A NEW ARCHITECTURE OF AN AUTONOMOUS SYSTEM IN CLOUD COMPUTING

Shahad Alqefari
shahad393@hotmail.com

Soha S. Zaghloul
smekki@ksu.edu.sa

Faculty of Computer and Information Science
Department of Computer Science
King Saud University
Riyadh, Saudi Arabia

ABSTRACT

The continuous increase of network devices complicates the management task of appliances and services. The complexity and the dynamism of the current global network system require an interacting architecture. This should be able to autonomously change the system’s structure and functionality with little human intervention. Moreover, the cloud elastic nature necessitates the frequent need to a dynamic reconfiguration of the offered services. This is useful in case of unpredictable failure such as system crashes, network problems, or natural disasters that typically cause service unavailability. This paper proposes a new network system architecture based on cloud computing for service management. The suggested architecture is built using two architectures; namely, the autonomic SOA framework and the distributed autonomous control framework. The autonomic SOA framework extends the intelligence and capability in the cloud by using case-based reasoning as well as the architectural consideration of autonomic computing paradigm. On the other hand, the main feature of the second framework is the classification of the services into basic real-time and sophisticated non-real-time applications. The former are realized through M2M (Machine-to-Machine) communication between the devices under the local network area; whereas the Internet manages the non-real-time applications and controls the supporting devices.

KEYWORDS

Autonomous; Cloud Computing; Service Oriented Architecture; Distributed Autonomous Control.

1. INTRODUCTION

The development of the Internet enabled access to many types of services over the web, networked and distributed systems that provide both resources and services. Nowadays, such types of services gain an increasing demand. Hence, current distributed systems provide dynamism thus they become more complex.

The Cloud Computing (CC) paradigm represents an efficient solution for the provision of computational resources as services. CC is an on-demand computational model that provides the required services to the end-users. The offered services are provided through the Internet; they are characterized by their flexibility and scalability. In fact, CC is considered to be the extension of the Service-Oriented Architecture (SOA) to autonomous systems.

Therefore, advanced solutions are to be developed such that they dynamically adapt to the cloud elasticity. However, continuous service is to be ensured, and high performance level is to be guaranteed. Furthermore, some of the network services depend on the quality of the cloud environment. Therefore, any malfunction in the network, such as traffic delays or links failure, drastically affects the quality of service (QoS) in the cloud. In sensitive scenarios, such as military or medical applications, this is considered disastrous.

This paper presents a new architecture that consists of self-organizing elements. These components manage their own behavior, the internal services, and their relationships with other elements. The main advantage of such framework is its ability to cope with unpredictable events that may cause services unavailability such as system...
crashes. In addition, the proposed architecture is independent of the network quality of network, since urgent services are allowed to run on the local area network (LAN) autonomously. Therefore, the continuity of services is always ensured even if the network is disconnected. Therefore, the suggested framework is highly robust, adaptive, and autonomic.

The rest of this paper is organized as follows: Section 2 gives a background about both CC and Autonomous Systems (AS). Section 3 discusses the related work. Section 4 proposes the new framework. Finally, Section 5 concludes this study.

2. BACKGROUND

In this section, an overview covers both the fields of cloud computing (CC) and autonomous systems (AS).

2.1 Cloud Computing

According to the National Institute of Standards and Technology (NIST), cloud computing is defined as “a model for enabling ubiquitous, 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” [1]. The CC model is composed of five essential characteristics; here they are:

- **On-demand self-service**: A consumer can unilaterally possess computing capabilities – such as server time, networking, and storage – as needed without requiring human interaction to connect with the service provider.

- **Broad network access**: Services are available over the network. They are accessed through standard mechanisms that promote use via heterogeneous thin or thick client platforms – such as mobile phones, tablets, laptops, and workstations.

- **Resource pooling**: The provider’s computing resources are pooled to serve multiple consumers using the multi-tenant model. Different physical and virtual resources are dynamically assigned. Moreover, the assignment is rescheduled according to the consumer’s demand. In other words, customers have a sense of location independence; they actually have neither control nor even knowledge about the exact physical location of the provided resources. Resources include storage, processing, memory, and network bandwidth.

- **Rapid elasticity**: Resources are elastically provided and released in an automatic way. They rapidly scale outwards and inwards according to the users’ demands. From the user’s perspective, resources are unlimited and are always available.

- **Measured service**: Cloud systems optimize resources use by providing a metering feature at some level of abstraction. The latter depends on the type of resource. Therefore, resources usage are monitored, controlled, and reported. Thus, transparency about the utilized services is provided to both the provider and the consumer.

2.2 Autonomous Framework

Autonomic computing is a concept inspired from biological systems. Such systems aim to overcome the complexity of managing wide-spread heterogeneous nature. This is accomplished by automating self-management capabilities. Therefore, human intervention is minimized in the administration of large systems. The autonomic computing paradigm was first introduced by IBM in 2001 [2].

A system may have one or more self-* properties that enable autonomic capabilities. These are:

- **Self-configuration**: this is the ability of the system to perform configurations automatically according to some pre-defined high-level
policies. This should be done seamlessly as much as possible.

- **Self-optimization**: this is the ability of the system to continuously monitor and control the available resources with the aim of improving the overall performance.

- **Self-healing**: this is the ability of the system to automatically detect, diagnose and repair the faults that may take place.

- **Self-protection**: this is the ability of the system to proactively identify and protect itself against any malicious attack or cascading failure. Such faults are not covered by self-healing measures.

Autonomic systems require high-level human guidance to decide the steps that should take place in response of specific events. Therefore, an automated system constantly adapts itself to the changes that occur in the environmental working conditions. Similar to biological systems – such as human body – AS maintain their state; then, they adjust the undertaken operations with the consideration of the changing components, workload, the external conditions, hardware, and software failures [3].

Usually, AS include one or more managed components such as QoS elements and an intelligent control loop. Figure 1 illustrates the components of an AS: the autonomic manager administrates the elements’ states and behavior. Thereby, any sensed change in the managed resources invokes the appropriate set of actions necessary to preserve the targeted system’s state.

Typically, control loops are implemented as MAPE functions (Monitor, Analyze, Plan, Execute). The monitor collects information about the states of the system and prepares them for analysis. If any deviation from the targeted state is discovered during the analysis phase, then the planner change the plans accordingly. New plans are then submitted to the executor. The four components of an AS are discussed in more details in the following subsections.

- **Monitor**: The manager component monitors its own behavior as well as that of the overall system. Examples of the monitored criteria include the services availability, the addition/removal of services, and the requests issued from the user. A sentinel provides monitoring services – such as registry and discovery – to the AS monitor component. The monitor continuously detects the status of the system and identifies users’ requests. A user’s request is submitted to the analyzer component. In case a change in the service status takes place, it is reflected in the knowledge base.

- **Analyzer**: The manager uses historical data, or cases, stored in the knowledge base to perform its task. Such data include information about the service such as its description, type (atomic/composite), provider and access. Cases may be revised if necessary.

- **Planning**: Actions are planned to provide the requested service. For composite services, the action plans include information about the required atomic services such as the list of available needed atomic services, their locations, the way to access them, and their sequence. In addition, the planner updates the knowledge base if new plans are to be created or revised. This speeds their access in the future when needed again. The created/revised action plan is then forwarded to the execution module.

- **Executing**: Plans are therefore executed in order to provide the required services. Brokers assist in the interaction with the service

![Figure 1. Structure of an AS](image-url)
providers to obtain the required services. For example, brokers may translate messages between incompatible protocols of the sender and the receiver. Note that if the requested service is atomic, then it is simply allocated by the service provider. If it is a composite service, then it is decomposed into the atomic services based on Business Process Execution Language (BPEL) or Choreography Description Language (CDL).

3. RELATED WORK

In this section, two autonomous frameworks from the literature are exposed. These are specifically selected since they are both the basis of the suggested architecture presented in this paper.

3.1 Autonomous SOA Framework into CC

In [4], the authors adopted the SOA framework of cloud computing. The contribution they made is the incorporation of autonomic computing paradigm in such extension.

The architecture is divided into three tiers as follow:
- The top tier is the presentation layer. It is the interface to the users through the web.
- The middle tier is the processing layer. Its role is to perform jobs and to coordinate between several tasks.
- The bottom tier is the service/resource layer. It actually enables the utilization of the distributed resources via Web services.

Figure 2 illustrates the autonomous SOA framework. The services layer corresponds to the service providers in a typical SOA framework.

The brokers in the processing layer act as service requestors. In addition, a knowledge base is added in the middle layer to extend the functionality of the service registry. This is a necessary requisite for an autonomic computing paradigm. Also, the autonomic computing paradigm is incorporated in the middle layer.

The presentation layer includes the end users who access the system through the Internet interface.

The autonomous manager performs the MAPE functions. The details are explained in the following paragraphs.

Once a service request is received, the analyzer starts searching for the corresponding profile in the knowledge base. If not found, then cases with similar features are retrieved. In this context, various metrics are used to calculate the similarity distance. The retrieved similar cases are then submitted to the planner. The latter revises the actions and generates a new plan accordingly. This is then added to the knowledge base. In case there is no similar case, then the monitor module searches in the composite services for similar atomic services. If not found, the system looks other service registries – such as online service registry – thus achieving system scalability.

Finally, the retrieved service is then used for action planning and saved in the knowledge base for future use. Moreover, the autonomic manager suggests other related services to the users based on the historical data stored in the knowledge base.

In summary, the autonomous SOA framework highlights the use of case-based reasoning with AS in a SOA architecture with CC.
3.2 Distributed Autonomous Control in CC

In [5], the authors suggest another solution to incorporate AS into CC. Network services are categorized into basic real-time and sophisticated non-real-time services. The former is controlled on the local area network (LAN) via a machine-to-machine (M2M) communication; whereas the latter is managed through the cloud. This architecture facilitates the system installation. In addition, it is independent of the quality of the network since urgent services may run on the LAN autonomously even if the Internet is disconnected. Figure 3, illustrates such architecture.

Basically, the architecture is based on a cloud and a LAN. Moreover, a distributed autonomous control is incorporated into the cloud. Primarily, services are executed as much as possible on the cloud in order to reduce costs. Only services classified as urgent real-time are allowed to be processed in the distributed autonomous control provided in the LAN. More details are given in the following paragraphs.

The most important point to start with in this architecture is to define the communication protocol used in the distributed autonomous control. Devices use the defined protocol to communicate with each other. Therefore, the pre-defined protocol is installed in the communicating nodes.

The architecture shown in figure 3 consists of two main components; namely the smart outlet (SO), and the Distributed Autonomous Control File (DACF).

- **Smart outlet**: The SO has two main roles. The first role is to send the device information to the cloud platform. SOs should use secure communication. In general, a RFID communication is used. The second role of the SO is to manage the distributed autonomous control between household devices. Therefore, SOs communicate with each other; thus, control all connected devices.

**Figure 3. Distributed Autonomous Control Framework**

- **Distributed autonomous control file (DACF)**: SOs operate based on the DACF, which is analogous to an executive instructions file. The order number and the control information are described in pairs in the DACF. If there are parent-and-child nodes, only parent nodes are connected to the SO. However, the children nodes are controlled by the SO.

4. THE PROPOSED ARCHITECTURE

The main problem encountered in the autonomous SOA framework is the dependency of the services on the quality of the network. In other words, if the network is disconnected, or even if the traffic is only delayed, the quality of offered services falls off. In fact, this is considered a critical situation in some applications. In particular, military and medical services might lead to a disaster in case of such failure; it might cost the lives of human beings.

Furthermore, clouds are subject to many types of disasters. Natural disasters include earthquakes, volcanoes, etc… Technical disasters include an overall system crash or a power outage. Human-induced disasters range from hackers to terrorists attacks. In case of a disaster, services are automated in an attempt to move the virtual machines with the running applications to a far-
away safer server. Thus, the network traffic increases with the aim of controlling many home devices simultaneously. This may lead to a network congestion resulting in a drastic decrease in the throughput. In extreme cases, the whole network may fall down.

On the other hand, the distributed autonomous architecture offers many advantages such as agility, maintainability, reusability, consistency, efficiency, integration, and economic services costs. However, the framework lacks adaptability, robustness, and dynamism. In fact, it is unable to organize the services delivered by the cloud. Moreover, it does not consider changing the functions included in the services. Furthermore, the creation of new functions is not taken into consideration. These characteristics should in fact be available on higher emergency levels, with less or no human intervention. This is critical since such environments are highly sophisticated.

In fact, services delivery, discovery, and composition are a few of the many challenges faced in an autonomous CC environment. Other difficulties include the adaptation to incongruent services, and the creation of a new service. It should be guaranteed that all services, especially the composite ones, do work properly. They should be robust, secure, and satisfy the required QoS level. Network problems should also be carefully considered.

Therefore, an autonomic system is an essential requirement to a cloud environment. An AS provides the cloud with dynamism, flexibility, adaptability, and independence of network problems.

It is found that combination of the advantageous components of the two previously systems is expected to achieve the target cloud characteristics. Like the distributed autonomous control framework, the proposed architecture classifies the services into basic real-time and sophisticated non-real-time ones. Therefore, a LAN is needed to run the critical non-real-time services; thus, achieving the goal of network problems independence.

On the other hand, the framework of the autonomous SOA is adapted to be built on the top of the LAN. This provides a powerful autonomous manager based on the historical data stored in the knowledge base. Therefore, services handling is more reliable. The upper layer is therefore responsible for basic real-time services execution. Figure 4 illustrates the proposed architecture.

5. CONCLUSION

In this paper, a new framework of a cloud environment is proposed. The suggested architecture merges the concepts of SOA, autonomic computing using knowledge-based reasoning, and distributed autonomous control.

Services are classified into two categories: basic non-real-time and critical real-time. On the other hand, the architecture is divided into four layers. The presentation layer is the interface of the cloud
to the end-users via the Internet. The processing layer consists of the autonomous manager, the knowledge-based, and the brokers. The non-real-time services layer run the basic services, therefore, a network failure doesn’t have a disastrous effect on services running. Finally, the real-time services layer, or the home network (LAN), in which the critical services are running using the distributed autonomous control. Since this is independent of the whole network environment, then the failure of the cloud does not affect the running of such services. As a result, a more reliable system is made available.

6. REFERENCES


