Machine Learning-based Management of Cloud Applications in Hybrid Clouds: a Hadoop Case Study

D. R. Avresky
IRIANC
Munich, Germany / Boston, MA, USA
autonomic@irianc.com

Alessandro Pellegrini
Pierangelo Di Sanzo
Sapienza, University of Rome
Rome, Italy
{pellegrini,disanzo}@dis.uniroma1.it

***Abstract*—This paper illustrates the effort to integrate a machine learning-based framework which can predict the remaining time to failure of computing nodes with Hadoop applications. This work is part of a larger effort targeting the development of a cloud-oriented autonomic framework to increase the availability of applications subject to software anomalies, and to jointly improve their performance. The framework uses machine-learning, software rejuvenation, and load distribution techniques to proactively prevent failures. We believe that this work allows to set a possible path towards the definition of best practices for the development of systems to support autonomic management of cloud applications, illustrating what are the issues that should be addressed by the research community. Indeed, given the scale and the complexity of modern computing infrastructures, effective autonomic management approaches of cloud applications are becoming mandatory.**

Keywords—autonomic; monitoring; cloud; rejuvenation; Hadoop; best practices

I. INTRODUCTION

This paper presents the results of the integration between an cloud application management framework, called Autonomic Cloud Manager (ACM) framework [21], [23], and applications based on Hadoop. ACM is a framework which, by relying on runtime statistics and machine learning (ML)-based models, allows to predict the remaining time to failure of computing nodes of applications on any number of cloud-based facilities. The framework is designed in such a way that computing nodes can be scattered on multiple geographically-distributed regions. By relying on load balancing techniques and overlay networks, the system is able to redirect any traffic across the different computing nodes. Particularly, it transparently hides away failures of computing nodes of the application by predicting faults and by rejuvenating any node which is approaching a failure point. Also, it spawns new nodes to join the network, in order to promptly replace nodes to be rejuvenated. The ACM framework can predict failures due to software anomalies, such as memory leaks and/or unterminated threads.

Apache Hadoop [17] is an open-source software framework used for distributed storage and big data processing. It is designed to exploit computer clusters built from commodity hardware. The core of Apache Hadoop includes a storage part, known as Hadoop Distributed File System (HDFS), and a processing part, based on the MapReduce programming model. Hadoop splits files into large blocks and distributes them across nodes in the cluster. It then transfers packaged code into nodes to process the data in parallel. This approach takes advantage of data locality [18] where computing nodes manipulate the data they have access to. This allows the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking.

In this paper, we illustrate our effort to integrate the ACM framework with cloud applications based on Hadoop. Our choice fell onto Hadoop due to the wide diffusion of this framework. Primarily, our effort aims at understanding the current limitations and the possible improvements required to integrate state-of-the-art autonomic cloud application management frameworks with a wide employed class of applications. As we will show, by the organization of Hadoop, the system we built as a result of our integration effort allowed us only to transparently determine the effects of software anomalies, as a step towards the possibility to increase the availability of generic Hadoop-based applications. This illustrates the need for clear future technical steps to be taken.

The remainder of this paper is structured as follows. Section II provides a short recap on the ACM framework. Section III presents how the integration with Hadoop applications has been carried out and the implications of the integration. Section IV discusses related work. Section V draws the conclusions, Recap on the ACM framework

As mentioned, the ACM framework [21], [23] targets a geographically-distributed cloud architecture composed of a number of cloud regions. Cloud regions can be either public or private, thus they can also form hybrid cloud infrastructures. The ACM framework enables to manage, in a straightforward zmulti-cloud architectures. It 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.

978-1-5386-1465-5/17/$31.00 ©2017 IEEE

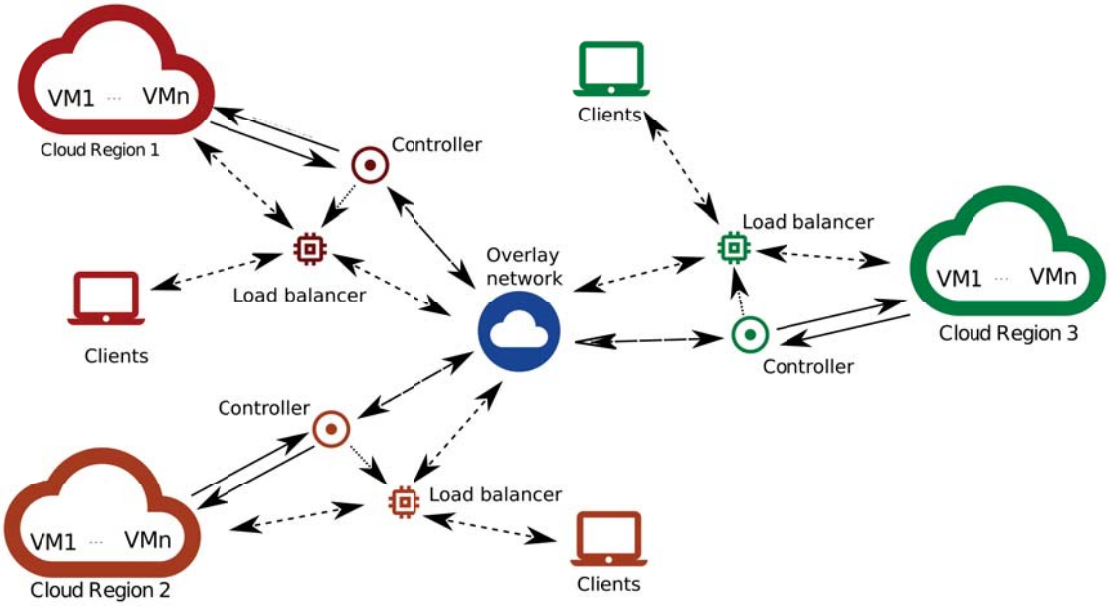

Fig. 1. The ACM framework with overlay network in a hybrid cloud composed of 3 regions.

Additionally, it is able to proactively distribute the workload across cloud regions, so as to balance both the workload and the overhead associated with the proactive management of failures of computing nodes, and to avoid overloaded regions. Particularly, it can balance the workload also accounting for different anomaly occurrences in different regions and different computing power of regions.

While for the technical details on the ACM framework we refer the reader to [23], we report in Fig. 1 the high-level \cloud composed of three geographically-distributed regions. We consider the case where a node of the application runs on a single Virtual Machine (VM). The main components of the ACM framework are the *Control Unit* (or *Controller*), and the *Load Balancer*. The former implements an autonomic monitoring and controlling system, while the latter distributes the load by managing incoming connections from clients. ACM allows to increase the availability of applications and keeps their response time below a certain threshold by exploiting several techniques in a joint manner:

- Replication of virtualized resources: at a single cloud region, a computing node of the application is replicated on a pool of VMs. A pool can be seen by as a single (virtualized) computing node of the application.
- Software rejuvenation: when a VM in a pool is approaching a failure, a new VM is proactively joined to the pool (taking the place of the about-to-fail one), and the failing one is rejuvenated. The application sees the new VM as if it is the *same* as the rejuvenated one.

In Fig. 1, we present an example deployment of the framework in the case of 3 cloud regions. The overlay network, depicted in the center of the figure, allows to determine what is the best-suited communication path across the regions, when an incoming connection at one end is redirected towards a remote node, in order to optimize the response time.

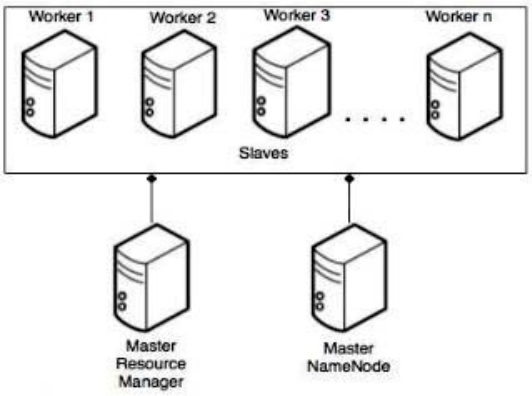

Fig. 2. Hadoop Cluster Configuration.

II. Integration of Hadoop with The ACM framework

An example of system architecture for an Hadoop-based application is shown in Fig. 2. Hadoop implements the MapReduce programming model. The Master Resource Manager and the Master NameNode control the evolution of the computation, and manage available computing resources. A number of workers carry out the computational tasks of the

application, which are therefore distributed across the nodes in data separation. At the end of the computation, the results are gathered and are delivered to the user.

Our first step for the integration of a generic application with the ACM framework was the construction of ML-based prediction models of the Remaining Time to Failure (RTTF) of VMs hosted in the cloud infrastructure where the application is deployed. We remark that the ACM framework allows to build prediction models of applications that are subject to software anomalies, whose accumulation over time can lead to failure of computing nodes.

For experimental purposes, we set up 5 VMs on Amazon EC2 Cloud Hosting, using the simple PI computation benchmark provided by Hadoop. We modified this benchmark so as to last longer and to artificially produce and accumulate memory leaks. This allows to meet the required experimental conditions for evaluating the ACM framework, and therefore to create ML-based models to predict when the VMs are about to fail. In our test case, VMs failures are due to the exhaustion of memory.

A. Building Prediction Models

As for the components of the ACM framework within a cloud region, we installed the Virtual Machine Controller (VMC) on a dedicated node of each region, and the Local Managing Unit (LMU) on all worker nodes of the Hadoop cluster of each region. The high level architecture and connections of these two components are shown in Fig. 3. Measurement Unit is the component in charge of monitoring, during the lifetime of the application, all system features (e.g., memory usage, CPU usage) which are used either for training ML models or to predict the RTTF at run-time.

During the data collection phase required to populate the model training sets, we varied the incidence of memory leaks on the application. This has been done by varying the inter arrival time (namely, the sleep period among two consecutive leak generations) so as to have a VM saturating the memory after different amounts of time per each run. The inter arrival time was selected using a random uniform distribution in between 15 and 35 seconds. To collect the training data, we configured LMU to consider a VM failed when the limit for Java Heap was reached. To show some example of measurements collected by LMU along one run of the application, we report in Fig. 4 the measured values of some system parameters monitored by LMU.

After the data collection phase, we run the ACM framework to build the ML-based prediction models. The first step entails running Lasso Regularization [24] to select only the most relevant parameters to the construction of prediction models. The parameters which have been selected by Lasso for the Hadoop, with the related weights, scenario are reported in Table 1. These results show that Lasso correctly identified that, in our specific case, the system fails only due to memory saturation. In fact, the only type of anomaly injected in the system were memory leaks, and Lasso correctly discarded any parameter not necessarily related to memory.

The ACM framework builds various linear prediction models from the data selected by relying on Lasso Regularization. While we again refer the reader to [21] for a thorough discussion on the prediction models, we report for the sake of clarity in Fig. 5 the output of the prediction model generated by the REP-Tree based model [25]. On the x-axis of the plot we have the real RTTF, while on the y-axis we have the RTTF predicted by the model. The straight line denotes the "right" prediction.

B. Predicting the RTTF

It is interesting to note that, as soon as the actual RTTF gets close to the actual failure point, the predicted RTTF is able to approximate the real RTTF with good precision (for brevity, we do not report data related to the prediction error, yet we refer the reader to [21] for a discussion).

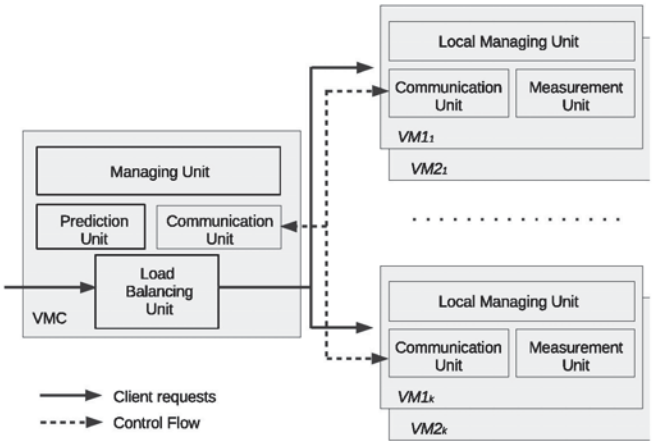

Fig. 3. Basic architecture components and interactions within a cloud region of the ACM framework.

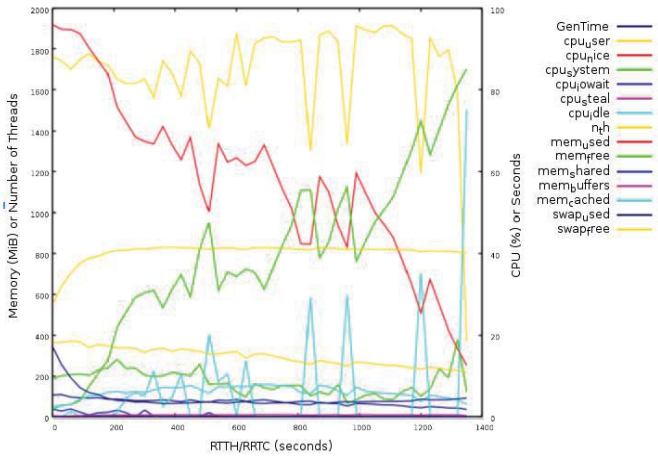

Fig. 4. System Parameters measured by LMU.

TABLE I. PARAMETERS SELECTED BY LASSO

PARAMETER	WEIGHT
mem_free	0.000300
mem_cached	0.001127
swap_used	-0.000132
swap_free	0.000152

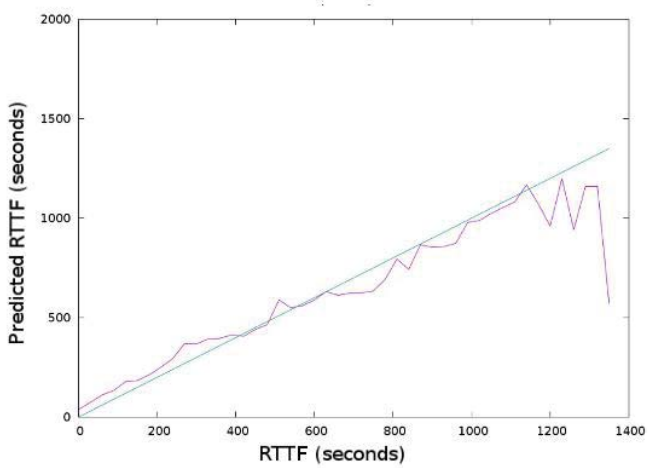

Fig. 5. Prediction model fitted using REP-Tree.

Anyhow, in order for the ACM framework to support the availability of the Hadoop deployment, it should be possible to dynamically add new VMs to replace about-to-fail ones, and remove the latter to rejuvenate them. Unfortunately, by itself, Hadoop does not allow this *join/leave pattern* required for software rejuvenation *at any point in computation*. Indeed, if a VM is added, the Hadoop NameNode and Resource Manager—see Fig. 2—must be informed of the availability. Similarly, depending on the application, removing a VM might require notifying the NameNode. In any case, the running node which should be removed may not have finished its computing task. In this case, some portion of the computation might be lost. While Hadoop allows to recover portion of the computation in this scenario, the ACM framework is a lower-level component that has no clue about the nature of the application it is managing.

Therefore, it is clear that any complex (distributed) application which has to be integrated with an autonomic management system, like the ACM framework, should expose some of its internal organization to the manager. The Hadoop integration effort described in this paper therefore calls for the existence of either some sort of *integration middleware*, or some sort of *standardized API for autonomic management*.

The possibility of predicting the actual RTTF of a generic application in an autonomic way is an important aspect which has been deemed fundamental for large-scale computations several times in the literature (see, e.g., [3], [4], [6], [8], [23]). Nevertheless, in order to cope with the large variety of applications which are available nowadays, some sort of standardized "autonomic communication layer" should be designed with an agreement across software producers.

Without this kind of layer, in order to support the integration of the ACM framework with the Hadoop application which we have used as a testbed, we had to modify the application itself and to realize an additional (although simple) middleware. While this allowed us to reach the goal of increasing the availability of the application, it has not been an easy task. This clearly contrasts with the spirit of the ACM framework, as well as of Hadoop and, generally, of autonomic computing. To illustrate more in detail our effort, with our additional middleware, when a VM is deemed to be failing soon, Hadoop Resource Manager is asked to decommission it. Therefore, no new application or task of actual running applications will be scheduled to that node. At this point, the ACM framework can provide a new VM. By using our middleware, the new node is assigned to Hadoop and it can start assigning jobs to it. Therefore, we are able to provide a new VM to be added to an existing Hadoop cluster. If a running application has a subtask mapped to the node which has been rejuvenated, the application—hence the need to change the original code—can simply remap the task to the Hadoop cluster again. In fact, the Hadoop cluster has already received a new (rejuvenated) worker node.

III. RELATED WORK

Predicting the effect of application anomalies is not a new idea [2]. Along this path, several works have already proposed prediction techniques and models [5]. In [6], the authors propose a proactive prediction and control system for large clusters. The proposal relies on time series, rule-based classification, and Bayesian networks, to filter the initial data, selecting only the entries, which are useful to carry on a prediction. A framework to automatically detect anomalies and track the performance of an application is proposed in [7]. The framework relies on a regression-based transaction model, which reflects the resource consumption model of the application. On the front of enforcing autonomic actions to avoid the effects of application anomalies, we find in the literature several proposals. While more traditional approaches try to recover a previous state by undoing wrong operations (see, e.g., [16]), proposals to predict upcoming crashes by using decision rules generated at runtime in order to support software rejuvenation better match autonomic properties (see, e.g., [8]). There is also a set of commercial tools [9], [10], [11] which can be used to monitor Java applications by instrumenting the JVM.

In the context of cloud-based applications, several works have been proposed to use ML techniques either for optimizing the proper cloud organization (see, e.g., [22]) or to enforce proactive rejuvenation (see, e.g., [12], [13]). Specifically in the context of hybrid cloud environments, the work in [14] proposes a hybrid cloud computing model to make the best use of public cloud services along with privately-owned data centers. In [15], 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

IV. Conclusions and Future Work

We have presented in this paper our efforts towards the integration of a state-of-the-art autonomic cloud application management framework to contrast the effects of software anomalies with applications based on the well-known Hadoop framework. While we have shown that it is possible, in practice, to put in place some level of interoperability across Hadoop applications and frameworks such as ACM framework, we have shown that it is no easy task, given the fact that some internals of the Hadoop framework must be exposed to the autonomic monitoring system. We therefore conclude that in order to effectively enforce autonomic properties, the existence of either some sort of integration middleware, or some sort of standardized API for autonomic computing must be devised in the future.

Acknowledgment

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] 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.

[2] L. M. Silva, J. Alonso, and J. Torres, "Using Virtualization to Improve Software Rejuvenation," IEEE Trans. Comput., vol. 58, no. 11, pp. 1525– 1538, 2009.

[3] 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.

[4] O. Brun, L. Wang, E. Gelenbe, "Big Data for Autonomic Intercontinental Overlays," IEEE Journal on Selected Areas in Communications 34(3): 575-583, IEEE Computer Society, 2016.

[5] F. Salfner, M. Lenk, and M. Malek, "A Survey of Online Failure Prediction Methods," ACM Computing Surveys, vol. 42, no. 3, pp. 10:1—10:42, 2010.

[6] R. K. Sahoo, A. J. Oliner, I. Rish, M. Gupta, J. E. Moreira, S. Ma, R. Vilalta, and A. Sivasubramaniam, "Critical Event Prediction for Proactive Management in Large-scale Computer Clusters," in Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD. ACM, 2003, pp. 426–435.

[7] L. Cherkasova, K. Ozonat, N. Mi, J. Symons, and E. Smirni, "Anomaly? Application Change? or Workload Change? Towards Automated Detection of Application Performance Anomaly and Change," in Proceedings of the 2008 International Conference on Dependable Systems and Networks. IEEE Computer Society, 2008.

[8] D. Simeonov and D. R. Avresky, "Proactive Software Rejuvenation Based on Machine Learning Techniques," in Cloud Computing, ser. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, D. Avresky, M. Diaz, A. Bode, B. Ciciani, and E. Dekel, Eds. Springer Berlin Heidelberg, 2010, vol. 34, pp. 186–200.

[9] IBM Corporation, "Tivoli Web Management Solution," http://www.ibm.com/software/tivoli/.

[10] Hewlett-Packard, "HP Diagnostics," http://www8.hp.com/us/en/softwaresolutions/diagnostics-software/.

[11] Dell, "Foglight," http://www.quest.com/foglight-for-java/.

[12] J. Alonso, L. Belanche, and D. R. Avresky, "Predicting Software Anomalies Using Machine Learning Techniques," in Proceedings of the 2011 IEEE 10th International Symposium on Network Computing and Applications, ser. NCA. IEEE Computer Society, 2011, pp. 163–170.

[13] 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.

[14] H. Zhang, G. Jiang, K. Yoshihira, and H. Chen, "Proactive Workload Management in Hybrid Cloud Computing," 2014..

[15] 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.

[16] D. Cingolani, A. Pellegrini, F. Quaglia, "Transparently mixing undo logs and software reversibility for state recovery in optimistic PDES". ACM Transactions on Modeling and Computer Simulation (TOMACS) 27 (2), 11

[17] "Hadoop Releases". apache.org. Apache Software Foundation.

[18] "What is the Hadoop Distributed File System (HDFS)?". ibm.com. IBM.

[19] "Resource (Apache Hadoop Main 2.5.1 API)". apache.org. Apache Software Foundation.

[20] A. Murthy, "Apache Hadoop YARN: Concepts and Applications". hortonworks.com. Hortonworks.

[21] A. Pellegrini, P. Di Sanzo and D. R. Avresky, "Proactive Cloud Management for Highly Heterogeneous Multi-Cloud Infrastructures". In Proceedings of the 21st IEEE Workshop on Dependable Parallel, Distributed and Network-Centric Systems (DPDNS), Chicago, IL, USA, IEEE Computer Society, May 2016.

[22] P. Di Sanzo, F. Quaglia, B. Ciciani, A. Pellegrini, D. DIdona, P. Romano, R. Palmieri, S. Peluso, "A flexible framework for accurate simulation of cloud in-memory data stores". Simulation Modelling Practice and Theory 58, 219-238. Elsevier.

[23] D. R. Avresky, P. Di Sanzo, A. Pellegrini, B. Ciciani, L. Forte, "Proactive scalability and management of resources in hybrid clouds via machine learning". In Proceedings of the 2015 IEEE 14th International Symposium on Network Computing and Applications (NCA), IEEE Computer Society, 2015, pp. 114-119.

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

[25] 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.