Proactive Scalability and Management of Resources in Hybrid Clouds via Machine Learning

Dimiter R. Avresky
IRIANC–Munich, Germany
autonomic@irianc.com

Pierangelo Di Sanzo*, Alessandro Pellegrini†, Bruno Ciciani‡, Luca Forte§
DIAG–Sapienza, University of Rome
{*disanzo, †pellegrini, ‡ciciani}@dis.uniroma1.it
§luca.forte@hotmail.it

***Abstract*—In this paper, we present a novel framework for supporting the management and optimization of application subject to software anomalies and deployed on large scale cloud architectures, composed of different geographically distributed cloud regions. The framework uses machine learning models for predicting failures caused by accumulation of anomalies. It introduces a novel workload balancing approach and a proactive system scale up/scale down technique. We developed a prototype of the framework and present some experiments for validating the applicability of the proposed approaches.**

I. Introduction

Nowadays, software anomalies are recognized as some of the major problems affecting performance and availability of computer applications. For example, in [1] it has been shown that in web applications an average of 40% of anomalies is due to software errors. Accumulation of software anomalies (such as memory leaks, unterminated threads, unreleased locks and file fragmentation) may cause performance loss and/or reduce service availability due to system hangs or crashes. These problems become more complex to be addressed when dealing with distributed applications deployed at a geographical scale. Indeed, these kinds of applications typically involve large amount of resources, complex system architectures and heavy and high dynamic workload. This makes it particularly hard to cope with failures of any nature, to understand associated causes and to promptly perform proper actions to address associated problems.

In this paper, we propose a novel framework for proactive management and optimization of geographically-distributed applications which are affected by the presence of software anomalies. We focus on the case of applications deployed on IaaS (Infrastructure as a Service) cloud architectures composed of multiple and geographically-distributed cloud regions. The framework leverages, as a building block, our previous work described in [2], where we presented the PCAM (Proactive Client-server Application Management) Framework. PCAM is a framework designed for the case of a single cloud region, where a number of virtual machines (VMs) host server replicas of a client-server application. PCAM exploits machine learning (ML) models for predicting, at run-time, the Remaining Time to Failure (RTTF) of VMs, where failures of VMs are caused by accumulation of anomalies. A failure can be a crash, a violation of a given threshold of the response time, or (more generally) any situation where the VM does not adequately respond to user requests—these situations can be established by user. Based on RTTF predictions of VMs, PCAM proactively removes a VM from the pool of active VMs (i.e. the set of VMs serving client requests) when it is approaching a failure. Upon removing a VM, PCAM adds a new VM (which is available in a pool of stand-by VMs) to the pool of active machines. Additionally, it triggers a *software rejuvenation* [3] procedure for the deactivated VM (e.g. by restarting the VM) to remove accumulated anomalies in order to restore it to a correct working state. Once this procedure terminates, the VM is added to the pool of stand-by VMs and becomes available to replace an active VM when it is approaching a failure. By using the above-described strategy, PCAM constantly ensures efficiency of all VMs in the poll of active VMs. Ultimately, PCAM can improve system availability and guarantee adequate average system response time, although VMs are subject to accumulation of anomalies. For technical details and experimental results regarding PCAM, we refer the reader to [2]. Further, as for details on ML-based prediction models used by PCAM, we refer to [4].

As we discussed, PCAM has been designed for a single cloud region. In this paper, we consider a geographically-distributed cloud architecture composed of a number of cloud regions. We assume to have in each cloud region an *Autonomic Cloud Manager (ACM)* which uses an instance of PCAM to manage local VMs. Cloud regions can be either public or private, thus they can also form hybrid cloud infrastructures. We call the new framework that we present in this paper *Inter-ACM Framework (I-ACMF)*. Basically, the contribution of I-ACMF is twofold: 1) I-ACMF extends the usability of PCAM to the case of a geographically-distributed cloud architecture, and 2) it introduces a novel load balancing strategy across different cloud regions based on the Mean Time to Failure (MTTF) of VMs. Specifically, as for the first contribution, I-ACMF enables to create, in a straightforward way, large scale deployments of applications on dynamic multi-cloud architectures. I-ACMF allows to dynamically add or remove cloud regions (also belonging to different cloud providers) at run-time, in order to cope, e.g., with overall system workload fluctuations. As for the second contribution, I-ACMF is able to proactively distribute the workload across cloud regions, so as to balance the overhead associated with proactive management of failures of VMs in different regions, and to avoid overloaded regions. Particularly, I-ACMF can balance the workload also accounting for different anomaly occurrences in different regions and different computing power of regions.

I-ACMF leverages a distributed architecture composed of a set of controllers, one for each cloud region, each one connected to the local PCAM instance. A controller monitors the values of the predicted MTTF of each VM in the pool of

active VMs of its region. These values are used to evaluate the average MTTF of VMs of the cloud region, which we refer to as RMTTF. The RMTTF allows to calculate the average VM failure rate of the region (i.e. the reciprocal of the RMTTF). This rate is used as an indicator to evaluate if the cloud region is able to properly respond to the current workload. In fact, the average VM failure rate of a region is affected by various factors, including the current workload of the region, the available computing power in the region (e.g. the number of VMs and their configurations), the specific anomalies affecting the application software components deployed in the regions. As an example, since the workload of a region is spread across VMs of the local pool of active VMs, in a region with a small pool, each VM is subject to higher workload. This may increase the anomaly generation rate of VMs of the region, thus also the average VM failure rate of the region.

We note that, although PCAM can ensure the efficiency of active VMs of a regions, this comes at the cost of performing rejuvenation procedures and activation/deactivation of VMs. Thus, when a region is subject to higher average VM failure rate, the management overhead of PCAM grows. In other scenarios, when some regions have low computing power, or they are subject to different anomalies with respect to other ones, some regions may be excessively overloaded (thus reducing service availability and causing performance loss), while resources of other regions may be underutilized. I-ACMF copes with these problems by using the average VM failure rate of regions for both local a global purposes. Locally, it uses the average VM failure rate of a region to determine when additional VMs should be added to the pool of active VMs (or when the size of the pool can be reduced) to ensure adequate availability and performance in the region. Globally, I-ACMF uses the average failure rates of all cloud regions to decides how to balance the workload across regions. The strategy used by I-ACMF aims at ensuring that all available regions of the multi-cloud architecture show the same RMTTF (thus also the average VM failure rate). Overall, the I-ACMF approach can prevent various undesired situations and provide various advantages:

- Locally, it improves both availability and performance of a region.
- Globally, it can avoid situations where some regions show lower availability and lower performance with respect to other ones.
- Workload is automatically balanced with respect to different computing power of regions, and to different anomaly occurrences in different regions.
- When a VM joins (leaves) a region, the region workload is automatically spread across local VMs (by the local PCAM instance), and, consequently, the global workload is automatically balanced across regions (according to the new global configuration) by I-ACMF, which detects the variation of the RMTTF caused by the VM join (leaving).
- Cloud regions can be added or removed by simply calling *join* or *leaving* operations. I-ACMF automatically manages the new global cloud configuration, and it automatically spreads the workload across regions based on the variation of the RMTTF of all regions of the new configuration.

Overall, I-ACMF aims at simplifying the deployment and the management of applications on multi-cloud architectures. Indeed, I-ACMF has been designed to transparently perform various tasks to improve availability and provide optimal performance at both local and global level. Particularly, I-ACMF aims at sparing system administrators from the burden of performing complex deployments, configurations and recovery tasks in front of the presence of software anomalies and various kinds of failures.

The remainder of this paper is structured as follows. In Section II we discuss related work. Section III discusses the design and implementation choices of our Framework. Experimental data to assess the validity and viability of our proposal are reported in Section IV. Section V draws the conclusions.

II. Related Work

In the context of management of resources for cloud computing environments, several proposals have been presented in order to manage the accumulation of anomalies via proactive software rejuvenation [3]. Nevertheless, our work stands as an innovative solution, because it can manage any number of VMs, even geographically distributed, without any constraint on the topology of the distributed deploy of the application. Specifically in the context of hybrid cloud environments, the work in [5] proposes a hybrid cloud computing model to make the best use of public cloud services along with privately-owned data centers. The paper presents as well a workload factoring service designed for proactive workload management. In [6], a resilient hierarchical distributed loop self-scheduling algorithm able to cope with VMs crashes is presented. Contrarily to these works, in our proposal we enforce proactive rearrangement of the cloud organization, and we are further able to redirect incoming requests to different geographical areas, so as to reduce the impact of software rejuvenation of availability of the system.

A work similar in spirit to our one is presented in [7]. This paper proposes a capacity allocation algorithms which can coordinate multiple distributed resource controllers operating in geographically distributed cloud sites, coupled with a load redirection mechanism which distributes incoming requests among different sites. Nevertheless, the main focus of the proposal in [7] is to reduce the cost of allocated resources. In [8], custom interfaces for implementing policies and provisioning techniques for allocation of VMs under inter-networked Cloud computing scenarios are presented. Compared to these works, we offer a transparent deploy of virtualized applications on a geographical scale, offering at the same time an increase in the availability of the system.

In [9], workload forecasting and optimal resource allocation is studied. This is done by illustrating a model-predictive algorithm for workload forecasting that is used for resource auto scaling. Similarly, the work in [10] presents a provisioning technique that automatically adapts to workload changes related to applications for facilitating the adaptive management of system and offering end-users guaranteed Quality of Services (QoS) in large, autonomous, and highly

dynamic environments, using an analytic model. As well, in [11] Markovian Arrival Processes are used for the same purpose. Statistical models, for the same goal, are presented in [12]. Differently from these proposals, we offer a complete framework which, being based on ML techniques, allows for the integration with any virtualized application. In particular, transparent deploy at a geographical scale, with self-tuning capabilities and proactive management of the workload are specific differences with these proposals.

In [13], the authors present a simulation framework for online capacity planning of cloud-based in-memory data stores. This work relies as well on ML methods, but only to determine network latency, while other aspects proper of the application are predicted using a simulative/analytic model. Contrarily, we broaden the applicability of our proposal to any kind of application, not only in-memory data stores. Furthermore, we rely only on ML techniques, which can capture hidden dynamics of the application.

III. The I-ACMF Architecture and Implementation Details

I-ACMF allows to manage distribute resources over cloud regions in different geographical locations. Resources may belong to different cloud providers, private clouds, as well as to hybrid infrastructures. In each cloud region, I-ACMF manages a set of VMs. Each VM hosts a server replica of the application. According to the PCAM protocol described in [2], each VM of a cloud region can be either in the `ACTIVE` state or in the `STANDBY` state. VMs in the `ACTIVE` state process requests coming from remote clients. Upon receiving an activation command, a `STANDBY` VM switches to the `ACTIVE` state and starts to process incoming requests. The activation command is sent by PCAM to a `STANDBY` VM whenever an `ACTIVE` VM has to undergo the rejuvenation procedure (because it is predicted approaching a failure), or when whenever I-ACMF decides that a higher number of VMs is necessary to process local workload of the cloud region. Conversely, if I-ACMF decides that the pool of active VMs has to be reduced, it changes the state of an active VM to the `STANDBY` state.

In Figure 1, we show the I-ACMF architecture for a scenario with three cloud regions. Different colors are used to represent the components belonging to different cloud regions. In each cloud region there are two main components:

- A **Controller**, which is in charge of managing local resources of the cloud region and of communicating with controllers of other cloud regions.
- A **Load Balancer** (LB), which receives requests from remote clients and forwards them to local VMs or to LBs of other cloud regions. LB receives from CON information about fractions of workload to forward locally and to LBs of other cloud regions.

The internal architecture of a cloud region is shown in Figure 2. In the I-ACMF architecture, LBs act as entry points of the application for remote clients. We note that having multiple LBs in the system is typical of several real-world scenarios. In fact, applications deployed on geographically scale usually have multiple entry points, which can be reached by clients through, e.g., directory services, such as DNS servers.

In the follow, we provide specific details about I-ACMF architecture components and their implementations.

A. MTTF Prediction Models

As discussed, I-ACMF leverages ML models provided by PCAM for predicting the MTTF of each active VM in a cloud region. Here, we briefly provides some details about ML models generation. These models are generated by a training process which exploits data related to run-time measurements of a set of system features (such as memory and CPU usage) and events related to VM failures, which are collected while observing VMs running. To reduce the set of system features used to build prediction models, PCAM uses Lasso regularization [14]. This allows to reduce both the training time of ML models and the overhead for VM monitoring at run-time. PCAM uses WEKA [15] to generate differentiated ML prediction models. Particularly, PCAM produces models by using the following algorithms: Linear Regression [16], M5P [17], REP Tree [18], SVM [19], LS-SVM [20] and Lasso as a predictor [14]. Predictions models produced by the training process are complemented by a set of indicators which allow the user to chose the most suitable MTTF model for the application. For each prediction model, the following indicators are provided: Maximum Absolute Prediction Error (MAPE), Relative Absolute Prediction Error (RAPE), Relative Absolute Prediction Error (RAPE), Relative Absolute Prediction Error (RAPE), Relative Absolute Prediction Error (RAPE), Mean Absolute Error (MAE) and Soft-Mean Absolute Error (S-MAE). We refer the reader to [4] for a thorough description of the training process and the provided indicators.

B. Global Workload Balancing Strategy

In the I-ACMF architecture, one controller acts as a leader. I-ACMF relies on the leader election algorithm presented in [21]. It is an efficient and scalable algorithm and is highly resilient to changes in the network of participants. By relying on this algorithm, I-ACMF is able to enforce dynamic network reconfiguration, even in case of multiple node and link failures in high-speed networks with arbitrary topology.

The leader collects, from other controllers, data related to incoming client request rate and to RMTTF of each cloud region. The controller of a cloud region periodically sends these data to the leader. Upon receiving updated data from controllers, the leader decides how to distribute the current global workload across cloud regions. Specifically, it decides the fraction of the global workload to be assigned to each cloud region. This is calculated based on the *Sensible Routing* approach, as described in [22]. In detail, upon receiving updated data from a cloud region i at time t, the RMTTF of the cloud region, say $RMTTF_i^t$, is calculated by using a weighted average. Thus, upon receiving the n^{th} measurement, it is calculated as:

$$RMTTF_i^n = (1-\beta) \cdot RMTTF_i^{n-1} + \beta \cdot RMTTF_i^n, \quad (1)$$

where $0 \leq \beta \leq 1$.

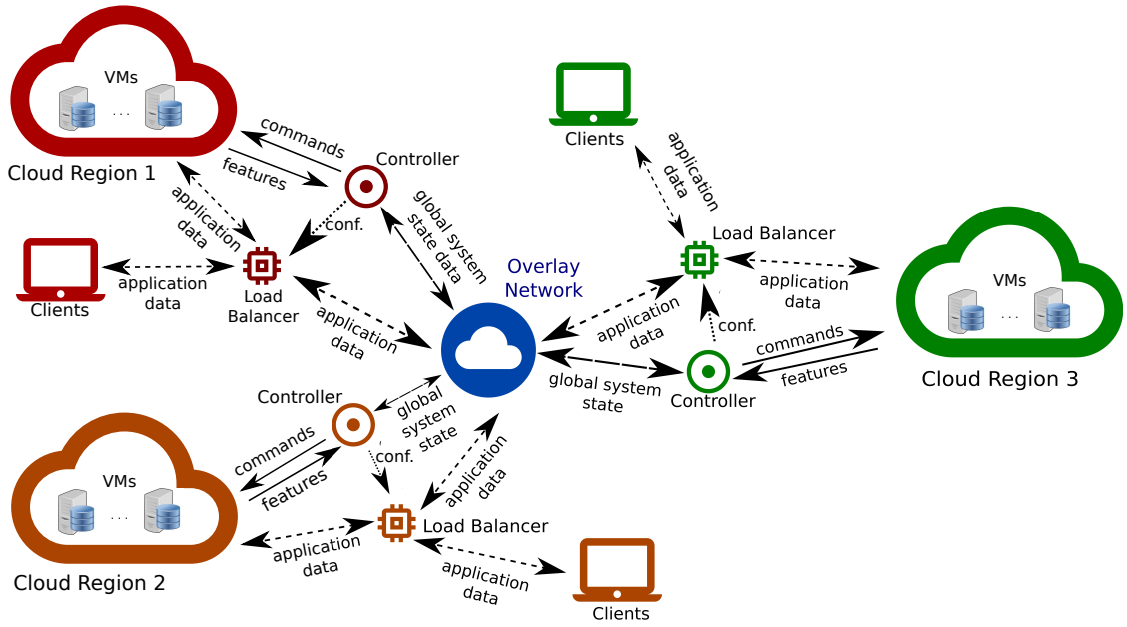

Figure 1. Global System Architecture with 3 Cloud Regions

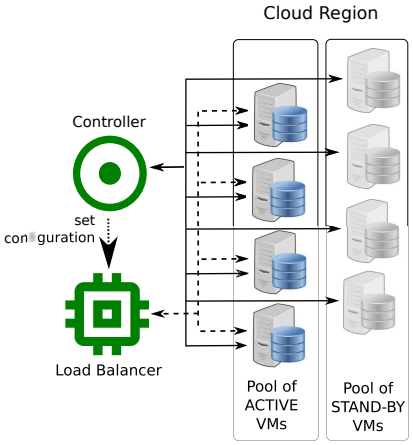

Figure 2. Internal architecture of a Cloud Region

Assuming to have N cloud regions, the fraction f_i of global incoming client requests to be forwarded to cloud region i is calculated as:

$$f_i = \frac{RMTTF_i^n}{\sum_{j=1}^{N} RMTTF_j^n} \quad (2)$$

We note that, using this approach, if a cloud region shows a higher RMTTF (i.e. lower VM failure rate), then I-ACMF forwards to this cloud region a higher fraction of workload with respect to cloud regions with lower RMTTF (i.e. higher VM failure rate). The effect of this policy is to level the RMTTF of cloud regions.

Once calculated f_i for each region, I-ACMF calculates a global forward plan. This plan establishes, for each cloud region, the fractions of incoming client requests received by the local LB to be forwarded to local VMs and to be forwarded to LBs of other cloud regions. I-ACMF uses the following algorithm. Assume that s_i is the fraction of incoming client requests of LB of the cloud region i with respect to the global incoming client requests. In the first step, the algorithm evaluates, for each LB of each cloud region, if $s_i \leq f_i$. If yes, all client requests received by LB are locally forwarded. For cloud regions for which the previous condition in not true, a fraction of incoming client requests of the cloud region has to be forwarded to other cloud regions. Then, in the second step, for each cloud region for which $s_i > f_i$, the algorithm distributes the fraction of requests $s_i - f_i$ to other cloud regions for which the already assigned total fraction of incoming client requests is less then f_j, where j is the cloud region identifier. Then, the final output of the algorithm is sent to LBs, which start to forward local incoming client requests according to the new global forward plan.

C. Proactive Activation/Deactivation of VMs

When the global workload increases, the failure rate of VMs in one or multiple cloud regions may increase, so that excessive performance loss and low availability may be experienced by clients. As a countermeasure to this issue, I-ACMF can proactively change the number of active VMs in each cloud region. If the RMTTF of a cloud region becomes less (more) than a given threshold, then the local controller can activate news VM (deactivate some active VMs) by using MTTF prediction models to evaluate the expected RMTTF as a result of the VM activation (deactivation).

IV. Experimental Evaluation

In this section, we present results of an experimental study we conducted to assess I-ACMF. We show results for the case an hybrid cloud architecture, where we used three cloud regions: Region 1, hosted in the Ireland Region of Amazon EC2, Region 2, hosted in the Frankfurt Region of Amazon EC2, and Region 3, privately hosted in a 32-cores HP ProLiant server with 100 GB RAM, located in Munich (Germany). We used

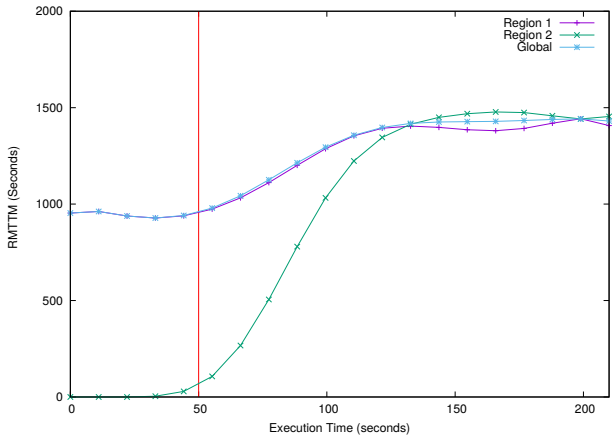

Figure 3. RMTTF variation when a new cloud region joins the system

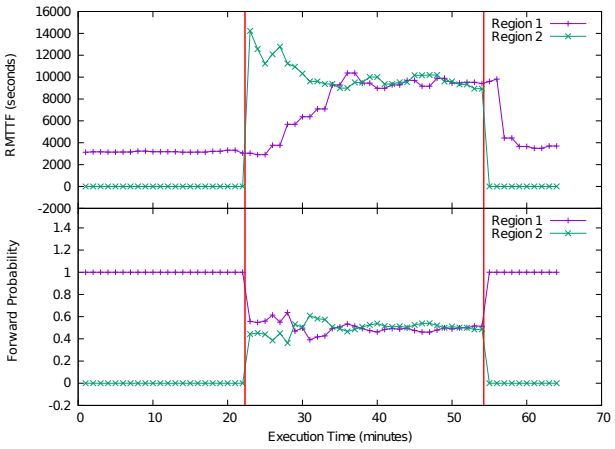

Figure 4. MTTF and Forward Probability, 2 regions, Constant rate (500 requests per sec)

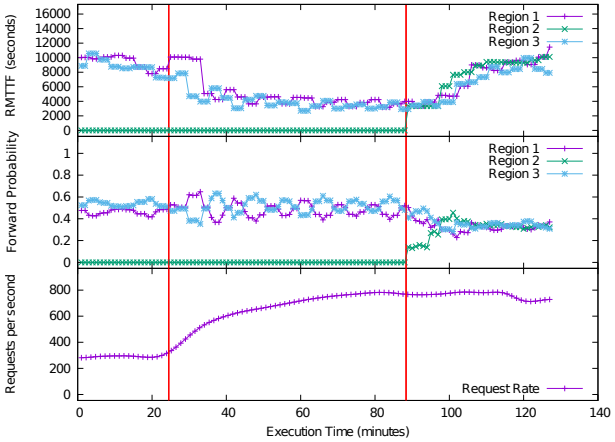

Figure 5. Request rate, MTTF, and Forward Probability, 3 regions, variable rate

`m3.medium` Amazon EC2 instances in Region 1 and Region 2, and VMs with equivalent configuration in Region 3. The HP ProLiant server was equipped with VMware Workstation 10.4 as a hypervisor. All VMs were equipped with Ubuntu 10.04 Linux Distribution (kernel version 2.6.32-5-amd64). The test-bed application was the TPC-W benchmark [23], a multi-tier e-commerce web application that simulates an on-line book store. We used a Java implementation of TPC-W [24] developed using servlets, and relying on Mysql [25] as a database server. Clients were emulated using emulated web browsers to generate requests according to TPC-W specifications. We modified the TPC-W implementation for randomly generating software anomalies at run-time, including memory leaks and unterminated threads. Specifically, anomalies were generated with different probabilities on each VM when receiving a client request. This led to scenarios where each VM (thus each cloud region) was showing different anomaly occurrence patterns. We varied the number of active clients (towards each cloud region) in the interval $[16, 512]$. Based on our previous result in [4], we selected REP Tree as a ML model for predicting the MTTF.

As a preliminary assessment, in Figure 3, we show the effects on the RMTTF when a cloud region joins the system. We started the experiment using only Region 1. After about 50 seconds (the time is marked by the vertical red line in the figure) Region 2 joins the system. At this point, the two controllers start exchanging information. By the figure, we can see that, before Region 2 joins the system, the RMTTF of Region 1 is about 1000 seconds (also, it is equal to the global RMTTF). After Region 2 joins the system, the RMTTF of Region 1 and Region 2 start to converge to comparable values. Overall, the global RMTTF increases. This shows that I-ACMF has been able to distribute the global workload between the two regions, in a way to even out the RMTTF of the two cloud regions and to reduce the global VMs failure rate.

In Figure 4, we show some results for demonstrating how the load balancing approach of I-ACMF works. We used Region 1 and Region 2. In this experiment, the emulated web browsers are connected only to Region 1. Initially, only Region 1 is connected to the system, thus all client requests are processed by Region 1. We note that, initially, the RMTTF of Region 1 is about 3000 seconds and the forward probability of client requests (i.e. the fraction of global workload) towards Region 1 is equal to 1. After about 22 minutes (first red line in the figure), Region 2 joins the system, thus I-ACMF starts to balance the workload. Just after the join, the RMTTF of Region 2 is higher than Region 2, given that all VMs in Region 2 are in a "clean" state (i.e. no anomalies are accumulated). Thus, the forward probability rapidly increases for Region 2 and decreases for Region 1. Around minute 34, the system reaches an equilibrium: the forward probability is constant (about 0.5 per region), and the RMTTF of the two regions becomes almost the same. Around minute 55 (second red line in the figure), Region 2 leaves the system. Then, given that I-ACMF can no longer distribute the workload, the RMTTF of Region 1 returns to the same low value as in the first part of the experiment.

In Figure 5, we report the results of an experiment using all three cloud regions. Differently from the previous scenarios, in this experiment the load generated by the emulated web browsers connected to Region 1 changes over time (while it is constant for Region 2). In particular, after minute 22 (first red line in figure), the client request generation rate increases from 300 to about 700 requests per second. This highest value is reached around minute 75. Initially, there are

only Region 1 and Region 3 (Region 2 joins the system after about 90 minutes - second red line in figure). We note that, while the incoming request rate increases, the RMTTF of both regions decreases. However, although the incoming request rate increases only for Region 1, I-ACMF is able to keep balanced the RMTTF of both cloud regions. Finally, when Region 2 joins the system, the RMTTF of the all cloud regions starts to increase, as long as the forward probabilities changes and reaches an equilibrium. This shows that I-ACMF has been able to cope with different incoming client request rates in different cloud regions, as well as with variations due to scale up/scale down of the system.

V. Conclusions and Future Work

In this paper, we presented I-ACMF, a framework for proactive management and optimization of application subjects to software anomalies and deployed on distributed geographical (hybrid) multi-cloud infrastructures. As building block. I-ACMF exploits machine learning models for predicting the mean time to failure of virtual machines of different cloud regions caused by accumulation of anomalies. Overall, I-ACMF uses a combination of strategies and techniques, such as a proactive rejuvenation, a workload balancing approach based on virtual machine failure rates, and a proactive system scale up and scale down technique. Experimental results show that I-ACMF can simplify application deployment and management, also in front of non-uniform client request rates in different cloud regions, workload variations and system scale up/scale down at both cloud region level and global system level. As a future work, we plan to address issues related to communication over geographical scale in I-ACMF. We plan to use an overlay network to support reliable and efficient communication between cloud regions.

Acknowledgements

The research presented in this paper has been supported by the European Union via the EC funded project PANACEA, contract number FP7 610764.

References

[1] S. Pertet and P. Narasimhan, "Causes of Failure in Web Applications," Carnegie Mellon University, Tech. Rep. CMU-PDL-05-109, 2005.

[2] P. Di Sanzo, A. Pellegrini, and D. R. Avresky, "Machine Learning for Achieving Self-* Properties and Seamless Execution of Applications in the Cloud," in *Proceedings of the Fourth IEEE Symposium on Network Cloud Computing and Applications*, ser. NCCA. IEEE Computer Society, 2015.

[3] D. Cotroneo, R. Natella, R. Pietrantuono, and S. Russo, "Software aging and rejuvenation: Where we are and where we are going," in *Proceedings - 2011 3rd International Workshop on Software Aging and Rejuvenation, WoSAR 2011*, 2011, pp. 1–6.

[4] A. Pellegrini, P. Di Sanzo, and D. R. Avresky, "A Machine Learning-based Framework for Building Application Failure Prediction Models," in *Proceedings of the 20th IEEE Workshop on Dependable Parallel, Distributed and Network-Centric Systems*, ser. DPDNS. IEEE Computer Society, 2015.

[5] H. Zhang, G. Jiang, K. Yoshihira, and H. Chen, "Proactive workload management in hybrid cloud computing," *Network and Service Management, IEEE Transactions on*, vol. 11, no. 1, pp. 90–100, March 2014.

[6] Y. Han and A. T. Chronopoulos, "A Resilient Hierarchical Distributed Loop Self-Scheduling Scheme for Cloud Systems," in *IEEE 13th International Symposium on Network Computing and Applications*, 2014.

[7] D. Ardagna, S. Casolari, M. Colajanni, and B. Panicucci, "Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems," *Journal of Parallel and Distributed Computing*, vol. 72, no. 6, pp. 796–808, 2012.

[8] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software - Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[9] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, 2011, pp. 500–507.

[10] R. N. Calheiros, R. Ranjan, and R. Buyya, "Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments," *2011 International Conference on Parallel Processing*, pp. 295–304, 2011.

[11] S. Pacheco-Sanchez, G. Casale, B. Scotney, S. McClean, G. Parr, and S. Dawson, "Markovian workload characterization for QoS prediction in the cloud," in *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, 2011, pp. 147–154.

[12] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, "Statistics-driven workload modeling for the cloud," in *Proceedings - International Conference on Data Engineering*, 2010, pp. 87–92.

[13] P. Di Sanzo, F. Quaglia, B. Ciciani, A. Pellegrini, D. Didona, P. Romano, R. Palmieri, and S. Peluso, "A Flexible Framework for Accurate Simulation of Cloud In-Memory Data Stores," *Simulation Modelling Practice and Theory*, 2015.

[14] R. Tibshirani, "Regression Shrinkage and Selection Via the Lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1994.

[15] I. H. Witten, E. Frank, L. E. Trigg, M. A. Hall, G. Holmes, and S. J. Cunningham, "Weka: Practical machine learning tools and techniques with Java implementations," 1999.

[16] K. P. Murphy, *Machine Learning: a Probabilistic Perspective*. MIT Press, 2012.

[17] Y. Wang and I. H. Witten, "Inducing Model Trees for Continuous Classes," in *Procedings of the 9th European Conference on Machine Learning*, 1997, pp. 128–137.

[18] H. A. Chipman, E. I. George, and R. E. Mcculloch, "Extracting Representative Tree Models From a Forest," in *IPT Group, IT Division, CERN*, 1998, pp. 363–377.

[19] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[20] J. A. K. Suykens and J. Vandewalle, "Least Squares Support Vector Machine Classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[21] D. R. Avresky and N. Natchev, "Dynamic reconfiguration in computer clusters with irregular topologies in the presence of multiple node and link failures," *IEEE Transactions on Computers*, vol. 54, no. 5, pp. 603–615, 2005.

[22] L. Wang and E. Gelenbe, "Adaptive dispatching of tasks in the cloud," *CoRR*, vol. abs/1501.00567, 2015. [Online]. Available: http://arxiv.org/abs/1501.00567

[23] W. D. Smith, "TPC-W: Benchmarking an ecommerce solution," 2000.

[24] T. Bezenek, T. Cain, R. Dickson, T. Heil, M. Martin, C. McCurdy, R. Rajwar, E. Weglarz, C. Zilles, and M. Lipasti, "Characterizing a Java implementation of TPC-W," in *Proceedings of the Third Workshop On Computer Architecture Evaluation Using Commercial Workloads*, 2000.

[25] MySQL, "MySQL database server," http://www.mysql.com, 2004.