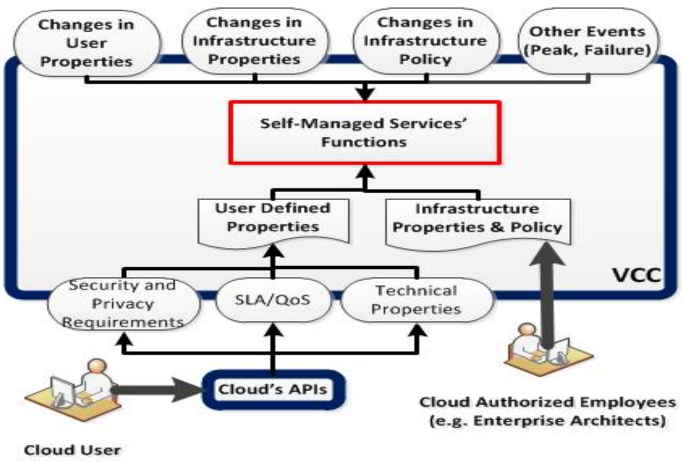
There are different tools, which help cloud employees managing cloud infrastructure. These cover virtual resource management, physical resource management, network management, cluster management, etc. In this paper we are mainly concerned about virtual resource management tools. There are many virtual management tools provided by different manufacturers (e.g. VMWare tool is referred to as vCenter [12], Microsoft tool is referred to as System Center [6]). Few open source tools have also been recently developed (e.g. OpenStack [10] and OpenNebula [9]), which support additional services (e.g. common APIs that can be used by users to interface with the cloud). These tools are still immature in providing self-managed services for virtual resources, as it requires human intervention. In this paper, for convenience, we refer to such tools using a common name Virtual Control Center (VCC). We believe that VCC will play a major role in supporting self-managed services. It is outside the scope of this paper to discuss current VCC tools nature and management; however, in this paper we propose the main functions that VCC should support and the factors that would affect the behaviors of the functions. We now identify the main factors, which would affect decisions made by self-managed services, as illustrated in Figure 1.

1. *User Properties (Dynamic Properties)* — A cloud user interacts with the cloud provider via cloud webpage and supplied APIs. This enables users to define *user properties*. User properties covers technical requirements (e.g. VMs, storage, and network architect), QoS/SLA requirements (e.g. system availability, reliability measures, and lower/upper resource limits), and user-centric security and privacy requirements (e.g. location of data distribution and processing).

2. *Infrastructure Properties (Static Properties)* — Clouds' physical infrastructure are very well organized and managed by, for example, enterprise architects, system administrators, and security administrators. Those people define the physical infrastructure properties, which includes: components reliability and connectivity, components distribution across cloud infrastructure (how far components are from each other), redundancy types, servers clustering and grouping, network speed, etc.

3. *Infrastructure Policy* — Policies should be defined by cloud authorized employees to control the behaviors of self-managed services.

4. *Changes and Events* — These represent changes in: user properties (e.g. security/privacy settings), infrastructure properties (e.g. components reliability, components distribution across the infrastructure, redundancy type), and other changes (increase/decrease system load, component failure, network failure).



Changes in User Properties
Changes in Infrastructure Properties
Changes in Infrastructure Policy
Other Events (Peak, Failure)
Self-Managed Services' Functions
User Defined Properties
Infrastructure Properties & Policy
VCC
Security and Privacy Requirements
SLA/QoS
Technical Properties
Cloud's APIs
Cloud Authorized Employees (e.g. Enterprise Architects)
Cloud User

Figure 1: Factors Affecting Self-Managed Services Behavior

2.3 Conceptual Model

In this section we propose a conceptual model for the functions and their interactions, which self-managed services should provide to manage cloud virtual infrastructure, as illustrated in Figure 2. These are as follows.

1. *System Architect* — This function resembles enterprise architect professionals. It is a an automated process that provides a well architect virtual infrastructure based on user properties, infrastructure properties and policies. The system architect process should provide a "resilient" system architect, as illustrated in Figure 2. This includes automatically deciding on the following (a.) individual components and their reliability; (b.) components/data redundancy type (e.g. RAID 1+0, RAID 5, dual channel); (c.) well crafted process management scripts and documents; and (d.) components distribution, grouping and management across cloud infrastructure. The outcome of this process, i.e. the resilient system architect, is fed to the "resilience process".

2. *The resilience process* — This function resembles system administrators. It process deploys and manages a resilient system architect (derived from the system architect process). It communicates with other required components or other management tools to create the resilient architect. It is outside the scope of this paper to discuss the deployment process. The resilience process is also in charge of marking failed virtual components unusable, (e.g. due to physical component failure), and to trigger this event to other functions (e.g. adaptability and availability functions).

3. *Adaptability* — This function is concerned about adapting end-to-end system for changes. Example of such changes (see Figure 1) include: component failure, increase in demand of a service, changes in user properties (e.g. security/privacy requirements), environment changes (e.g. interacting with different cloud service provider, using components from different vendors). Such changes must not compromise the overall service proper-

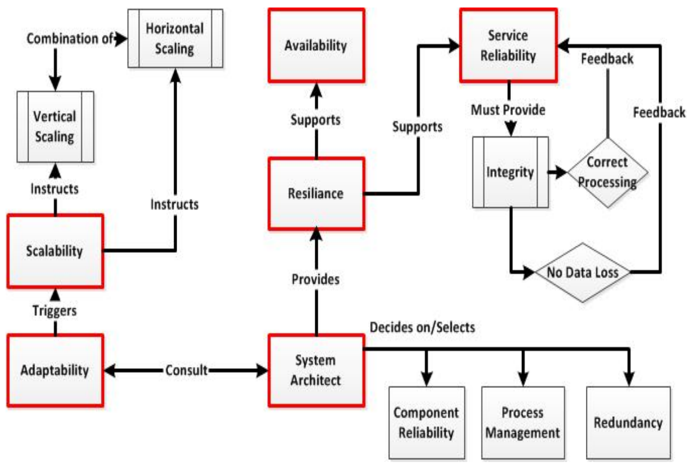


Figure 2: Self-Managed Services Conceptual Model In Cloud Context

ties as agreed with the customer. For example, adding/removing a VM to a group of VMs should not compromise the group security or integrity.

The adaptability process should automatically decide on an action plan, which either does it itself or delegates it to other processes. For example, when a group of VMs need more resources, the adaptability process first check if this group are authorized to scale their resources up by adding another VM or increase resources allocated to an existing VM. If the group is authorized, the adaptability process identifies other groups that might be affected by such scaling, and then coordinates with the other groups and decide by how much they would need to scale up. Once everything is coordinated and planned it triggers the scalability process to increase resources, as explained latter.

The adaptability process consults with the system architect process before taking an action. This is to ensure that the action plan will not have any impact on system properties (e.g. availability, reliability, resilience, and security).

4. *Scalability* — This function supports cloud elasticity feature by scaling resources up and down when needed. The adaptability process when detecting a need for either adding resources (e.g. peak period) or removing resources it instructs the scalability process to do so after doing preliminary steps, as described in the previous point.

5. *Availability* — System availability is supported by a resilient system design. The higher resilient a system the higher availability/reliability would be expected. The availability process is in charge of distributing the load evenly across cloud redundant resources. If a channel is marked unusable by the resilience process, the availability process immediately stops diverting traffic to that channel, and re-diverts the channel traffic to other active channels.

6. *Service Reliability* — This function is in charge of maintaining end-to-end service reliability, which is of higher priority than service availability. Most

importantly it ensures that the end-to-end service integrity is maintained (i.e. no data loss and correct service execution). If service integrity is affected by anyway and cannot be immediately and quickly recovered, service reliability then notifies the availability service to immediately bring the service down. This is to ensure that data integrity is always protected. Simultaneously, adaptability and resilience process should automatically attempt to recover the system and notifies system administrators in case of a decision cannot be automatically made (e.g. data corruption that requires manual intervention by an expert domain administrator).

3 Discussion — Main Challenges and Requirements

In this section we discuss the challenges involved in providing self-managed services. We then provide a high level overview of system requirement and how to meet the requirements for addressing the challenges.

3.1 Challenges

Providing self-managed services require careful consideration and analysis not only because of their complexity and inter-dependability but also for the following reasons.

1. Securing self-managed services should take into consideration the heterogeneous and complex nature of clouds' infrastructure. Providing self automated services for a virtual component requires: (a.) understanding the relative position of the component within the physical infrastructure, (b.) what are the user and infrastructure properties, (c.) how the management of the component would affect others, and (d.) what are the dependencies between the component and others.

2. Cloud infrastructure mixed nature consists of different types of hardware/software technologies, which are provided by multiple, and most likely competing vendors. The self-managed services require complex communications at different stages across various cloud entities. This in turn means different technologies from competing vendors should establish standard interfaces for exchanging messages. These are not present at the time of writing, and even providing such services using components from the same vendor are very complex to setup, error prone, and raise unique security challenges in comparison with traditional systems.

3. Cloud infrastructure is not hosted at a single data center that is located at a specific location; it is rather the opposite, as most likely it is distributed across distant data centers. This factor has a major impact on decisions being made by self-managed services for several reasons; for example, the distance and the communication medium between distant data centers will have an impact on data transfer speed. Automated services must consider this important factor and other related factors (e.g. data volume, data access mode, etc) when providing a service. For example, it is sometimes the

right decision to provide redundant active/active resources across distant locations and in other cases it is wiser to provide active/passive.

4. Cloud-of-cloud is a term that is used to refer to the collaboration of multiple cloud providers to support dependable cloud infrastructures; i.e. cloud providers collaborate to help each other in enhancing self-managed services as in the case of higher resilience, reliability, scalability, and dependability. For example, if a cloud provider has an emergency other cloud providers can temporarily provide their unoccupied resources to support customers eliminating service failures. Self-managed services must consider the existence of cloud-of-cloud, and it must also be designed to enforce cloud provider related policies when considering a decision to use other cloud resources, as this would have a major impact on security, practicality and legislation related issues.

3.2 Requirements

We are still working on identifying the requirements.

3.3 Meeting the Requirements

We are still working on these.

4 Conclusion

Current cloud infrastructure does not provide the full potential of automated self-managed services, and relies on cloud's employees (system architects, system and security administrators) to support the virtual infrastructure. In this paper we present a conceptual model of self-managed services in the cloud. These help in understanding the required functions and their interdependencies when providing self-managed services in clouds' infrastructure. Also, these help in realizing the challenges involved in providing automated management functions of clouds' virtual infrastructure. We have just started working on this as part of EU funded TCloud (Trusted Cloud) project.

5 Acknowledgment

This research has been supported by the TCloud project[2], which is funded by the EU's Seventh Framework Program ([FP7/2007-2013]) under grant agreement number ICT-257243.

The author would like to thank Andrew Martin and Matthias Schunter for their discussion and valuable input.

References

[1] Amazon. Amazon Elastic Compute Cloud (Amazon EC2), 2010. http://aws.amazon.com/ec2/.

[2] http://www.tclouds-project.eu

[2] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the Clouds: A Berkeley View of Cloud Computing, 2009. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf.

[3] IBM. Autonomic computing, 2001. http://www.research.ibm.com/autonomic/.

[4] Keith Jeffery and Burkhard NeideckerLutz. The Future of Cloud Computing — Opportunities For European Cloud Computing Beyond 2010.

[5] Peter Mell and Tim Grance. The NIST Definition of Cloud Computing.

[6] Microsoft. Microsoft System Center IT Infrastructure Server Management Solutions, 2010. http://www.microsoft.com/systemcenter/.

[7] Sun Microsystems. Take Your Business to a Higher Level, 2009.

[8] John D. Musa, Anthony Iannino, and Kazuhira Okumoto. *Software reliability: measurement, prediction, application (professional ed.)*. McGraw-Hill, Inc., New York, NY, USA, 1990.

[9] OpenSource. OpenNebula, 2010. http://www.opennebula.org/.

[10] OpenSource. OpenStack, 2010. http://www.openstack.org/.

[11] B. Randell, P. Lee, and P. C. Treleaven. Reliability issues in computing system design. *ACM Comput. Surv.*, 10:123–165, June 1978.

[12] VMware. VMware vCenter Server, 2010. http://www.vmware.com/products/vcenter-server/.