Managing Application Level Elasticity and Availability

Maria Toeroe
Ericsson, Montreal, Canada
Maria.Toeroe@ericsson.com

Neha Pawar, Ferhat Khendek
Engineering and Computer Science
Concordia University,
Montreal, Canada
{n_pawa, khendek}@encs.concordia.ca

***Abstract*—Elasticity and availability are two features associated with the cloud. Existing solutions focus on providing both at the level of the virtual infrastructure through virtual machines (VMs), their restart, addition, and removal as needed. These assume a specific application design paradigm, which equates the application and its workload to the VM. High-availability applications are typically composed of redundant components that recover from failures through state-full failover orchestrated by a middleware (MW). For such applications handling elasticity purely through addition and removal of VMs is not sufficient, the application level also needs to be considered. This requires solutions that coordinate the availability and elasticity management at the application level. In this paper we propose a solution in the context of the Service Availability Forum (SAF) defined MW. It manages the application level elasticity through the manipulation of the application configuration used by the MW to ensure service availability. This in turn triggers the MW to change the workload distribution in the system.**

Keywords— Cloud; elasticity; availability; application; management

I. INTRODUCTION

Availability is defined for a given interval of time in terms of the percentage of time the application is up and its services are provided [1]. High availability is achieved when the outage is less than 5.25 minutes per year [1]. Availability is an implicit expectation of cloud systems. However when it comes to high-availability, it is a big challenge. Elasticity, on the other hand, is one of the key features of cloud computing that makes it economically attractive and drives its fast spreading. Elasticity is defined as the provisioning/de-provisioning of resources according to the workload variations [2] allowing for the "pay-as-you-go" charging model.

Current cloud solutions typically handle both availability and elasticity at the level of the virtual infrastructure: They manage the availability by starting and restarting virtual machines (VMs); and similarly elasticity is reduced to provisioning and de-provisioning VMs as the workload changes. This approach implies a simple design paradigm, which equates the application and its workload to the VMs running the application. It also assumes that the application starts with the VM, it is stateless and the VMs running the same application participate in a load sharing schema. These assumptions are not necessarily true for telecom applications that provide highly available services. They usually run in a cluster; their availability is managed by a middleware (MW) according to some redundancy model (RM). They are state-full: Some components act as standbys to protect the services provided by their active peers. The MW is in charge of the life-cycle of the application components and requires a configuration describing the organization of the application. In this context removing VMs because of decreased workload would be perceived as VM failures and trigger recovery, while adding new VMs in response to workload increase may not necessarily lead to their utilization at the application level, potentially causing repeated triggers for the elasticity management at the infrastructure level. Repeated triggers may also happen due to the availability management performing switch or failover at the application level, meaning that the workload cannot be associated with a given VM.

In this paper we take a look at these issues and propose a solution that handles the elasticity of applications relying on the Availability Management Framework (AMF) [4] of the Service Availability Forum (SAF) [3]. AMF is responsible for maintaining the availability of the application services through the coordination of the redundant application components. The redundancy and the logical organization of the applications are described for AMF in a configuration, which is part of the information model maintained by the SAF Information Model Management (IMM) [6] service.

The SAF MW is a proven solution for service availability in the context of cluster computing; however it was not explicitly designed to handle elasticity as expected in cloud computing. The purpose of this paper is to demonstrate that it nevertheless has the flexibility necessary to handle elasticity. We can use the configuration designed for service availability to achieve elasticity therefore achieving both simultaneously. We propose an elasticity engine (EE), which is responsible of reacting to workload variations by modifying the AMF configuration so that AMF rearranges the service assignments in the cluster, thus providing more or less resources to the entities handling the given service. This solution can also be integrated with the elasticity management of the cloud at the infrastructure level. The EE can close the gap between today's cloud solutions for elasticity management and the SAF MW.

II. BACKGROUND INFORMATION

SAF has defined standard interfaces that facilitate the development and deployment of carrier-grade and mission critical applications and systems [3]. These are defined as MW services among which AMF [4] and IMM [6] are the most relevant to this paper.

AMF manages the availability of application services by coordinating the redundant application components. To do so AMF requires a configuration, which describes the organization of the logical entities composing the application. For the purpose of availability management AMF distinguishes the services provided by an application from the entities providing them. The main service provider entities are the service units (SU) capable of providing some required application services. To protect the application services the SU is replicated on different nodes (e.g. VMs) to create redundancy and form a service group (SG). These SUs collaborate according to a RM. Five RMs have been defined in [4], namely 2N, N+M, N-way, N-way-Active and No-Redundancy. An application may consist of several SGs with different RMs. In the AMF configuration the services are represented as service instances (SI), which AMF assigns to the SUs at runtime. Depending on the RM configured for the SG, AMF assigns each SI to one or more SUs in the active and/or in the standby roles. Subsequently, should an SU fail AMF moves its service assignments to another healthy SU within the SG according to its RM. Several applications may be deployed in the same cluster, which may be composed of heterogeneous nodes/VMs.

In the N-way-Active RM an SI is assigned to multiple SUs in the active role resulting in a load sharing. Today's cloud deployments (e.g. web-application) follow this RM. A special case of the N-way-Active RM is the No-Redundancy RM where each SI has one assignment and there is at most one assignment per SU. Without a standby assignment these RMs cannot preserve the application state. To do so in the N-way RM an SI has one active assignment, which is backed up with some standbys. An SU can play simultaneously the active and the standby roles for different SIs, which also means that it can have multiple assignments. The N+M RM is a special case of N-way RM where each SI may have at most one standby assignment and an SU may play only one role for all its SIs: active or standby. In the SG there are N active and M standby SUs. For the 2N RM, N and M are equal to 1. Thus, the RM determines the number and the way the SI assignments are distributed among the SUs [4]. The SIs are discrete units in the AMF configuration. Serving an SI assignment generates an actual workload, which may fluctuate at runtime. Handling this fluctuation adequately is our goal through the customization of the RMs. The AMF provides attributes to configure the RMs and also to observe the resulting changes in the system.

III. MANAGING ELATICITY THROUGH AMF CONFIGURATION CHANGES

Whenever AMF receives a configuration change, it evaluates the execution state of the system whether it matches the requested configuration. If not, AMF tries to rearrange the SI-to-SU assignments to achieve a better match while also maintaining the availability of the services. Thus, we can force AMF to change the SI-to-SU assignments by modifying some configuration attributes and as a result re-distribute the workload within a given SG and in the cluster in general. This in turn provides more (or less) resources to the different SIs representing different workload in the system, therefore we can fulfill the goal of elasticity management by selecting an appropriate set of configuration modifications that change the resources provided to a given SI which represents a given fluctuating workload.

The prerequisite for managing the elasticity using the AMF configuration is to identify the attributes that play a role in the elasticity management, the constraints applicable to them, as well as their impact on the load distribution. We have evaluated the AMF configuration attributes with respect to their role for managing the elasticity for the different RMs. As expected, most of the relevant attributes are defined in the class representing the SGs (e.g. the attributes specifying N and M for the N+M RM). Some of the relevant configuration attributes are also defined in the class representing the SIs, e.g. the attribute that specifies the number of active assignments for an SI protected by the N-way-Active RM and therefore the number of SUs sharing the workload represented by the SI. We defined a strategy for each of the RMs to handle the fluctuating workload of their protected SIs through configuration changes.

Many of these attributes are interrelated and constrain each other, we had to characterize these constraints. For instance, in the N+M RM the values of N and M, are subject to the following constraints: 1) The number of active SUs and the number of standby SUs is less or equal to the number of in-service SUs in the SG; 2) there is at least one active SU; 3) the number of active SUs is sufficient to provide all SIs protected by the SG; and finally 4) the number of standby SUs is sufficient to protect all SIs protected by the SG. Note that while constraints 1) and 2) check the validity of the configuration; constraints 3) and 4) ensure that the configuration is valid with respect to the availability requirements. More details about the elasticity related attributes and the constraints on their modifications can be found in [7].

IV. OVERALL ARCHITECTURE FOR ELASTICITY MANAGEMENT

We assume that as long as the AMF configuration remains valid with respect to the availability constraints, AMF will maintain the availability as required. Therefore, our focus is to complement AMF with a solution for the elasticity management. The overall architecture of our solution is shown in Fig. 1. Internally, the EE is composed of the Elasticity Controller, the RM Adjustors (one for each RM) and the Buffer Manager. To detect workload fluctuations workload monitors monitor the different SIs. Each workload monitor informs the Elasticity Controller of the EE about any significant change in the workload of a given SI. Whenever the EE receives such a signal, depending on the RM of the SG the Elasticity Controller calls the appropriate RM Adjustor. The RM Adjustor calculates the necessary configuration changes to adjust the configuration. It applies the configuration changes according to the various strategies described in Section V. In turn, AMF implements these changes by rearranging the SI-to-SU assignments as necessary. To speed up future adjustments

the Buffer Manager reserves additional nodes or free up allocated ones through additional configuration changes.

Depending on the outcome of the adjustments the Elasticity Controller may initiate additional adjustments to collocated SGs and therefore sharing resources with the given SG. It may also request the addition or removal of nodes if the cluster size is insufficient or some nodes were freed up, and/or request the installation of required software on additional nodes within the cluster.

V. ELASTICITY STRATEGIES

A. Workload increase

When a monitor reports an increase in the workload of an SI, the RM Adjustor will try to increase the capacity that this SI can use by:

1) Spreading the SI workload

This strategy is used when the impacted SG has the N-way-Active RM. The EE will try to increase the number of assignments of the SI, therefore spreading its workload across more SUs. At the cluster level the EE may apply the same strategy to SIs of other SGs with the N-way-Active RM, which share node(s) with the SG protecting the SI experiencing the increase. By spreading the load of these SIs on more nodes they leave more capacity available to the SI with the increased workload, which may be protected by any of the RMs.

2) Distributing the SIs over more SUs

This strategy cannot be applied to the 2N and the No-Redundancy RMs. At the SG level the RM Adjustor may handle the increase of the workload of an SI by distributing the SIs to more SUs within the SG, and therefore giving more capacity to each SI including the one experiencing the increase. The strategy can be applied also at the cluster level to other SGs sharing the node(s) with the SI experiencing the increase.

3) Prioritizing the SU on the least loaded node

The RM Adjustor uses this strategy primarily for the 2N or No-Redundancy RMs. AMF choses an SU for an assignment based on the SU's rank, therefore swapping the rank of the SU currently active for the SI with the rank of the SU on the least loaded node will cause AMF to move the SIs of the currently active SU to that node having more capacity. The same strategy can be applied again at the cluster level.

B. Workload decrease

If the workload decreases, the RM Adjustor will try to free up some capacity at the SG level first, then if needed at the cluster level. When the RM Adjustor handles the workload decrease it always tries to free an SU. The EE handles workload decrease also according to the RM of the involved SGs. The three strategies used to handle decrease 1) Merging the SI workload; 2) Re-grouping SIs on less SUs of the SG; and 3) Prioritizing the nodes that serve other SIs correspond to those handling the increase.

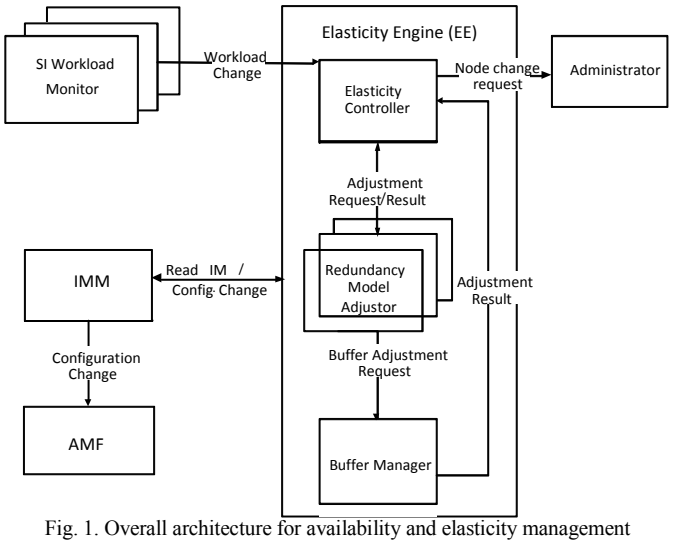

Fig. 1. Overall architecture for availability and elasticity management

C. Buffer Management

Having an SU configured on a node for an SG does not mean that AMF will use that SU for assignments. Indeed AMF distinguishes the subset of assigned SUs in the set of in-service SUs, which is in turn a subset of the SUs configured in the SG. In-service SUs are those instantiated and ready to serve assignments but not necessarily assigned yet, while SUs configured in the SG but not in service are un-instantiated spares that fulfill the prerequisites to be instantiated. Thus, we can use these different subsets to prepare AMF for sudden workload increases by reserving additional SUs at different levels of readiness as buffers. Accordingly, we maintain two buffers: The *in-service-SU-buffer* and the *un-instantiated-SU-buffer*.

Whenever the EE increases the number of assigned SUs of the SG it also prepares for possible sudden workload increase by bringing into service some additional SUs to maintain the configured size of the in-service-SU-buffer. The EE can increase the number of in-service SUs only if there are additional SUs configured in the SG. Therefore, it also ensures that there are some un-instantiated spare SUs configured in the SG to maintain the un-instantiated-SU-buffer. Otherwise, it tries to configure new SUs on additional nodes. If this is not possible the EE signals the need for new nodes. This is the point where the EE can be integrated with the cloud orchestration. In case of workload decrease the EE tries to free up resources by decreasing the number of assigned SUs. If this is successful the EE also decreases the number of in-service SUs while maintaining the in-service-SU-buffer. The EE may also decrease the number of configured SUs provided the un-instantiated-SU-buffer is maintained.

D. Handling changes in the number of SIs

A change in the workload may also manifest as an increase or decrease of the number of SIs. Such a change means a configuration change therefore the EE will receive the notification from IMM. It handles the increase in the number of SIs by first checking if the assignments of the new SI can be accommodated in the SG. If yes, no action may be needed. If it is not the case, then the EE may try to increase the SG capacity

by increasing the number of SIs that each SU of the SG can handle. This strategy is applicable to all RMs except for the No-Redundancy RM. Since this change may decrease the capacity provided to the existing SIs the EE may combine it with an appropriate strategy seen in Section A. Alternatively, the EE may increase the number of SUs handling the SI load within the SG. This is similar to the strategy presented in A.2. In the case of SI removal the EE tries to adjust to the decrease by re-grouping the SIs on less SUs of the SG and therefore freeing up an SU.

VI. Related work

Since the introduction of the cloud computing paradigm a lot of work has been done on elasticity. Some of it is more about scalability as distinguished in [2]. Other papers propose techniques for elastic provisioning and de-provisioning of resources in the cloud. For instance, [8] considers an architecture with incoming jobs that stores these jobs in a queue and according to some scheduling policies and job characteristics decides to add more resources when the queue length increases or remove resources when fewer jobs are in the queue. In [9] the authors introduce a system for dynamically increasing the virtual organization of clusters. In this work they monitor a job queue and spawn VMs when needed. [10] uses a profile-based approach to develop just-in-time scalability (i.e. elasticity) by increasing or decreasing the number of components composing an application for improved resource utilization and better performance. Other researchers have proposed workload prediction for more efficient elasticity handling [11]. Very little work has been done to handle availability and elasticity at the same time in the same architecture. OpenStack [12] is a popular cloud controller. It consists of several related projects, among which Heat [13] and Ceilometer [14] are of interest to our work. Heat aims at providing the auto-scaling (i.e. elasticity) and availability. It can automatically increase or decrease the number of VMs in response to workload changes based on policies defined in the configuration. Ceilometer [14] provides a metering service to inform Heat. Heat also reacts to application failures by restarting the appropriate resources. However, failure detection may take up to a minute, which is significant when considering HA. Heat is comparable to our solution from the elasticity perspective provided one can equate the services of a VM to the application service; and if we consider only the N-way-Active RM. In this respect it is still up for debate whether the cloud design paradigm is sufficient for carrier-grade and mission critical applications to meet their stringent requirements.

VII. Conclusions

The solution presented in this paper has been prototyped and tested using the OpenSAF [5] implementation of the SAF specifications. However, as the workload monitor is still under development and requires further research the workload increase and decrease is provided through a user interface. Nevertheless with the prototype implementation, we have demonstrated that the application level elasticity can be managed within the framework of an existing MW designed to manage service availability. Thus, positioning AMF as a MW solution for managing application availability in the context cloud as well. In this respect the integration of the EE with the cloud orchestration requires further considerations as moving clustered applications into the cloud context can result in contradicting actions if availability is handled at the different levels simultaneously, therefore requiring proper coordination. The EE strategies has not been tested for multiple simultaneous SI-workload changes. Although these changes may be serialized and applied in a sequence, this may not be optimal as the relations between these changes would be ignored.

Acknowledgement

This work has been partially supported by Natural Sciences and Engineering Research Council of Canada (NSERC) and Ericsson.

References

[1] M. Toeroe and T. Francis, “Service Availability: Principles and Practice” Wiley and Sons, Chichester, 2012.

[2] N. R. Herbst, S. Kounev, and R. Reussner, “Elasticity in Cloud Computing : What It Is and What It Is Not”, 10th International Conference on Autonomic Computing (ICAC ’13), pp. 23–28, 2013.

[3] Service Availability™ Forum available at http://www.saforum.org [Accessed on 10 Nov 2013].

[4] SAF, Availability Management Framework SAI-AIS-AMF-B.04.01 available at http://www.myassociationvoice.com/HOA/assn16627/images/SAI-AIS-AMF-B.04.01.pdf.

[5] OpenSAF available at http://www.opensaf.org.

[6] SAF, Application Interface Specification Information Model Management Service SAI-AIS-IMM-A.03.01 http://www.myassociationvoice.com/HOA/assn16627/images/SAI-AIS-IMM-A.03.01.pdf.

[7] N. Pawar, “Managing high-availability and elasticity at the application level”, M.A.Sc. Thesis, Engineering and Computer Science, Concordia University, August 2014.

[8] P. Marshall, K. Keahey, and T. Freeman, “Elastic Site: Using Clouds to Elastically Extend Site Resources,” 2010 10th IEEE/ACM Int. Conf. Clust. Cloud Grid Comput., pp. 43–52, 2010.

[9] M. A. Murphy, B. Kagey, M. Fenn, and S. Goasguen, “Dynamic Provisioning of Virtual Organization Clusters,” 2009 9th IEEE/ACM Int. Symp. Clust. Comput. Grid, pp. 364–371, 2009.

[10] J. Yang, J. Qiu, and Y. Li, “A Profile-Based Approach to Just-in-Time Scalability for Cloud Applications,” IEEE Int. Conference on Cloud Computing., pp. 9–16, 2009.

[11] N. Roy, A. Dubey, A. Gokhale, “Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting”, IEEE Int. Conference on Cloud Computing., pp.500,507, 2011.

[12] OpenStack available at https://www.openstack.org/.

[13] Heat avaialble at https://wiki.openstack.org/wiki/Heat D. Michelino, J. C. Leon, and L. F. Alvarez, “Implementation and testing of OpenStack Heat,” no. September, 2013.

[14] Ceilometer avaialble at https://wiki.openstack.org/wiki/Ceilometer.