Introducing Software Performance Antipatterns in Cloud Computing Environments: Does it Help or Hurt?*

Catia Trubiani
Gran Sasso Science Institute
L'Aquila, Italy
catia.trubiani@gssi.infn.it

ABSTRACT

Performance assessment of cloud-based big data applications require new methodologies and tools to take into consideration on one hand the volume, the variability and the complexity of big data, and on the other hand the intrinsic dynamism of cloud environments. To this end, we introduce software performance antipatterns as reference knowledge to capture the well-known bad design practices that lead to software products suffering by poor performance.

This paper discusses some of the challenges and opportunities of research while introducing software performance antipatterns in cloud computing environments. We present a model-based framework that makes use of software performance antipatterns to improve the Quality-of-Service (QoS) objectives of cloud-based big data applications.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling techniques, Performance Attributes; D.2.8 [**Software Engineering**]: Metrics—*performance measures*

Keywords

Software Performance Antipatterns; Cloud Computing Environments; Big Data Applications

1. INTRODUCTION

Cloud computing environments offer a variety of solutions and services to their customers in fact they provide new opportunities while performing the service provisioning, i.e. the capability of acquiring and releasing resources on demand. However, beside the advantages, cloud computing introduced new issues and challenges. In particular, the heterogeneity of the services offered while dealing with big data applications makes the process of identifying a deployment solution that minimizes costs and guarantees Quality-of-Service (QoS) very complex.

In last years many EU projects were targeting cloud environments and their quality assessment, for example:

- *Artist* (http://www.artist-project.eu) aims to migrate legacy software to gain improved performance from the service provisioning of cloud infrastructures;
- *MODAClouds* (http://www.modaclouds.eu/) aims to provide a run-time environment that guarantees QoS for applications deployed on multi-Clouds;
- *CloudScale* (http://www.cloudscale-project.eu/) aims to provide an engineering approach for building scalable cloud applications and services;
- *Cloud-TM* (http://www.cloudtm.eu/) aims to provide a data-centric middleware platform facilitating development and abating costs of cloud applications;
- *PaaSage* (http://www.paasage.eu/) aims to provide an open source integrated platform to support both design and deployment of Cloud applications;
- *SeaClouds* (http://www.seaclouds-project.eu/) aims to guarantee agility after deployment by considering different aspects of the cloud development life-cycle.

All these projects confirm the growing interest for cloud environments not only in the academic field, but also in the industry. In fact, many existing issues have not been fully addressed by academic research, and new challenges keep emerging from industry applications. Automated service provisioning, virtual machine migration, server consolidation, energy management, traffic management and analysis, data security, etc. are cited as key features of cloud computing that also represent the major barriers to broader adoption [7, 28].

This paper is focused on the cloud capability to provide automated service provisioning (i.e. the ability of acquiring and releasing resources on demand) while dealing with big data applications. The goal of a cloud provider is to allocate and de-allocate resources from the cloud to satisfy the QoS while minimizing their operational costs. However, it is not obvious how a cloud provider can achieve this objective. In the context of big data applications it is even more difficult to determine the migration of services, as well as allocating and de-allocating resources from the infrastructure offered by the cloud. In fact, services are conceived as abstract specifications, typically defined and managed by third party organizations, aimed at modeling dynamic and complex business workflows [1].

*This work has been developed in the context of the Microsoft Azure Research Award for the project DESPACE (DEtecting and Solving Performance Antipatterns in Cloud Enviroments).

ICPE'15, Jan. 31–Feb. 4, 2015, Austin, Texas, USA.

http://dx.doi.org/10.1145/2668930.2695528.

In this context we propose to introduce Software Performance Antipatterns (SPA) [21] to drive the process of deploying big data applications on cloud-based environments. The rationale of using performance antipatterns is two-fold: on the one hand, a performance antipattern identifies a bad practice in the big data application that negatively affects the performance indices, thus to support the identification of performance flaws; on the other hand, a performance antipattern definition includes a solution description that lets the software architect devise refactoring actions, thus it aims to improve the system performance.

Goal of this paper is to discuss the challenges and opportunities of research in the area of using performance antipatterns in cloud computing environments. In particular, we propose a model-based framework (named *SPA-CloudMeter*) that makes use of software performance antipatterns to optimise the QoS of big data applications deployed on cloud environments. To this end, we focus on modelling, analysis, and feedback software performance engineering activities to highlight the current open issues of the domain and the expected benefits.

The paper is organised as follows. Section 2 presents related work. Section 3 discusses the research vision of our model-based framework that aims to estimate the benefit of using SPA in cloud computing environments. Finally Section 4 concludes the paper with remarks for future research.

2. RELATED WORK

In the last decades software architects have proposed and implemented several concepts and best practices to build highly scalable applications. However, due to ever-growing datasets, unpredictable traffic, and the demand for faster response times these concepts need to be adapted in the context of cloud computing. The business and technical benefits of cloud computing as well as the issues and challenges of architecting cloud-based systems have been formulated and discussed in [13, 26].

In literature a variety of solutions have been provided to the individual challenges, e.g. virtual machine migration, server consolidation, energy management, traffic management and analysis, etc. [28]. In this paper we focus on the challange of automated service provisioning that is not a new problem. Dynamic resource provisioning has been studied extensively in the past [25, 29, 9, 3, 12]. These approaches typically involve: (i) constructing a performance model that predicts the number of application instances required to handle the demand, in order to satisfy quality requirements; (ii) predicting future demand and determining resource requirements using the performance model. However, to the best of our knowledge, none of the existing approaches proposes the usage of performance antipatterns as support for the automated service provisioning. Several approaches have been recently introduced to specify and detect code smells and antipatterns [16, 10, 20, 27, 18]. They range from manual approaches, based on inspection techniques [22], to metric-based heuristics [14, 17], using rules and thresholds on various metrics [15] or Bayesian Belief Networks [11].

Our previous work in software performance antipatterns includes the following latest contributions: (i) in [4] we tackled the problem of providing a more formal representation by introducing first-order logic rules that express a set of system properties under which an antipattern occurs; (ii) in [24] we introduced a methodology to rank performance antipatterns and optimise the solution process; (iii) in [23] we explored the synergies in the process of combining performance antipatterns with bottleneck analysis; in [6] we introduced a model-driven approach to broaden the detection of software performance antipatterns at runtime.

3. SPA-CLOUDMETER

This section presents the model-based framework, named *SPA-CloudMeter*, we propose to introduce software performance antipatterns for improving the QoS of big data applications deployed on cloud environments.

Figure 1 schematically represents the operational steps of our *SPA-CloudMeter* framework: in the *modelling* phase, an application model is built to design the software and hardware artifacts for the big data application under study; in the *analysis* phase, a QoS model is built to monitor the software and hardware cloud resources employed by the big data application, and such model is solved to obtain QoS results of interest; in the *feedback* phase, the QoS results are interpreted and, if necessary, antipattern-based refactoring actions are devised with the goal to improve (from a performance perspective) the application under study.

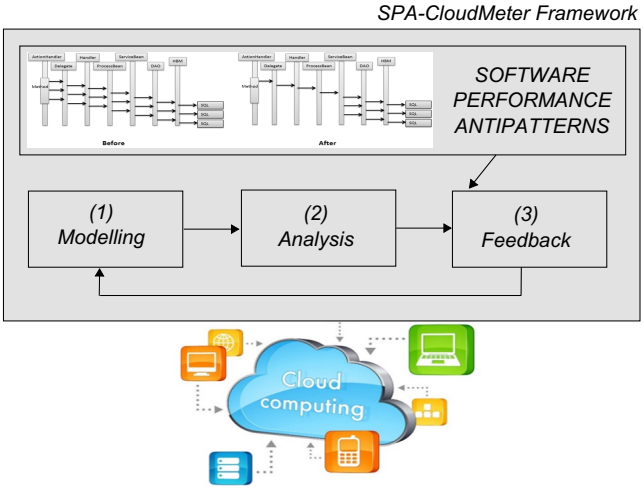

Figure 1: SPA-CloudMeter framework.

A preliminary step consists in the specification of cloud-related antipatterns. In fact, big data applications deployed on cloud environments include new performance related challenges, and practitioners continuously highlight more advanced pattern problems, e.g. for *Hadoop*[1] and *Cassandra*[2]. We are investigating the problems that have an analogy with the high-level specifications of the performance antipatterns we considered up to now [21]. For example, we found that some practitioners found that Hadoop map/reduce is not efficient for data locality, i.e. the more data nodes and data implies the less locality, especially larger clusters tend not to be complete homogeneous and data distribution and placement is not optimal. This latter problem is very similar to the *Circuitous treasure hunt* antipattern [21] that basically refer to software applications retrieving data in a not efficient way, i.e., such applications retrieve data from a first location, use those results to search in a second location, and so on until the ultimate results are obtained.

[1]http://hadoop.apache.org/
[2]http://cassandra.apache.org/

The first work-in-progress activity is the specification of the performance antipatterns, in the context of big data applications deployed on cloud environments, we are able to handle. Inspired by the DECOR method [15], we identified the following operational steps to specify antipatterns in the cloud computing context: (i) *Domain Analysis*: key concepts are identified in the text-based descriptions of reports provided by experienced big data technologists in their summits. They form a unified vocabulary of reusable concepts to describe bad practices and their solution; (ii) *Specification*: the concepts, which constitute a vocabulary, are combined to systematically specify performance antipatterns; (iii) *Processing*: the specifications are translated into operational ones that can be directly applied for the detection.

The second work-in-progress activity is the specification of QoS properties, in the context of big data applications deployed on cloud environments, we are able to analyse. In [2] we presented a graph of relationships highlighting the dependencies among some QoS attributes. In the cloud computing context we started focusing on performance and security that are related by a *trade-off* relationship, hence we aim to quantify the performance degradation incurred to achieve certain security requirements. From our previous work [5, 19] we experimented that the values of indices coming from the solution of the performance model (i.e. the one that includes security aspects) can be compared to the ones obtained for the same model (i) without security solutions, (ii) with different security mechanisms and (iii) with different implementations of the same security mechanism. Such comparisons help software designers to decide whether it is feasible to introduce/modify/remove security strategies on the basis of the stated performance requirements.

Modelling. SPA-CloudMeter allows to model big data applications deployed on cloud computing environments by specifying their functionalities, i.e. software and hardware services and their provisioning. It is necessary to model what are the application's software and hardware resources (e.g. software components, active virtual machines, hypervisors, etc.) and their expected resource demand or consumption. Key features of this domain are: for big data applications the volume, the variability and the complexity of data software services need to manage; for cloud computing environments the dynamic behaviour of hardware services that have the ability to scale workload peaks. In case of dynamic deployment of software services it is necessary to explicitly model elastic methodologies for hardware services while avoiding premature resource release. Finally, it is fundamental to model the security properties of software and hardware services as well to devise strategies suitable to protect them against not authorised accesses.

Analysis. SPA-CloudMeter allows to transform the big data application and the cloud computing environment model along with their security settings into a performance model. The performance indices [8] we expect to calculate are: the system response time, the throughput of software resources, and the utilization of hardware resources. All these indices contribute to quantify the QoS of the modelled big data application deployed on the cloud computing environment. QoS analysis results have to be interpreted in order to detect, if any, performance problems. Once performance problems have been detected (with a certain accuracy) somewhere in the application model, solutions have to be applied to remove those problems. A performance flaw originates from a set of unfulfilled requirement(s), such as the estimated average response time of a software service is higher than the required one. In case of unfulfilled requirement(s), our framework makes use of software performance antipatterns as reference knowledge to capture the well-known bad design practices that lead to software products suffering by poor performance.

Feedback. SPA-CloudMeter allows to detect and solve software performance antipatterns. In particular, antipattern-based rules interrogate the model elements to look for occurrences of the corresponding antipattern, whereas antipattern-based refactoring actions can be applied on the model elements with the final goal to improve (from a performance perspective) the application under analysis. The feedback operational step takes as input an application model (AM) and a set of performance results (PR), and it is constituted by two main operational steps. First, the detection of performance antipatterns is performed on the AM application model by running the antipatterns operational specifications, and it returns the detected antipattern instances with the list of suspicious model elements involved in them. Second, the solution of performance antipatterns is performed on the AM application model by using the antipattern-based refactoring actions that are a set of design alternatives suggested by the detected antipatterns. This step returns a set of refactored AM application models (AM'_1, ..., AM'_n) where the detected antipatterns have been removed, and each of these models undergo the same process of the initial model hence their analyses lead to a corresponding set of performance results (PR'_1, ..., PR'_n).

Note that the process of solving performance antipatterns includes further issues that may hurt the application under study. For example, a certain number of antipatterns cannot be unambiguously applied due to incoherencies among their solutions. It may happen that the solution of one antipattern suggests to split a software resource (with a high volume of data) into three finer grain resources, while another antipattern at the same time suggests to merge the original resource with another one (with a low volume of data). These two actions obviously contradict each other, although no pre-existing requirement limits their application. Even in cases of no explicit conflict between antipattern solutions, coherency problems can be raised from the order of application of solutions. In fact the result of the sequential application of two (or more) antipattern solutions is not guaranteed to be invariant with respect to the application order. Criteria must be introduced to drive the application order of solutions in these cases. Furthermore, antipattern-based refactoring actions do not a priori guarantee performance improvements, because the entire process is based on heuristic evaluations.

Summarizing our SPA-CloudMeter framework provides the following contributions: (i) specifying software performance antipatterns for cloud computing environments; (ii) modelling the activity flow in the specification of big data applications deployed on cloud environments; (iii) defining metrics and indices to evaluate the QoS of such applications; (iv) devising feedback strategies to optimise software and hardware services. SPA-CloudMeter currently considers only performance and security goals of big data applications deployed on cloud computing environments, however it can be extended to other quality criteria such as reliability, availability, etc., thus to support trade-off decisions.

4. CONCLUSION

In this paper we presented the research vision of a model-based framework that makes use of software performance antipatterns to optimise the quality of big data applications deployed on cloud environments. Modelling, analysis, and feedback activities have been discussed to highlight the current open issues of the domain and the expected benefits. We showed that both big data applications and cloud computing environments offer very promising challenges for research. As future work it is necessary to implement the SPA-CloudMeter framework for the performance assessment of real-world systems, thus to estimate its effectiveness.

5. REFERENCES

[1] A. Barker, C. D. Walton, and D. Robertson. Choreographing web services. *IEEE T. Services Computing*, 2(2):152–166, 2009.

[2] S. Becker, L. Happe, R. Mirandola, and C. Trubiani. Towards a methodology driven by relationships of quality attributes for qos-based analysis. In *ICPE*, pages 311–314, 2013.

[3] P. Bodík, R. Griffith, C. Sutton, A. Fox, M. Jordan, and D. Patterson. Statistical machine learning makes automatic control practical for internet datacenters. In *HotCloud*, 2009.

[4] V. Cortellessa, A. Di Marco, and C. Trubiani. An approach for modeling and detecting software performance antipatterns based on first-order logics. *Software and System Modeling*, 13(1):391–432, 2014.

[5] V. Cortellessa and C. Trubiani. Towards a library of composable models to estimate the performance of security solutions. In *WOSP*, pages 145–156, 2008.

[6] A. Di Marco and C. Trubiani. A model-driven approach to broaden the detection of software performance antipatterns at runtime. In *International Workshop FESCA*, pages 77–92, 2014.

[7] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel. The cost of a cloud: Research problems in data center networks. *Computer Communications Review*, 2009.

[8] R. Jain. The Art of Computer Systems Performance Analysis. *SIGMETRICS Performance Evaluation Review*, 19(2):5–11, 1991.

[9] E. Kalyvianaki, T. Charalambous, and S. Hand. Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters. In *ICAC*, pages 117–126, 2009.

[10] F. Khomh, M. D. Penta, Y.-G. Guéhéneuc, and G. Antoniol. An exploratory study of the impact of antipatterns on class change- and fault-proneness. *Empirical Software Engineering*, 17(3):243–275, 2012.

[11] F. Khomh, S. Vaucher, Y.-G. Guéhéneuc, and H. A. Sahraoui. Bdtex: A gqm-based bayesian approach for the detection of antipatterns. *Journal of Systems and Software*, 84(4):559–572, 2011.

[12] J. Kirschnick, J. Alcaraz Calero, L. Wilcock, and N. Edwards. Toward an architecture for the automated provisioning of cloud services. *Communications Magazine, IEEE*, 48(12):124–131, 2010.

[13] H. C. Lim, S. Babu, J. S. Chase, and S. S. Parekh. Automated control in cloud computing: challenges and opportunities. In *Workshop on Automated control for datacenters and clouds (ACDC)*. ACM, 2009.

[14] R. Marinescu. Detection strategies: Metrics-based rules for detecting design flaws. In *ICSM*, pages 350–359, 2004.

[15] N. Moha, Y.-G. Guéhéneuc, L. Duchien, and A.-F. L. Meur. Decor: A method for the specification and detection of code and design smells. *IEEE Trans. Software Eng.*, 36(1):20–36, 2010.

[16] N. Moha, F. Palma, M. Nayrolles, B. J. Conseil, Y.-G. Guéhéneuc, B. Baudry, and J.-M. Jézéquel. Specification and detection of soa antipatterns. In *ICSOC*, pages 1–16, 2012.

[17] R. Oliveto, F. Khomh, G. Antoniol, and Y.-G. Guéhéneuc. Numerical signatures of antipatterns: An approach based on b-splines. In *European Conference on Software Maintenance and Reengineering (CSMR)*, pages 248–251, 2010.

[18] R. Peters and A. Zaidman. Evaluating the lifespan of code smells using software repository mining. In *European Conference on Software Maintenance and Reengineering (CSMR)*, pages 411–416, 2012.

[19] R. J. Rodríguez, C. Trubiani, and J. Merseguer. Fault-Tolerant Techniques and Security Mechanisms for Model-based Performance Prediction of Critical Systems. In *ISARCS*, 2012.

[20] D. Romano, P. Raila, M. Pinzger, and F. Khomh. Analyzing the impact of antipatterns on change-proneness using fine-grained source code changes. In *Working Conference on Reverse Engineering (WCRE)*, pages 437–446, 2012.

[21] C. U. Smith and L. G. Williams. More New Software Performance Antipatterns: Even More Ways to Shoot Yourself in the Foot. In *Computer Measurement Group Conference*, pages 717–725, 2003.

[22] G. Travassos, F. Shull, M. Fredericks, and V. R. Basili. Detecting defects in object-oriented designs: using reading techniques to increase software quality. In *ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 47–56, 1999.

[23] C. Trubiani, A. Di Marco, V. Cortellessa, N. Mani, and D. C. Petriu. Exploring synergies between bottleneck analysis and performance antipatterns. In *ICPE*, pages 75–86, 2014.

[24] C. Trubiani, A. Koziolek, V. Cortellessa, and R. Reussner. Guilt-based handling of software performance antipatterns in palladio architectural models. *Journal of Systems and Software*, 95:141–165, 2014.

[25] B. Urgaonkar and A. Chandra. Dynamic provisioning of multi-tier internet applications. In *ICAC*, pages 217–228. IEEE Computer Society, 2005.

[26] J. Varia. Amazon Web Services - Architecting for the Cloud: Best Practices, May 2010.

[27] A. F. Yamashita and L. Moonen. Do code smells reflect important maintainability aspects? In *ICSM*, pages 306–315, 2012.

[28] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *J. Internet Services and Applications*, 1(1):7–18, 2010.

[29] Q. Zhang, L. Cherkasova, and E. Smirni. A regression-based analytic model for dynamic resource provisioning of multi-tier applications. In *ICAC*, 2007.