Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 32 (2014) 850 – 855
Procedia Computer Science

1st International Workshop "From Dependable to Resilient, from Resilient to Antifragile Ambients and Systems" (ANTIFRAGILE 2014)

Toward Antifragile Cloud Computing Infrastructures

Amal Abid[a,b,*], Mouna Torjmen Khemakhem[a], Soumaya Marzouk[a], Maher Ben Jemaa[a], Thierry Monteil[b,c], Khalil Drira[b,c]

[a] *University of Sfax, ReDCAD, B.P.W, 3038 Sfax, Tunisia*
[b] *CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France*
[c] *Univ de Toulouse, LAAS, F-31400 Toulouse, France*

Abstract

Cloud computing systems are rapidly growing in scale and complexity. They are also changing dynamically as a result of dynamic addition and removal of system components, different execution environments, common updates and upgrades, runtime repairs, mobility of devices and more. Such large-scale, complex and dynamic cloud environments are prone to failures and performance anomalies. Thus, dependability and resilience in cloud computing are of paramount importance to guarantee availability and reliability of services and application execution, even in the presence of large number of faulty components. Antifragility is the key to such techniques. It proposes that some systems could be strengthened by changes and faults instead of be weakened by them. In contrast to classical resilience methods, antifragile techniques aim to build systems that handle unpredictable and irregular events, while growing and getting stronger. Most of the classical resilience techniques are not sufficient to build highly available cloud infrastructures. In fact, they just resist shocks and stay the same. They should be complemented by some other aspects like learning from failure to built more elastic and stronger cloud infrastructures. This may represent the idea of building antfragile cloud systems. In this paper, we discuss the existing resilience techniques and propose a solution to design antifragile systems in cloud computing environments.

Keywords: Antifragilty; Cloud Computing; Dependablity; Resilience

1. Introduction

Cloud computing is earning an increasing popularity over traditional information processing systems for storing and processing huge volumes of data. This concept consists in offering services and resources on-demand over the Internet in the pay-as-you-go model. The Cloud infrastructure is built on modern data centers covering thousands of interconnected servers with capability of hosting a large number of applications. These data centers are often virtualized and computing resources are provisioned to the user in the form of configurable Virtual Machines (VMs).
This aspect promises various benefits such as scalability, mobility and lower usage costs. Unfortunately, failure is

* Corresponding author. Tel.: +216-74-274-088 ; fax: +216-74-666-578.
E-mail address: amal.abid05@gmail.com

doi:10.1016/j.procs.2014.05.501

unavoidable due to internal or external conditions like human-made faults, unreliable hardware/software, natural disasters, etc. In the context of cloud, the inherent complexity and large-scale nature of cloud infrastructures increase the likelihood of failure occurrence. In fact, todays cloud infrastructures are still vulnerable to various system anomalies[1]. Moreover, keeping the trace of the execution status of thousands of running virtual machines (VM) is a hard manual task for system administrators. Therefore, there is a significant requirement to design a reliable and automatic anomaly management system for large-scale cloud infrastructures.

Existing anomaly management approaches for cloud environments can be commonly classified into two categories: reactive and proactive fault tolerance approaches and failure induction approaches.

Reactive methods for failure management and fault tolerance[2],[3],[4] take corrective actions after an anomaly occurs. They rely on checkpointing/restart mechanisms, which periodically save a snapshot of the system and use it for recovery in case of failures. Although the application does not need to be restarted from the beginning, work loss is inevitable due to its rollback process. It is also difficult to reproduce the anomaly-inducing environments for offline anomaly diagnosis. Moreover, in a large-scale system, checkpointing mechanisms could incur significant overhead[1]. On the other hand, proactive methods take preventive actions on all system components in advance and avoid failures by predicting them. They rely on runtime monitoring and various techniques that exploit the current state of a system as well as the past experience. The proactive approach[5],[6] offers better responsiveness, but can lead to continuous overhead for prediction, prolonged service downtime, and especially possible penalties for imprecise failure detection (eg. false alarms).

The second category corresponds to induce failures in the system to improve its resilience and maintain the cloud environment[7]. This approach is based on some strategies such as embracing failure and the design-for-fail. Increasing the frequency of failures not only in the test environment but also in the production one, can lead to a better preparation for the inevitable failures that will occur. With failure induction, cloud providers focus on building systems that are more resilient rather than trying to eliminate or predict failures.

In this paper, we propose a solution for automatic anomaly management system in large-scale cloud infrastructures that relies on failure induction approach and combines it with monitoring and learning mechanisms to further enhance resilience and antifragility. The main contribution of this paper is to propose a solution which gains from the complementarity between failure induction and learning mechanisms as a first step towards antifragile cloud infrastructures.

This paper is organized as follows: Section 2 presents background concepts related to antifragility. Section 3 summarizes related work. Section 4 details our approach. Section 5 concludes and suggests future directions.

2. Background

The concept of antifragility, coined by Nassim Nicholas Taleb in his book *"Antifragile: Things That Gain from Disorder"*[8], specifies systems that are capable of absorbing shocks and be improved positively by them. Like the human body which needs to be exposed to germs and illness in order to get stronger immune system, the antifragile system adapts and evolves in response to stress and changes to its environment. The body can achieve this because it has not been built by a set of rules based on predictable use cases. Systems, on the other side, are created to deal with known risks. Thus, when the system is exposed to shock that was not predicted, there will be serious consequences.

Furthermore, Nassim Taleb explains the behavior of complex systems and distinguishes three kinds: fragile, robust, and antifragile systems. These categories of systems differ in how they respond to shocks, as shown in Figure1:

- **Fragile** system is built to take account of its environment assumptions as stable and immutable. It is characterized by the attitude of taking actions without considering any external event or condition[9].
- **Robust** system resists shocks, tolerates unexpected conditions and stays the same.
- **Antifragile** system grows stronger in response to volatility, randomness and disorder and improves its performance.

The concept of antifragile was developed by Nassim Taleb in the socioeconomic context, not in cloud computing. But we think that this concept may have its largest practical utilization and be very useful if it is applied to cloud systems and environments. Thus, we will focus on this concept in our work.

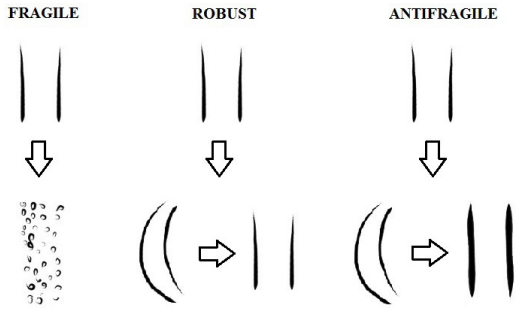

Fig. 1. The fragile versus the robust and the antifragile

3. Related work

As it was already mentioned in the introduction, we distinguish two categories of works: reactive and proactive fault tolerance based works and failure induction based works.

3.1. Works based on reactive and proactive fault tolerance techniques

First research and works[10],[11],[12] have concentrated on architecture-specific solutions to introduce fault tolerance in cloud environments. To use these solutions, users should adapt the properties of fault tolerant environment in their applications. This implies the limitations of creating non transparent and inflexible cloud environments that require significant developers and users efforts. In[4],[13] and[14]authors proposed modular solutions based on the idea that a fault tolerance solution can be viewed as a combination of a set of distinct activities (replication, detection, recovery, etc.) coordinated in a sequential logic to realize a general fault tolerance solution. Authors in[4], implement each module as an independent web service and offer fault tolerance properties to the applications as an on-demand service. In such solution, the challenge of interoperability and portability, and integrability can be covered by the principles of Service-Oriented Architectures, and the composition of activities can be dynamically produced using BPEL-based orchestration engine[15].

On the other side, most of the existing failure prediction systems are based on statistical learning methods[5]. They employ supervised learning techniques to estimate the failure occurrences. Recently, research on proactive anomaly management system[6],[16],[17],[1] has shifted to unsupervised and semi-supervised learning techniques. It has profited from these learning techniques by characterizing system behaviors and suggesting adaptive approaches to predict failure occurrences in cloud computing environments. Other studies[18],[19] use execution migration techniques to improve resource management by avoiding possible failures.

3.2. Works based on failure induction technique

There are few works that study the failure induction technique in cloud computing. More precisely, we found two works: GameDay[20] and Netflix Simian Army[7].

GameDay presents a set of scheduled exercises where failure is manually injected or simulated to test outage response procedures, to create iterative learn, fix, and validate loops and to increase resilience[1].
With GameDay, failure induction is performed in a testing environment identical to the production, so it would be as realistic as possible. Used by Amazon and Google, GameDay is a good way to induce failure regularly, validate hypothesis about system behavior, and improve system response.

Some organizations go even further, employing a solution that is more scalable and automated so-called Netflix Simian Army (a suit of tools referred as ”monkey services”). Netflix - an American provider of on-demand Internet streaming media, says ”We have found that the best defence against major unexpected failures is to fail often. By

[1] Resilience here means the ability of a system to adapt to changes, failures and disturbances.

frequently causing failures, we force our services to be built in a way that is more resilient" [2]. They want failure to be a nonevent-something that runs all the time in the background so that when a real failure happens, it will be simply handled without any impact. To achieve this, Neflix manages failure to occur in the production environment. This is how they got the idea of the first monkey "Chaos Monkey" of Simian Army. Chaos Monkey is a service that runs on Amazon Web Services. It randomly terminates production instances (virtual machines) to ensure that services are resilient to node failure and can survive without any end-user impact. Inspired by the success of the Chaos Monkey, Netflix has designed the Simian Army that induces several types of failures and tests its capacity to survive them: Latency Monkey to test resilience to network and service degradation, Chaos Gorilla to test resilience to a complete AWS Availability Zone (data center), Chaos Kong to test resilience to an entire region of multiple data centers and many other monkey services.

Although many parts of Simian Army remain an aspiration and only some monkeys have already been built, our approach will take into account the Netflix idea about failure induction.

4. Proposition

Our objective is to design a reliable and automatic anomaly management system for large-scale cloud infrastructures that improves the antifragility concept in cloud computing.

The rational reason of choosing antifragile concept is that cloud computing requires a completely new attitude of viewing failure. In fact, just building fault tolerant systems is not enough, because they process just a list of known use cases and try to avoid failures. These systems often lead to long delivery cycles and a high degree of complexity which in fact, rise the likelihood of failure and create fragile systems. Thus, we adopt the failure induction technique for our cloud management system since it increases both resilience and antifragility (section 4.1). Furthermore, building antifragile systems requires an other step which is monitoring and learning mechanisms (section 4.2).

4.1. Failure induction technique

The failure of an instance (virtual machine) is the most frequent kind of failure met in a cloud environment. Regardless of the cause, the instance becomes unavailable. Consequently, inducing instance failure boosts the resilience of the system and improves availability. To realise such induction, we should just kill a random instance and let new and healthy instance take its place. The termination of instances should be common enough to not cause negative user impact and service degradation.

This idea came to us after making analogy with biological systems, which are antifragile by nature[7]. In fact, biologic systems do more than shock resisting; they require perturbations and disorders to gain power and maturity. In addition, the aspect of "degeneracy/functional redundancy" of living systems supposes that physical death is a necessary part of this systems because it gives birth to new life and a renewal of function[21]. So, we can correspond (1) the death of a biological organ in living systems, to the death of a virtual machine in cloud systems; and (2) the creation of a new biological organ in living system, to the creation of new virtual machine in cloud systems.

In contrast to the Neflix work which terminates instance and automatically brings up a new and identically configured instance, our approach allows to generate a new VM configuration from the learning system in order to create a new healthier instance. This idea will be detailed in the next section.

4.2. Monitoring and learning mechanisms

Failure induction strategy presents the initial part to build antifragile cloud systems. This part should be imperatively accompanied by monitoring and learning mechanisms which are fundamental to build resilient and antifragile cloud systems.

On the first hand, resistance cannot be complete without monitoring which corresponds to the ability to observe and record the state of the system in order to react properly. Monitoring is essential because it brings a deep comprehension

[2] The Netflix Tech Blog. http://techblog. netflix.com/2011/07/netflix-simian-army.html

of the system performance and behavior. Therefore, we propose a monitoring system that can provide information gathering from VM. These information include metrics such as CPU, memory, disk I/O, and network trafic.

On the other hand, antifragility is based on the fact that systems become stronger after shocks. In fact, every failure provides a great learning opportunity to strengthen the system. This property can be achieved by learning mechanisms. Therefore, our anomaly management system should integrate machine learning methods to discover weaknesses in bad instances. Machine learning approaches can be classified into supervised and unsupervised approaches. The supervised approach relies on labeled training data. It may produce a more precise model of classes, but in real-world cloud computing systems, most collected data are not labeled and the failure class is a rare one. So, it is challenging to identify failure occurrences in such systems. In addition, supervised approach requires considerable human effort and can only handle known anomalies. To build an antifragile and dependable cloud computing system, the learning approach should handle both known and unknown anomalies. As a result, we propose to adopt the unsupervised learning approach which is more suitable and practical in cloud environment. Although it is less accurate than supervised approach, unsupervised learning technique captures emergent system behaviors and identifies known and unknown anomalies since it does not require any labeled training data.

We aim to realize an unsupervised behavior learning system that performs as follow:
The first stage presents a health monitoring and control where data are not labeled. We suggest characterizing normal states of the system and detecting anomalous behaviors using an unsupervised learning. For that, this system should use a set of continuous VM behavior learning modules to catch the patterns of normal operations of all VMs, and should look for early deviations from normal system behaviors to detect anomalous behaviors. After the anomalies are verified, either confirmed as failures or classified as normal, data should be labeled. To achieve this, we can use some learning methods such as Bayesian methods or Self Organizing Map method (SOM)[22]. In cloud context, SOM fits our need since it is able to capture complex system behaviors while being computationally not expensive. Besides, the unsupervised behavior learning system can dynamically induce a SOM for each VM to catch its behavior.

Finally, we can combine these two steps by proposing a real-time monitoring for unsupervised behavior learning for virtual machines. This proposition ensures building an antifragile cloud infrastructure that learns from failure, deals with randomness and becomes stronger.

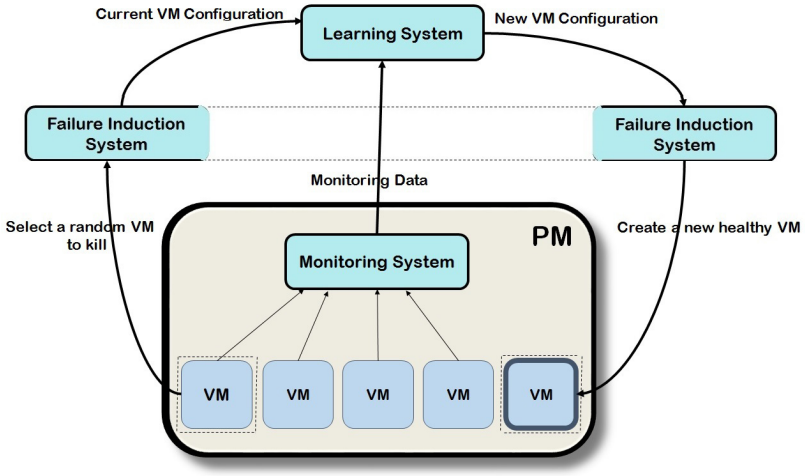

Fig. 2. Antifragile System Architecture

We illustrate our proposition with a simple schema (see Figure2). In the following, we detail the different steps: (1). The monitoring system continuously tracks system-level metrics (e.g. CPU usage, free memory, network traffic, disk I/O statistics) of different VMs. (2). Monitoring data are transferred to the learning system in order to apply unsupervised learning mechanisms over system-level metrics. (3). On the other side, the failure induction system selects a random instance to kill and sends VM configuration to the learning system. (4). The learning system, which contains data collected from monitoring system and old configuration of killed VM, tries to generate a new better VM configuration. In fact, our learning system identifies system metrics that are related to the performance

anomaly and exploits them to produce new VM configuration. (5). Finally, the induction system receives this new VM configuration and creates new healthy instance from it.

The worst case here corresponds to kill a healthy instance and to generate a new VM configuration, which is almost the same to the old one. This is due to the failure induction mechanism that increases the frequency of failures in the production environment.

5. Conclusion

In this paper, we introduced an innovative proposition to create an antifragile cloud computing infrastructure.

Our approach is based on failure induction technique, monitoring and learning mechanisms. With failure induction technique, we increase the frequency of failures in the production environment to a better preparation for the inevitable failures that will occur. This technique enhances the idea of failure embracing instead of failure fighting and avoiding. In fact, every failure is a great opportunity to learn and to become stronger. Thus, we proposed an unsupervised behavior learning system for cloud infrastructures that relies on real-time monitoring.

In future work, we plan to present a detailed architecture of our anomaly management system. We plan also to propose an architecture that performs analysing data not only from virtual machines (VMs) but also from physical machines (PMs), and combines them to improve system quality. This would allow our system to perform at multiple levels (physical and virtual node) and cover a greater variety of potential failures.

References

1. Y. Tan, X. Gu, H. Wang, Adaptive system anomaly prediction for large-scale hosting infrastructures, in: Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC '10, ACM, 2010, pp. 173–182.
2. S. Sidiroglou, O. Laadan, C. Perez, N. Viennot, J. Nieh, A. D. Keromytis, Assure: Automatic software self-healing using rescue points, SIGPLAN Not. 44 (3) (2009) 37–48.
3. G. Chen, H. Jin, D. Zou, B. B. Zhou, W. Qiang, G. Hu, Shelp: Automatic self-healing for multiple application instances in a virtual machine environment, in: Cluster Computing (CLUSTER), 2010 IEEE International Conference on, IEEE, 2010, pp. 97–106.
4. R. Jhawar, V. Piuri, M. Santambrogio, Fault tolerance management in cloud computing: A system-level perspective, IEEE Systems Journal 7 (2) (2013) 288–297.
5. F. Salfner, M. Lenk, M. Malek, A survey of online failure prediction methods, ACM Comput. Surv. 42 (3) (2010) 1–42.
6. D. J. Dean, H. Nguyen, X. Gu, Ubl: Unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems, in: Proceedings of the 9th International Conference on Autonomic Computing, ICAC '12, ACM, 2012, pp. 191–200.
7. A. Tseitlin, The antifragile organization, Commun. ACM 56 (8) (2013) 40–44.
8. N. N. Taleb, Antifragile: things that gain from disorder. "Book", Random House Digital, Inc., 2012.
9. V. D. Florio, On the constituent attributes of software and organizational resilience, Interdisciplinary Science Reviews 38 (2) (2013) 122–148.
10. J. Barr, A. Narin, J. Varia, Building fault-tolerant applications on aws. tech. rep., Amazon Web Services.
11. T. Wood, R. Singh, A. Venkataramani, P. Shenoy, E. Cecchet, Zz and the art of practical bft execution, in: Proceedings of the Sixth Conference on Computer Systems, EuroSys '11, ACM, 2011, pp. 123–138.
12. W. Zhao, P. M. Melliar-Smith, L. E. Moser, Fault tolerance middleware for cloud computing, in: Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD '10, IEEE Computer Society, 2010, pp. 67–74.
13. M. Hiltunen, R. Schlichting, An approach to constructing modular fault-tolerant protocols, in: Reliable Distributed Systems, 1993. Proceedings., 12th Symposium on, IEEE, 1993, pp. 105–114.
14. G. Vallee, K. Charoenpornwattana, C. Engelmann, A. Tikotekar, C. Leangsuksun, T. Naughton, S. L. Scott, A framework for proactive fault tolerance, in: Proceedings of the 2008 Third International Conference on Availability, Reliability and Security, ARES '08, IEEE Computer Society, 2008, pp. 659–664.
15. A. Albreshne, P. Fuhrer, J. Pasquier-Dorthe, Web Services Orchestration and Composition: Case Study of Web services Composition. "Book", 2009.
16. Q. Guan, Z. Zhang, S. Fu, Proactive failure management by integrated unsupervised and semi-supervised learning for dependable cloud systems, in: Proceedings of the 2011 Sixth International Conference on Availability, Reliability and Security, ARES '11, IEEE Computer Society, 2011, pp. 83–90.
17. Y. Watanabe, H. Otsuka, M. Sonoda, S. Kikuchi, Y. Matsumoto, Online failure prediction in cloud datacenters by real-time message pattern learning, in: Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on, IEEE, 2012, pp. 504–511.
18. F. Salfner, P. Troger, S. Tschirpke, Cross-core event monitoring for processor failure prediction, in: High Performance Computing and Simulation, 2009. HPCS'09. International Conference on, IEEE, 2009, pp. 67–73.
19. S. Fu, C.-Z. Xu, Exploring event correlation for failure prediction in coalitions of clusters, in: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, SC '07, ACM, 2007, pp. 1–12.
20. Resilience engineering: Learning to embrace failure, Commun. ACM 55 (11) (2012) 40–47.
21. V. D. Florio, C. Blondia, A survey of linguistic structures for application-level fault tolerance, ACM Comput. Surv. 40 (2) (2008) 1–37.
22. T. Kohonen, Self-organizing maps. "Book", Vol. 30, Springer, 2001.