Taxonomy of Autonomic Cloud Computing

Hamed Hamzeh[1*], Sofia Meacham[1], Botond Virginas[2], Keith Phalp[1]
[1] Smart Technology Research Group (STRG), Bournemouth University, Bournemouth, UK.
[2] British Telecom, Ipswich, UK.

* Corresponding author. Tel.: +447585525190; email: hamzehh@bournemouth.ac.uk
Manuscript submitted January 10, 2014; accepted March 8, 2014.
doi: 10.17706/ijcce.2018.7.3.68-84

Abstract: Cloud computing is a paradigm that has become popular in recent decade. The flexibility, scalability, elasticity, inexpensive and unlimited use of resources have made the cloud an efficient and valuable infrastructure for many organizations to perform their computational operations. Specifically, the elasticity feature of cloud computing leads to the increase of complexity of this technology . Considering the emergence of new technologies and user demands, the existing solutions are not suitable to satisfy the huge volume of data and user requirements. Moreover, certain quality requirements that have to be met for efficient resource provisioning such as Quality of Service (QoS) is an obstacle to scalability. Hence, autonomic computing has emerged as a highly dynamic solution for complex administration issues that goes beyond simple automation to self-learning and highly-adaptable systems. Therefore, the combination of cloud computing and autonomics known as Autonomic Cloud Computing (ACC) seems a natural progression for both areas. This paper is an overview of the latest conducted research in ACC and the corresponding software engineering techniques. Additionally, existing autonomic applications, methods and their use cases in cloud computing environment are also investigated.

Keywords: Autonomic computing**,** cloud computing, middleware resource management.

1. Introduction

Cloud computing [1] is a novel technology that enables the organizations and commercial sectors to have a large, cost-effective, unlimited resources, on-demand access to computing resources, scalable, and elastic computing infrastructure. This paradigm provides three main services for the users. Infrastructure as a Service (IaaS) which is applicable for delivering Software systems and applications over the net. It also includes IT infrastructures such as servers, virtual machines, network storage facilities. The platform as a service (PaaS) is related to the Software applications for testing, developing and managing them and Software as a Service (SaaS) which allows users to use different applications in the cloud. Although, cloud offers the above-mentioned facilities for the users, there are several limitations by taking into account the complexity of cloud infrastructure and QoS. Hence, a new management and operational system requires to deal with those challenges in existing cloud infrastructure. In a cloud environment, different applications require different aspects of QoS such as response time and Throughput. Accordingly, an autonomic management system can be applied for each element in the cloud system [2]. This paper concentrates on the autonomic management in the cloud environment by investigating and analyzing the latest findings in conducted research works related to ACC.

The main contributions of the paper are as follows:

- A taxonomy of AC and the management techniques in the cloud by focusing on the latest research works.
- Reviewing the existing techniques and platforms in designing and implementing of autonomic cloud-based applications.
- A review of existing autonomic cloud applications.

The rest of the paper is organized as follows. Section 2, discusses about autonomic computing and some related concepts along with its architecture. Section 3 covers the basic concepts of ACC. In section 4, we investigate different works related to ACC and provide a taxonomy of it in different perspectives such as multi-agent systems, middle-ware, resource provisioning and scheduling, multi- tier applications. Section 5, focuses on existing autonomic cloud applications like Qubell and Comet cloud. Finally, section 4 refers to different platforms and applications that can be used to deploy and implement ACC systems.

2. Autonomic Computing (AC)

The idea of AC emerged in 1970 to enhance the availability and functionality of mainframe systems at resource level to enable IT administrators to install some parts during system run-time [3]. By introducing distributed systems, automatically managing and monitoring solutions were proposed at system level. However, the mentioned ideas were not standardized and they could not meet the business policies. Additionally, the concept of AC has been introduced by Paul Horn [3] as a vice president of IBM in 2011 to deal with the complexity of industrial systems. The main focus of the system proposed by Horn was based on automatically analyzing, responding and sensing to assist the management and operational tasks. The main characteristics of AC systems are generally recognized as self-managing, self-configuration, self-optimization and self-healing.

Self-management is the essence of an AC system [4] which is broadly applied in computer network systems and especially mobile adhoc networks. Self-configuration is the ability of a system to configure itself by using high-level rules and regulations. Self-optimization is the capability of the system to enhance its actions and operations in an efficient way. Self-healing is the ability of the system to identify and discover the problems and errors recognition and fixing them automatically in a very short time. Last but not least, it should be noted that the base of AC is Artificial Intelligent (AI), however, it doesn't mean that it entirely eliminates human intervention. Indeed, more research works are necessary to be performed to develop algorithms include the self-* features of AC that can make correct decisions without human intervention.

3. Autonomic Cloud Computing (ACC)

In previous sections, we discussed about the advantages of using Cloud infrastructure which provides flexible, easy to use and scalable facilities. However, by taking into account the complexity of computing operations and high volume of workloads generated by users and applications, more innovative solutions are required. AC is the natural candidate as it has been successfully used in the past to handle complexity in computing administration tasks. ACC [5] is the management of a cloud system that needs the least human intervention and providing accurate results as well. These kinds of systems are able to adapt themselves against changes in the nearby environment in order to keep the required SLA and QoS. The main purpose of using ACC is resource scheduling for appropriate workloads and increasing the effectiveness of resource utilization. ACC is very beneficial and efficient approach to decrease the energy and time consumption in cloud data centers and computer networked systems.

It mainly focuses on self-management capability. As shown in Fig. 1, Autonomic Management System (AMS) has been implemented in the PaaS layer. The application scheduler within AMS, is in charge for determining where each application element will be executing. Dynamic resource provisioning is composed of different algorithms in order to meet the QoS requirements to improve the efficiency and reduce costs.

Workflow control is important to enhance the functionality of the system by increasing its throughput. Moreover, the workload adaptation is another important factor in autonomic environments which is a performance management process. Autonomic management system includes an autonomic manager which is capable of automating the management and functions by applying MAPE-K which is a knowledge-based loop consists of monitoring, analyzing, planning and executing by using different policies for example for resource provisioning [6]. The control loops are able to be in contact with each other inside or outside of the AC infrastructure. So, by collecting the information from the system and using real-time data, a control loop can decide to whether address the issues and adjust the system in a proper situation. Hence, it collects all the application requirements in favor of user context knowledge-base. Multiple autonomic managers are able to be coordinated in a cloud environment which is proposed as a platform in [7]. The aim of this framework is to enhance the synergy among AMs by decreasing intrusion created by one AM on systems managed by other AMs. In a large and distributed system, it is possible to divide the tasks like monitoring, management and control among the autonomic managers which leads to improve the performance and increase the scalability of the system [8].

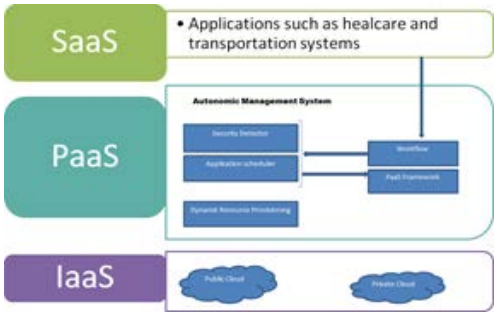

Fig. 1. Magnetization as a function of applied field.

4. Taxonomy on Autonomic Cloud Research

The application of ACC is varied in different perspectives and areas such as de- signing an application as a middle-ware to adapt in a cloud environment to use autonomic capabilities in optimizing and improving its infrastructure. In this section, the high-level classification of ACC is presented from different perspectives such as QoS, SLA, workload management and scheduling, resource allocation and provisioning. Specifically, in each subsection the recent conducted research works along with their evaluation methods are reviewed. The summary of important research works are categorized in Table 1.

4.1. Multi-agent Systems and Cloud

Multi-Agent Systems (MASs) [9] refer to a new technological paradigm in order to tackle the problems associated with large-scale distributed and software systems. MASs are a kind of collaborative agents based on interactions among each other to solve the related issues by using the knowledge of every agent. Moreover, MASs consist of a heterogeneous and distributed software architecture which makes them an appropriate solution for designing and developing the software applications for distributed architectures. MASs are conceived as an efficient model to address the complexity and dynamic problems of distributed systems by using a series of agents that are independent in behaviors [10]. In terms of cloud computing and in the traditional architecture there are some issues regarding the scheduling and management efficiency that multi-agent systems can address them by using the Multi Agent Cloud (MAC) [11]. The structure of MAC can be defined in two levels. Local Agent (LA) and General Agent (GL). Accordingly, GA is the main part of a MAC and it gets the requests from the users. In order to manage the cloud infrastructure, LA is introduced to manage the resources and synchronize the servers. This structure is salable to use in very

large cloud systems. In recent years, the combination of AC, MAS and MAC are used to achieve more efficient system management solutions, especially in large-scale networked systems and virtualization. In this paper we aimed to review and evaluate the recent conducted research works related to ACC in different areas.

B. Othmane and R. S. A. Hebri in [12] proposed a novel and cloud-based approach for distributed data mining by utilizing the MASs and the cloud computing infrastructures. An agent-based system is proposed which operates based on SaaS. The work concentrates on Distributed Data Mining (DDM) and using MAS to enhance its functionality. The advantages of using MAS in DDM are described such as dynamicity in data collection and data mining, the stability of autonomous data sources and high scalability. By using the parallelization capability of the cloud, the response time is also decreased. However, the work is in the implementation phase and there are no results and evaluations to judge about its effectiveness.

M. Ayyub, Y. Jararweh, M. Daraghmeh and Q. Altebyan proposed a model in [13] to optimize the resource provisioning tasks by relaxing the completion time of them and decreasing the costs as well. The conducted research also describes the dynamic resources provisioning and monitoring (DRPM) system which is a multi-agent system in order to manage the cloud resources by considering the QoS requirements. DPRM involves two basic layers. The customers applications layer and cloud providers resources layer. A global utility agent and a set of local utility agents are the main elements of the system. The main benefit of using MAS is to improve the SLA and resource utilization. Two new algorithms are proposed at this work. Host Fault Detection (HFD) to specify VM migration in terms of overloading the hosted physical machine. Without happening SLA violation, the local agent specifies the number of resources that will really be utilized. This operation is performed by using the Request Reformulation (RR) algorithm which applies regression analysis to specify the number of resources. For evaluating the proposed model, CloudSim tool is used as a simulation environment. The results present that the DRPM system permits cloud provider to increase resource utilization and decreases power consumption while avoiding SLA violations. It is indicated that HFD has not a big effect on energy consumption metrics and it is required to optimize it to be more efficient and applicable. However, the advantages of HFD in energy consumption, VM migrations, SLA violations and host shutdowns are undeniable. One of the main disadvantages of DPRM is SLA violations in term of costs. Also, HFD outperforms some specific selected policies but it is not much more positive against other policies. In some circumstances, HFD outperforms other policies but it is being to be outperformed by others.

A. Kumar, A. Tayal, S. Kumar and B. Humadhava, *et al*., proposed in [14] by considering multi-agent autonomic infrastructure. The proposed architecture involves a controller which interacts with the agent service manager that includes autonomic manager for monitoring, analyzing and executing operations. The agent service provider gets the information from the agent service manager to provide the suitable services for requests. Within the agent service manager there is a local registry which maintains available agent services. Other components into the Agent service managers are: autonomic manager, Monitoring subsystems and knowledge management subsystem. The autonomic manager has interaction with servers including WSDL, DDI, ACL, SOAP. The transactions within the server must be registered, updated and searched into the UDDI. The proposed architecture is applicable in enterprise systems and operations such as load balancing, data management, resource monitoring, etc.

The work conducted in [15] proposed a method according to multi-agent system for Cloud of Clouds elasticity management which aims to reduce the complexity of modeling and designing the cloud elastic system in both inter and intra cloud levels by using ClubFansApp which is being used in a football club. Three different scenarios are considered. The first one is the achievement of an optimal point of the system by solving the problem of overloaded requests using an elasticity solution. In the second scenario, solving the limited resources has been taken into account by replacing a resource using configuration by less

resource using configuration. Third scenario deals with additional service requests. A Multi Agent System for Cloud of Clouds Elasticity Management (MAS-C2EM) by using MAPE-K loop has been introduced to address the mentioned problems in mentioned scenarios. In the model, two agents are proposed. Resources Surveyor Agent (RSA) which is responsible for monitoring the computational metrics. Another one is the Service Surveyor Agent (SSA) which is responsible for evaluating and monitoring service metrics. C2EM model in the literature considers only the static view of the system without taking into account the dynamic behavior the elements

As a conclusion, the research works conducted in MAS targets different aspects of QoS by utilizing the functionality of AC by focusing mainly on the monitoring phase of MAPE-K strategy. Resource utilization, dynamic data collection, load balancing, reducing energy consumption and complexity that are the important achievements of those works.

4.2. Autonomic Middleware Solutions

The target of some research works is the development of appropriate middleware solutions that mediate between applications and the cloud. By increasing the number of subscribers and accordingly huge volume of workloads generated by them and applications, it is necessary to reduce the complexity of the computing tasks by giving the priority to design and implement autonomic middle-ware systems. The aim of designing middleware is to increase the efficiency, flexibility, reliability and decreasing costs. An AC middle-ware is subjected to have all the features or at-least the main components of AC infrastructure, such as self-management, self optimizing, self-healing and self-protection so that the autonomic manager is responsible for realizing those mechanisms.

W. Trumler, F. Bagci, J. Petzold and T. Ungerer proposed an autonomic middle-ware (AMUN) for ubiquitous environments [16]. It utilizes an intensive monitoring on different phases to gather the information required for the metrics to calculate the QoS used to operate the self-* features. The objective of AMUN is to simplify the operation of ubiquitous systems on a distributed peer to peer foundation. In that research, a flexible office environment (Smart Doorplate application) by considering different scenarios has been assumed. According to the IBM AC principles, monitoring and self-mechanisms, initial self-configuration, self-optimization during runtime and Self-healing due to service failure are taken into consideration. The main components of AMUN are as follows. Transport interface, event dispatcher, service interface and service proxy, autonomic manager, monitoring, transport interface monitoring and monitoring at the event dispatcher.

P. Lalanda, D. Morand and S. Collet, *et al*., proposed the Cilia [17] which is designed to support the requirements of cyber-physical systems. It is based on service-oriented elements in which delivers dynamic and autonomic characteristics and permits very good flexibility at runtime by considering little human intervention. The most important contributions for this work are: 1) reporting the unusual conditions to avoid performance degradation. 2) discovering slow derivations. 3) bandwidth optimization in order to reduce the energy consumption. The framework is constructed by components known as mediators. Each mediator has the responsibility of synchronization, routing and processing. Cilia is created on top of the Open Service Gateway Initiative (OSGi), Java based objects. Cilia offers Domain Specific Language (DSL). In a Cilia infrastructure, there are Cilia chains composed of adapters, mediators and java-based local communication. The autonomic manager deals with knowledge-based component regarding the Monitoring and adaptation directives. In terms of data processing, adaptation, feedback and the knowledge-based component are in interaction with Cilia. One of the use cases presented in this work is pumping stations for water management. In respect of the integration of Cilia, the HMI is used. The expected results of implementing the Cilia framework are: 1) The validation of new added-value services 2) Presenting the high

functionality of industry 4.0 3) Applying the Cilia in scientific and technological frames 4) Implementing it in a way that it can be usable by non-experts 5) providing a self-adaptivity approach to deal with data variations. Taking into account that the autonomic manager does dynamic updates, so there is an overhead especially in execution time. Also, creating the mediators is costly and also, replacing them is a time consuming action.

The research conducted in [18] proposed an autonomic service-oriented middleware AUSOM for IoT based systems according to the MAPE-K loop for changing application requirements and contextual changes in the environment. The proposed model is a multi-layered context model which is able to get the contexts in different layers involving middleware, resource and application to make the adaptation decisions. They have considered a farm alarm autonomic system that work flows are used in order to create the application commands. The structure of the model is based on monitor, analyze, plan, execution and knowledge modules as the heart of the system. Workflow repository, application command set, device capability set and context manager are also introduced in this model. In analyze module, three possible actions are examined including new workflow generation, eliminating unnecessary workflows and modifying the existing workflows. Since, workflow generation leads to a large latency, memoization of frequently used workflows is proposed. For the planning phase, AI planners are taken into account and for execution phase where the synchronization is an important factor, workflow description languages are utilized.

As a conclusion and according to the structure of middle-ware systems, it seems that the proposed systems are responsible for the whole components and structure of MAPE-K loop. So, the system is mainly responsible for self-* properties. According to the conducted research works in this section, simplifying the operations, flexibility at runtime, avoiding the performance degradation and bandwidth optimization are taken into consideration.

4.3. Autonomic Resource Management and Dynamic Resource Provisioning

Resource management in the cloud environments is a challenging issue with respect to the high volume of subscriptions from the users. VM management, clustering, scheduling and such related issues have led to many problems in managing the existing cloud infrastructure. So, there is a requirement to tackle the mentioned problems by utilizing the autonomic approaches that originated by Software engineering techniques. In a cloud layered structure, the PaaS layer is responsible for resource provisioning. Over-provisioning and under-provisioning are two important factors may lead to SLA violations and degrading the QoS level. To address these sorts of problems, the dynamic resource provisioning (DRP) techniques are utilized. DRP approaches are originated from MAPE-K loop that consist of: Monitor, Analyze, Plan and Execute. The basic working scheme of DRP is achievable by scaling up or down the resources via a sort of algorithms, rules, and policies to have a better correspondence to the existing resources with current workloads and user demands.

B. karakostas, *et al*., [19] proposed a prototype autonomic manager in the cloud built on top of Juju which is a cloud service orchestration and deployment manager. It takes autonomous decision to scale cloud services to enhance the overall performance. The high level rules and regulations are decided by Juju autonomic manager to scale the cloud resources and meet the SLA and performance requirements. Juju has a monitoring environment for service deployment along the multiple cloud and physical servers for the orchestration of these services. The monitoring feature of Autojuju involves monitoring the status of cloud deployment by utilizing Linux commands. The autonomic manager in auto Juju is responsible for monitoring Ubuntu OS and Juju/OpenStack. It uses the available local charm store database to store the metadata. For the evaluation, a wordpress application has been used bearing in mind that Autojuju has the knowledge of all dependencies so that by adding a service into the deployment process all the relationships

and dependencies will be loaded autonomously. For every new service instance, a new machine has been added. However, it is more economical if multiple instances use in the same machine which has been considered in Auto Juju. One of the most important factors used in the Autojuju is the built-in delay in order to prevent frequent termination or creation of instances in the time of variations in workloads. A drawback of the work is to apply simple detection, respond policies and rules that can be improved in future works.

The work presented in [20] proposed a distributed autonomic management framework for cloud computing orchestration. The targets of the research work are: 1) Dealing with the heterogeneity, dynamism and stochastic nature of the cloud. 2) A framework to simplify the enhancement and formation of tools to manage and optimize cloud environments. 3) Emphases on the management of clouds that offer IaaS. 4) The proposed model allows each cluster to be managed by any management agent at any time. 5) It promises a harmless and efficient execution, while other agents are able to explore and manage different clusters at the same time. In order to design the platform, the designers have created a cluster manager which is in charge for running the migration of VMs and shut down the idle services when required. The proposed algorithm gives the priority to the higher capacity hosts and shuts the ones by considering the lower capacity. Additionally, clusters with extra idle resources have great importance of assigning VMs and starting hosts when required. Three important factors are used in heuristics: memory, CPU frequency, numbers of cores. In the experiments, the model gives the priority to the hosts with higher capacity and powers off the lower capacity hosts. The workload type is selected as static such as email and Git. In this case, the number of servers can be estimated to shutdown. By identifying the servers, the model is able to power them off and reduce the number of running servers. So, the allocation manger investigates hosts to commence in clusters with higher capacity. First of all, the clusters are analyzed by the heuristic algorithm and then the hosts are analyzed. In terms of VM allocation, whenever the framework requires to start a host, the VM allocation delay will be high. The delay for a specific source in the cloud environment can be relaxed by using SSD to achieve a faster boot. In the context of the research, the author mentioned that if one worker starts to work, other workers will not work. It is stated that this is for more simplicity in terms of designing and developing. This is a problem when there is necessary to solve the conflict issues among the agents.

The work proposed in [21] presents an autonomic power and performance management technique for cloud systems to dynamically match the application requirements with just-enough system resources at runtime which contributes to significant power reduction while meeting the QoS requirements of cloud applications. The proposed model delivers: 1) Real-time monitoring and describing of the workloads to specify the cloud resources that meet their QoS requirements. 2) Specifying the running operating spot of both workloads and the VMs running these workloads. 3) Allocating the existing cloud resources that can warranty the optimal power consumption. 4) Dynamically scaling up and down the VM resources within run time. 5) Controlling several resources such as number of cores, core frequency, and allocated memory at the same time. The proposed model has been compared to different resource management methods such as Autonomic Performance-per-watt Management (APM), static resource allocation and frequency scaling method by using RUBiS web application regarding the execution time overhead and VM power consumption reduction. By increasing the number of clients, power consumption in the new model is significantly lower than other methods. The output of the research contributes to decrease in power consumption up to 87% in a situation compared to the static resource allocation approach, 72% compared to adaptive frequency scaling strategy and 66% compared to a similar multi-resource management strategy. In the research, different features of the workloads are monitored such as CPU, Memory, Network and Disk. Monitoring information are stored in a database by using timestamps. The main targeted area of the work is web application workloads by considering different number of clients from low to high. Additionally, rule-based

data mining techniques are prioritized because of their efficiency in interpretation in characterization the input data. The execution time in the worst case is up to 3.27% and 87.15% reduction in power consumption. Hence, the proposed model outperforms other methods and offers a major reduction in power consumption with a small execution time overhead

N. K. Salih and T. Zang, *et al.,* in [22] proposed an autonomic cloud environment to assist management of health-care services through designing a complete monitoring system to decrease the costs by using the distributed services. The system architecture of the proposed model is considered three layer services in the cloud which starts with the SaaS layer and using the software system which uses a web service. In the PaaS layer, an Aneka-based computing cloud has been used to control the software execution which brings a management scheme for the healthcare system. In the IaaS layer, the capability of the Aneka is utilized to retrain different resources in the cloud. The major components of the Aneka consist of master, worker, manager and user. The heart of the autonomic management is MAPE loop. Service Oriented Architecture (SOA) is used to reduce the costs and improve better data administration. SOA is a business IT- alignments method in which that applications may depend on services to simplify business processes. Self-analyzing and self-monitoring capability is considered in this work, so that a doctor is able to monitor the vast amount of data provided by cloud storage in order to from a single access-point. However, this research is not conducted as a real work and it also needs to be analyzed and maintained for QoS.

The work in [23] presents SLA-aware autonomic management of cloud resources (STAR) for reducing the SLA violation rate, improving user satisfaction and effective scheduling of resources which considers SLA violation rate along with other QoS parameters like execution time, cost, latency, reliability and availability. The research focuses on the main problems in cloud environments especially, uncertainty and resource dispersion. The primary assumptions in STAR are: 1) Allowing multiple user to access concurrently to the cloud system. 2) Workloads may have different execution time and deadlines. 3) Users have various QoS parameters that may be changed dynamically. In the working scheme of STAR mechanism, initially, cloud consumer attempts to perform the workloads via the Cloud Workload Management Portal (CWMP) such as a web based application. Once the authentication level finished, STAR requests to submit the cloud consumer requirements (SLA) and legitimated cloud consumer, loads it and submits the request for the accessibility of specific resource with appropriate requirement for the execution of their workload. STAR receives the information from the suitable workload after analyzing the numerous workload details which cloud consumer required. In designing such as system, the cloud consumer in SaaS layer is assumed who has interact with a cloud mobile application in PaaS layer. The workload information is provided by the user to the application and the results are given back to the user. The application may consists of a performance monitor, service manager, SLA-manager, QoS manager, Resource manager and workload manager. So, the application workload is sent to the resource scheduler in the IaaS layer for workload execution. By taking into account that there are different resources, so, the k-means clustering algorithm is used in order to re-clustering the workloads to execute on various sort of resources. For resource scheduling process, time and cost are taken into consideration. Hence, in designing phase, workload details, pricing plans and estimating the execution charges are considered. By comparing the results of STAR with other methods presented in the research work, in terms of SLA cost, the proposed model works better than other models 7-15%. Another performance metric is execution time, where the STAR performs better than other models (9.6% - 14) and (16%-21%). For the future works, the STAR will be extended for self-healing, self-protecting and self-healing. Most importantly, the parameters such as scalability, resource utilization, energy efficiency are necessary to be considered.

L. Hadded, F. B. Charrada and S. Tata in [24] proposed a method for optimization the autonomic environment. The main focus of the work is the Service- Based Application (SBA) which involves the

elements for the business services. The work also concentrates on the Autonomic Manager (AM) components to identify the exact number of AM components to optimize them in order to decrease their costs and increase the management performance by assigning AM elements to only one service to prevent the management bottleneck issue. For the optimization process, two algorithms are used. One of them is A/M assignment algorithm and another one is P-E assignment algorithm in which the first one is used for analyzing and monitoring and the second one is used for planning and executing. The minimum number of monitors are planned to be assigned to services by using the A/M algorithm. Different monitors are assigned at a time to services to avoid the management bottleneck. The quality of the work is assessed by using a linear model and CPLEX tool and a real database which is randomly created. As far as the SBA systems are shown as a directed graphs, so in this work, different graphs are used to implement different combination of services. In the experiments, the number of monitors, analyzers, planners and executors are saved up to 54%, 56%, 56% and 62% respectively. Moreover, the response time of the model is approximately 66 milliseconds and the optimal solution is more than 86%.

M. G. Arani, S. Jabbehdari and M. A. Pourmina, *et al*., proposed in [25] an autonomic system for resource provisioning of cloud services. The algorithm pursues the MAPE control loop and manages the Virtual Machines (VMs) that are delivered to each cloud service in any time interval. Also the papers aims to enhance the performance of the planning phase by using the learning automata (LA) as a decision-maker. In order to predict the future demands, a linear regression model has been used in workload analyzer. The CloudSim toolkit has been used to setup the experiments. The proposed model is compared by considering three base-line strategies: Cost-aware (LRM), which utilizes the LRM, and the second one is known as Cost-aware (ARMA), which uses a second-order ARMA model for the workload prediction. DRPM is the third strategy which is a multi-agent- based DRPM system for cloud computing infrastructure. Different performance metrics have been used such as cpu utilization, VM allocation, SLA violation, Total cost and profit. By using the proposed methods, there is approximately 40% reduction in SLA violations and 35% in cost saving. The advantage of using ARMA and LRM is that both of them have the ability to fully utilization of resources. In terms of utilization, ARMA performs better than LRM. However, in terms of SLA violation, LRM works better than ARMA. SLA violations are happened by increasing the number of workloads because of increasing the response time for many service requests. So, decreasing the response time can be solved in future works.

C. Pautasso, T. Heinis and G. Alonso, *et al*., proposed autonomic resource provisioning for software business processes [26] to present whether resource provisioning for software business processes can be solved by utilizing AC strategies. The research work has two main contributions. The first stage is, how to arrange execution of a software business process corresponding to a stage-based architecture. The second is how to utilize the AC capabilities to solve the management issues. A distributed engine is presented to execute the web services compositions. The autonomic controller used in this engine aims to monitor the workload and the status of the system. A simple workload along with JOpera are applied to evaluate the work and also an optimal configuration is considered to reduce the response time. In evaluations, total execution time and resource allocation are measured by using different workload sizes.

As a conclusion, using different resource allocation and scheduling algorithms in conducted research works are utilized to reduce the costs and SLA violations that contribute to increase the overall system performance. All those mentioned parameters are the basic requirements to satisfy the user demands. Using the scheduling algorithms and resource provisioning methods may lead to increase in overall management performance of the system by monitoring the behaviors of incoming workloads.

4.4. Using AC in Managing the Multi-tier Cloud-based Applications

The concept of multi-tier/n-tier architecture refers to a client-server infrastructure in Software Engineering. The layers include presentation, application and data management. The most widespread use of multi-tier architecture is the three-tier architecture. In order to have a flexible application, it is necessary to use the n-tier one. By dividing the application into different layers, a developer is capable of managing the systems in individual layers without requiring to re- configure the whole system. In a cloud environment where the Virtual Machines (VMs) are being used, different layers may run in different VMs.

R. Gergin, B. Simons and M. Litoiu, *et al*., proposed in [27] a decentralized autonomous architecture for multi-tier applications deployed in cloud environments. The aim of using this architecture is to maintain the Service Level Objective (SLO)(which is basically determined in a SLA) and reduce the costs. The architecture applies a set of autonomic controllers in which every controller regulates a tier of the application in an independent manner. The autonomic controllers are implemented by the Proportional-Integral-Derivative (PID) which is originated from the process control industry so that the purpose of utilizing it is to maintain the utilization of the cluster at a stable value. The architecture utilizes feedback loops and implements Proportional, Integrative and Derivative control laws at each autonomic controller. At this work, each tier is presented by a cluster so that the monitoring and controlling phases are done by using the feedback control loops. The autonomic controller has been implemented by using the Java classes which uses AWS SDK 1.2.15 library and mySQL database to store historical VM utilization observations. All the experiments are performed by utilizing the Amazon EC2 public cloud. A custom data mining web application which performs queries for a large database with millions of randomly generated and a workload generator according to the FIFA 1998 workload to simulate the user interaction with the application have been used. As a result, the main contribution of this work is to improve the response time in a specific time intervals based on HTTP requests. Load balancers in future works requires attention in terms of delays and it would help to improve the functionality of the proposed method.

V. Goswami, S. S. Patra and G. B. Mund, *et al*., in [28] proposed a dynamic autonomic resource provisioning scheme for multi-tier applications to meet the QoS and SLA requirements by using a queuing model to address the congestion problem associated with multi-tier systems by taking into account the mean service rate in VMs. The aim of this work is to develop an analytic model to minimize the total amount of VMs as well as the integration of load prediction method strategy to fit the workload features. A recursive model consists of SLA- based negotiations by applying the supplementary variable techniques are mainly used to evaluate the work.

Table 1. Taxonomy

Target(s)	Algorithms & platforms	Preformance metrics
Multi-agent Systems and Cloud		
[11] Optimization of resource provisioning by relaxing the completion time and decreasing the costs,improving SLA and resource utilization	-Host Fault Detection (HFD) - Request Reformulation (RR) -CloudSim	- Computational power - Memory - Permanent storage Network Bandwidth
[12]- Dynamicity in data collection -Decreasing response time - Improving the functionality	- Using the parallelization capability of the cloud	- This work is in implementation phase
[13] Reducing the complexity of modeling and designing cloud elastic system	-ClubFansApp	-CPU -RAM -Price

Autonomic Middle-Ware		
[15] The major aim of AMUN is to simplify the operation of ubiquitous systems on a distributed peer to peer foundation	- Smart Doorplate application - JXTA -A Service Proxy	- Computing power -Memory space -Energy supply
[16] Designing a framework named Cilia as an integration tool to meet the requirements for Cyber-physical systems	-HMI is a Schneider Electric Magelis G5U Open Box -Google Cloud Messaging	- Mediator Creation -Mediator Removal -Binding removal -Mediator removal -Binding Creation Monitoring cost
[17] - Making adaptation decisions in different layers involving middleware, resource and application	- A farm alarm system - AI planners - Workflow description languages	changing application requirements and contextual changes
Autonomic Resource Management and Dynamic Resource Provisioning		
[20] Scaling cloud services to enhance the overall performance by using the rules and regulations	- Linux LXC virtual containers (LXC)(Ubuntu OS) - OpenStack -Word press	- The types of mandatory and optional components/services -The types of services it requires for scaling the deployment
[21] Deal with the size, heterogeneity and dynamism	- Developing simple heuristics to consolidate VMs -CloudStack	CPU, Memory, Cluster
[24]- Reducing the SLA violation rate, improving user satisfaction and effective scheduling of resources - Focuses on uncertainty and resource dispersion	-Cloud Workload Management Portal (CWMP) - k-means clustering	workload details, pricing plans and estimating the execution charges

5. Autonomic Cloud-Based Industrial Applications/Frameworks

5.1. Qubell

Qubell [29] is one of the companies that has launched an autonomic application management platform for cloud applications which allows accelerated development, testing, delivery and operations of web, ecommerce and big data applications along with the family of SaaS products. The platform is considered for a wide variety of web, commerce and big data applications with a stress on retail, financial services and high-tech industries. The autonomic application management platform of Qubell contains: 1) Self-Service Portal, which allows developers to set the applications as self-managed and self-adaptive. Hence, it allows operations teams to describe policies to administrate how applications integrate and respond to changes in the environment. The Self-Service portal permits users to provide high-level instructions, for instance launch a new environment or update to the latest version. 2) Control Fabric constantly monitors the constancy of the applications, forms their response to the changes, and executes cascading change requests causing from instructions, events, triggers, failures or other changes in the environment.

Qubell's autonomic application management platform supports the following four use cases:

1. **Self-service Test Environments:** It gives the permission to developers in order to start and arrange of whole application environment by only one click. The application understands how to create the right sandbox on request.
2. **Continuous Delivery:** It works with CI pipeline tools such has Jenkins. Qubell makes sure that a correct construction has been verified in the correct environment.
3. **Visibility of Configuration Changes:** within a continuous stream of launches, upgrades, scaling or failure recovery events, Qubell preserves a block of modifications for verification and adoption, and allows administrators to recognize the situation of any configuration at any time with powerful tools for changing visualization, tracking and audit.

5.2. CometCloud

CometCloud [30] is an AC engine for cloud and Grid environments. It is based on the Comet decentralized coordination substrate, and supports highly heterogeneous and dynamic cloud or Grid infrastructures, integration of either public or private clouds and autonomic cloudbursts. CometCloud is composed of a programming layer, service layer, and infrastructure layer. The infrastructure layer uses the Chord self-organizing overlay, and the Squid information discovery and content-based routing substrate built on top of Chord. The routing engine supports flexible content-based routing and complex querying using partial keywords, wildcards, or ranges. It also guarantees that all peer nodes with data components that match a query or message will be situated. The service layer offers a sort of services to supports autonomic at the programming and application stage. Asynchronous messaging and event services are also offered through this layer. To end with, online clustering services support autonomic management and allow self-monitoring and control. Events describing the behavior of the system elements are clustered in which the clustering is applied to identify abnormal behaviors. The programming layer offers a basic framework for application development and management. It supports a range of paradigms including the master and worker. Other supported paradigms include workflow-based applications as well as Mapreduce and Hadoop.

5.3. Cloud TM

Cloud-TM [31] is an extremely advanced data-centric middleware platform to simplify development the costs of operational and administration tasks in cloud applications. The Cloud-TM programming model offers a clear support for object orientation, data structures and frameworks to control distributed execution of tasks, conceal the problems like fault-tolerance, load distribution and data placement. At the end, Cloud-TM's follows relaxing the main costs in cloud- based applications, known as operational costs in two ways: resource provisioning in an autonomic manner according to the cloud structure based on the user requirements and budget constraints. This permits assuring that applications only use the minimum amount of necessary resources to sustain the incoming load, reducing the operational and administration costs.

6. Platforms and Algorithms for Developing Autonomic Cloud Applications

6.1. Amazon Elastic Compute Cloud

This technology [32] provides the virtual systems to the users to run their applications without requiring hardware. EC2 lets scalable extension of applications by offering a web service in which a user may create a virtual machine which is known as an instance. Also, a user may create, launch and terminate service instances when required. Amazon also offers a range of instances with different configurations of CPU, memory, storage, and networking resources to fulfill the user requirements.

6.2. Microsoft Azure

Microsoft Azure [33] is a platform offered by Microsoft to have an efficient management over the cloud resources. Corresponding to the AC, Azure is able to automate managing, creating and deleting resources through the powershell or Azure command-line interface. Azure PowerShell is a collection of modules that offers cmdlets to manage the Azure. It is possible to use the cmdlets to create, manage, and remove Azure services. The cmdlets allows user to obtain stable, repeatable, and hands-off extensions. By using the Azure command-line interface, a user is able to create, manage, and remove Azure resources from the command line. The Azure CLI is available for Linux, Mac OS X, and Windows.

6.3. CloudSim

CloudSim [34] is a Java-based simulation tool provided by the University of Melbourne for Cloud Environments. It offers a system and behavioural modelling of the Cloud computing elements and it is a library to do the simulation in cloud environments. Different classes are provided for data centres, computational resources, virtual machines, applications, users, and policies for the system management like resource scheduling and provisioning. It has a layered architecture that involves the simulation engine, cloud services and source code. The data center in CloudSim architecture has one or more Host and every host has one or more VMs. The IaaS layer in cloud can be simulated using the CloudSim.

6.4. Apache OpenStack

OpenStack [35] is a platform for IT infrastructure as a service which permits to collegiate computing resources that can be used to create public, private and hybrid services to deliver IT infrastructure such as compute nodes, network and storage as a service to end users on demand.

6.5. Open Nebula

OpenNebula [36] offers very simple and flexible solution for the broad management of virtualized data centers to allow private, public and hybrid IaaS clouds. It also includes all the properties required to provide public cloud services. Additionally, OpenNebula offers properties at two major layers of Data Center Virtualization and Cloud Infrastructure:

1. **Data Center Virtualization Management:** Most of the users users use Opennebula in order to manage data center virtualization, consolidate servers, and unify current IT resources for computing, storage, and networking.
2. **Cloud Management:** Users also use OpenNebula to offer a multi-tenant, cloud- like provisioning layer on top of the current infrastructure management solution . These users are looking for provisioning, elasticity and multi-tenancy cloud properties such as virtual data centers provisioning, datacenter federation or hybrid cloud computing to connect in-house infrastructures with public clouds, while the infrastructure is managed by already familiar tools for infrastructure management and operation.

6.6. Bayesian Networks Probabilistic Model (BNPM)

Bayesian networks [37] refer to the combination of statistics and artificial intelligence. A probabilistic model can be created according to the results from the input data and it is being operated based on the random data. The applications of BNPM in cloud environments are varied such as auto-scaling the cloud resources.

6.7. Multi-objective Optimization

When there are a set of constraints, the aim of using the multi-objective optimization [38] is to minimize and maximize multiple objective functions. This is important to take into account finding an optimal

solution. There are different applications of multi-objective optimization such as: Economics, Finance, optimal control, optimal design, Process optimization, resource management and etc. MOO, is broadly used in the area of ACC especially in terms of creating large numbers of large-scale data centers, resource and power management.

6.8. Ant Colony Optimization

This optimization method is one of the swarm intelligence algorithms in order to optimize resource utilization and scheduling. Ant colony optimization algorithms have been used to produce near-optimal solutions to the traveling salesman problem. They have an advantage over simulated annealing and genetic algorithm approaches when the graph may change dynamically. The ant colony algorithm can be run continuously and can be adapted to changes in real-time.

7. Conclusion

Cloud computing is a new paradigm which brings many opportunities for managing the complex operations dynamically in the industry and organizations without relying on hardware and by leveraging the virtualized services. However, the complexity of the existing cloud environments in terms of management and resource provisioning, requires applying autonomic capabilities by using Software engineering techniques to overcome. the problems such as QoS, SLA and QoE. AC leads to new concept of integrating with cloud to solve the existing issues in cloud infrastructure. It applies MAPE loops to enhance the dynamicity, workflow management, dynamic resource provisioning and scheduling. The AC is leveraged by many aspects of IT systems and applications such as Multi-agent systems, Multi-tier applications, designing middleware solutions.

In this paper, we investigated the ACC and the variety of areas which it has been used. We tried to evaluate the recent conducted researches in a taxonomic way. We categorized those works in their specific areas. Meanwhile, we presented important applications, platforms and algorithms used in designing autonomic cloud systems.

References

[1] Erl, T., Puttini, R., & Mahmood, Z. (2013). *Cloud Computing: Concepts, Technology Architecture*. Upper Saddle River, N.J.: Prentice Hall.

[2] Parashar, M., & Rana, O. (2014). Autonomic clouds. *Proceedings of IEEE/ACM 7th International Conference on Utility and Cloud Computing* (pp. 539-540).

[3] Kephart, J. O., & Chess, D. M. (2003, Jan.). The vision of autonomic computing. *Computer, 36(1)*. IEEE.

[4] MacArthur, L. P., & Leaney, A. J. (2005). Defining autonomic computing: A software engineering perspective. *Proceedings of the 2005 Australian Software Engineering Conference* (pp. 88-97). IEEE.

[5] Buyya, R., Calheiros, R. N., & Li, X. (2012). ACC: Open challenges and architectural elements. *Proceedings of the Third International Conference on Emerging Applications of Information Technology (EAIT)*. IEEE.

[6] Calinescu, R. (2009). Resource definition policies for autonomic computing. *Proceedings of the 2009 Fifth International Conferences on Autonomic and Autonomous Systems.*

[7] Oliveira, F. A. D., Ledoux, T., & Sharrock, R. (2013). A Framework for the coordination of multiple autonomic managers in cloud environments. *Proceedings of the 2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*.

[8] Sano, M., Stefano, A., Morana, G., & Zito, D. Controlling distributed systems using parallel autonomic managers. *Proceedings of the 2013 Seventh International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*.

[9] Wooldridge, M. (2009). *An Introduction to Multiagent Systems*. New York: J. Wiley.

[10] Marzo, S., & Gleizes, G. M. (2006, Jan.). A self-organisation and emergence in MAS: An overview. *Informatica, 30(1)*, 45-54.

[11] Sun, P., Dai, Y., & Qiu, X. (2017, June). Optimal scheduling and management on correlating reliability, performance, and energy consumption for multiagent cloud systems. *IEEE Transactions on Reliability, 66(2)*, 547-558.

[12] Othmane, E., & Hebri, R. S. A. (2012). Cloud computing multi-agent systems: A new promising approach for distributed data mining. *Proceedings of the 2012 34th International Conference on Information Technology Interfaces (ITI)*. IEEE.

[13] Al-Ayyoub, M., Jararweh, Y., Daraghmeh, M., & Althebyan, Q. (2015). Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure. *Cluster Comput., 18*, 919.

[14] Kumar, A., Tayal, A., Kumar, R. K. S., & Bindhumadhava, B. S. (2008). Multi-agent autonomic architecture based agent-web services. *Proceedings of the 2008 16th International Conference on Advanced Computing and Communications (ADCOM)*. IEEE.

[15] Moudjari, R., & Sahnoun, Z. (2017, May). A multi agent system for cloud of clouds elasticity management. *Proceedings of the 2017 8th International Conference on Information Technology (ICIT)* (pp. 605-613). IEEE.

[16] Trumler, W., Petzold, J., Bagci, F., & Ungerer, T. (2004). AMUN - autonomic middleware for ubiquitous environments applied to the smart doorplate project. *Proceedings of the International Conference on AC* (pp. 274-275). IEEE.

[17] Lalanda, P., Morand, D., & Chollet, S. (2017, Jan.-Feb.). Autonomic mediation middleware for smart manufacturing. *Internet Computing, 21(1)*. IEEE.

[18] Bellur, U., Narendra, N. C., & Mohalik, S. K. (2017, Jun.). AUSOM: Autonomic service-oriented middleware for IoT-based systems. *Proceedings of the 2017 IEEE World Congress on Services* (pp. 102-105).

[19] Karakostas, B. (2014). Towards autonomic cloud configuration and deployment environments. *Proceedings of the 2014 International Conference on Cloud and AC (ICCAC)*. IEEE.

[20] Weingartner, R., Brascher, G. B., & Westphall, C. B. (2016). A distributed autonomic management framework for cloud computing orchestration. *Proceedings of the 2016 IEEE World Congress on Services*.

[21] Fargo, F., Tunc, C., Al-Nashif, Y., Akoglu, A., & Hariri. S. (2014). Autonomic workload and resources management of cloud computing services. *Proceedings of the 2014 International Conference on Cloud and AC*.

[22] Salih, N. K., & Zang, T. (2012). Autonomic and cloud computing: Management services for healthcare. *Proceedings of the 2012 IEEE Symposium on Industrial Electronics and Applications.*

[23] Singh, S., Chana, I., & Buyya, R. (2017). STAR: SLA-aware autonomic management of cloud resources. *IEEE Transactions on Cloud Computing, PP(99)*, 1.

[24] Hadded, L., Charrada, F. B., & Tata, S. (2015). An efficient optimization algorithm of autonomic managers in service-based applications. In Debruyne C. *et al.* (Eds.), *On the Move to Meaningful Internet Systems: OTM 2015 Conferences* (pp. 19-37). Springer, Cham.

[25] Arani, G., Jabbehdari, M., & Pourmina, S. (2016, September). Autonomic mediation middleware for smart manufacturing. *Cluster Computing, 19(3)*, 1017-1036.

[26] Pautasso, C., Heinis, T., & Alonso, G. (2007, January). Autonomic resource provisioning for software business processes. *Journal Information and Software Technology Archive, 49(1)*, 65-80.

[27] Gergin, I., Simmons, B., & Litoiu, M. (2014). A decentralized autonomic architecture for performance

control in the cloud. *Proceedings of the 2014 IEEE International Conference on Cloud Engineering (IC2E)*.
[28] Goswami, V., Patra, S. S., & Mund, G. B. (2013, September). Dynamic provisioning and resource management for multi-tier cloud based applications. *Foundations of Computing and Decision Sciences, 38(3)*, 175–191.
[29] Qubell. Retrieved from the website: http://qubell.com
[30] Diaz-Montes, J., AbdelBaky, M., Zou, M., & Parashar, M. (2015, Jan.-Feb.). CometCloud: Enabling software-defined federations for end-to-end application workflows. *IEEE Internet Computing, 19(1)*.
[31] Cloud TM. Retrieved from the website: http://www.cloudtm.eu
[32] Retrieved from the website: http://aws.amazon.com
[33] Dordevic, B. S., & Jovanovic, S. P., & Timcenko, V. V. (2014). Cloud computing in Amazon and Microsoft azure platforms: Performance and service comparison. *Proceedings of the Telecommunications Forum Telfor (TELFOR)* (pp. 931-934).
[34] Long, W., Yuqing, L., &. Qingxin, Y. (2013). Using cloudsim to model and simulate cloud computing environment. *Proceedings of the 2013 Ninth International Conference on Computational Intelligence and Security.* IEEE.
[35] OpenStack. Retrieved from the website: https://www.openstack.org
[36] OpenneBula. Retrieved from the website: https://opennebula.org
[37] Robert, G. C., Dawid, P., Lauritzen, S. L., & Spiegelhalter, D. J. (2006, May). Probabilistic networks and expert systems. *Springer Science Business Media*, 324.
[38] Rangaiah, G. P. (2009). Multi-objective optimization. *World Scientific - Technology Engineering*, 454.

Hamed Hamzeh received his Ms.c. from Istanbul Sehir University, Turkey in data science program where he was working on big data and computer networks. He was also working on the project named (cloud-supported adaptive streaming of modern video) supported by the Scientific and Technological Research Council of Turkey (TUBITAK) and under the supervision of Prof. Shervin Shirmohammadi. Currently, he is a Ph.D. student at Bournemouth University in computing and informatics under supervision of Dr. Sofia Meacham.

Sofia Meacham received her diploma in computer and informatics engineering, in 1994, and her PhD degree, in 2000, from University of Patras, Greece. Her research interests fall in the area of system-level design for embedded systems, and include specification techniques for complex embedded telecommunication systems, hardware-software co-design, formal refinement techniques, and reuse practices. Dr Meacham has been working in EU-funded projects as researcher/embedded software engineer both in Industry and in University since 1995, and have accomplished a large amount of teaching experience in several institutions (UK and Greece) since 2000. Specifically, she worked as an embedded software engineer for INTRACOM telecommunications solutions and ISD S.A. and has participated in more than 10 research projects funded by the European Commission regarding developing telecommunications systems and system level methodologies. Dr Meacham has published a number of referred articles in international journals and well known conferences in the area of co-design with particular interest in system level design, design automation tools and real time applications. During the last couple of years, she has experience and publications (scientific committee/chair in conferences for

technology in education) on the use of technology in education and e-learning. She is also an active committee member of BCS (British Computer Society) and IET (Institute of Engineering and Technology) for the last three years. During that period, she participated in numerous events giving talks regarding the participation of "women in computing".

Botond Virginas recieved his Ph.D. from Portsmouth University, UK. He is currently working in British Telecom (BT) as a data scientist in different projects.

Keith Phalp is the executive dean of the Faculty of Science and Technology at Bournemouth University, and professor of software engineering. Previous leadership roles have included head of software systems and psychology, head of computing and informatics, and deputy dean for education and professional practice. His research foci cover software engineering, particularly the early phases of software projects and the relationship between business and software models, model driven development, applications of AI, and, in recent years, social computing, which encompasses digital addiction and online gambling. He has extensive project leadership expertise, having led major successful EC funded projects, as well as Knowledge Transfer Projects, and his research has brought tangible and financial benefits to the organisations involved.