A Model-Driven DevOps framework for QoS-aware Cloud applications

Michele Guerriero*, Michele Ciavotta*, Giovanni Paolo Gibilisco *, Danilo Ardagna *
*Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Milan, Italy
{michele.guerriero, michele.ciavotta, giovannipaolo.gibilisco, danilo.ardagna}@polimi.it

Abstract

Recently we witnessed a deep transformation in the the design, development and management of modern applications, which have grown in scope and size becoming distributed and service-oriented. A big part in this metamorphosis is played by the Cloud with the availability of almost-infinite resources, high availability and outsourced maintenance. This has led to the emergence of new software development methodologies to effectively deal with this paradigm shift in the field of software engineering. DevOps is one of them, it advocates for a greater level of collaboration and convergence between developers and other IT professionals. Consequently, new tools, purposely designed to ease this process, are required. In this scenario, we present SPACE4Cloud, a DevOps integrated environment for model-driven design-time quality of service assessment and optimization, and runtime capacity allocation of Cloud applications.

Index Terms

Auto-Scaling; Capacity Allocation; Optimization; QoS

I. Introduction

In recent years we have witnessed a paradigm shift in the process of development and management of software systems. Projects have progressively grown in size and scope. The legacy model of standalone applications has lost much of its relevance in favor of more flexible, self-managed, distributed and web-based architectures in which much of the work is done server-side leaving on clients machines only thin clients (*e.g.* web browsers). The concept of software version has faded out and the life-cycle management of a product, along with its quality assurance, has become a continuous process. Rapidity and agility, defined as the ability of a system to adapt to rapid changes in the requirements in productive and cost-effective ways [1] are now fundamentals keys to success.

Another factor to consider in the analysis of the change we are experiencing is the emergence and the success of Cloud computing. As a matter of fact, the possibility to have access to nearly infinite resources on pay-per-use basis and the outsourcing of maintenance operations to the Cloud provider deeply impacted the process of design and implementation as well as integration, deployment, and management of software systems. Cloud computing has meant for developers having a set of new tools and abstractions that simplify the development and operation processes. Dynamic systems capable to react to workload fluctuations by adapting themselves in order to keep the performance unchanged can be easily built and run, while delegating to the Cloud provider the intensive tasks of infrastructure management and maintenance.

Overall, new ideas for application development and management have appeared, which resulted in the DevOps methodology [2], *i.e.*, a software development method grounded on collaboration, more often on a real convergence, among software developers, system administrators, and performance engineers. Specifically, DevOps is a t software development and operation paradigm that fosters the interwoven cooperation, collaboration and communication (in a word convergence) of project stakeholders, in particular the teams devoted to software development and those devoted to operations. The fundamental axiom in this software operational paradigm is that developers and other IT teams must work jointly to realize high-quality products [3] [4]. In summary, DevOps is a movement that helps to bridge the cultural gap between development and operations extending Agile practices to operations.

Our work aims to bridge the gap between the processes of design-time performance evaluation and the runtime self-adaptation of a Cloud application. In fact, in the early phases of development only rough estimates of resource demands are available and exploiting such information a user can use the design-time module of our solution to attain an early deployment configuration, a set of adaptation policy (enforced by the runtime module) and a prediction of execution costs and performance. Later on, while application versions with different degree of maturity succeed, from tests performed on production-like environment new and more accurate values can be gathered and used to improve the design-time models. This, in turn, can be used to discover possible bottlenecks and bugs and to improve the runtime adaptation actions. Ultimately, this cyclic process should results in a more stable, cost-effective, dynamic, and QoS-aware solution. Despite many are the works presented in literature focusing on the design time or on the runtime part of this process, we found a lack of existing solutions that integrate both design-time and runtime quality assurance techniques.

In this context we propose SPACE4Cloud (System PerformAnce and Cost Evaluation on Cloud), an integrated environment for model-driven design-time Quality of Service (QoS) assessment, optimization and QoS-aware runtime capacity allocation of Cloud applications. It comprises two main tools: **SPACE4Cloud**^{Dev} and **SPACE4Cloud**^{Ops} At design-time SPACE4Cloud^{Dev} takes in input models in extended PCM format [5] describing the application under development in terms of functionalities, QoS requirements, and a user-defined workload specified over a 24-hour time horizon. The tool is able to perform, through a local-search-based metaheuristic, a fully automatized exploration of the space of possible Cloud deployments, seeking for the configuration that minimizes the execution costs fulfilling at once the QoS requirements. The outcome of this module is a new set of models describing the Cloud deployment and a set of runtime adaptation actions to be enacted at the beginning of each hour of the day. Such second set of models is eventually fed into SPACE4Cloud^{Ops}, which is the module in charge of guaranteeing at runtime the pledged QoS levels by identifying and enacting suitable adaptation policies, over a finer grained timescale, within a open-loop control approach. This solution implements the so-called Receding Horizon Control paradigm. It optimizes resources allocation over a finite time window at a finer grained scale (e.g., 5-10 minutes); this process generates a set of scaling actions to be enforced in order to obtain a satisfactory QoS during the time window, yet only the adaptations for the first time step are executed. The process is then repeated considering the subsequent time step as the beginning of a new time window.

The SPACE4Cloud environment is implemented in Java within the framework of MODAClouds[1] EU FP7 project and released under Apache License 2.0[2,3].

The reminder of the paper is organized as follows: in Section II we introduce SPACE4Cloud and detail its components; the overall MODAclouds environment in which our solution is executed is presented in Section III. Some preliminary experimental results on an application case study are presented in Section IV. Related works are analyzed in Section V. Conclusions are finally drawn in Section VI.

II. SPACE4Cloud: a model-driven QoS-aware DevOps environment

The aim of this section is to present the SPACE4Cloud environment, detailing the most important elements of its components (namely, SPACE4Cloud^{Dev} and SPACE4Cloud^{Ops} and outlining the importance of a model-driven approach in the DevOps ecosystem devised within the MODAClouds project.

A. Design-time: SPACE4Cloud^{Dev}

SPACE4Cloud^{Dev} is the design-time component of SPACE4Cloud. It offers two main functionality developed with the goal of ease the design of Cloud applications by assessing the impact of deployment decisions in the early design stages.

SPACE4Cloud^{Dev} can be used by the application development team to evaluate the expected QoS and cost of a design decision and assess different application design alternatives. The more advanced feature offered by SPACE4Cloud^{Dev} aims at automatizing the exploration of the space of possible Cloud configurations, seeking for the one that minimizes the execution costs fulfilling at once QoS and architectural constraints.

The performance analysis implemented by SPACE4Cloud^{Dev} is focused on providing insights on the behavior of the application as seen by the user, as well as information about the utilization of the cloud infrastructure. From the application point of view, SPACE4Cloud^{Dev} is capable of deriving averages and percentiles of response times for both exposed and internals functionality. From the infrastructure point of view, SPACE4Cloud^{Dev} can provide application developers with an estimation of the utilization for all the computational resources used to deploy the application. This information is particularly useful to identify potential bottlenecks within the system.

At the core of SPACE4Cloud^{Dev} lies a model-based approach to application development: it uses an extended version of PCM models [5], which allows to model many characteristics such as the use of Cloud services to host application components, a profile for the incoming workload that spans on a 24-hour period, a set of constraints related to QoS and to some structural characteristics of the application (e.g., minimum replication factor for key components).

The features offered by SPACE4Cloud^{Dev} enable the development team to adopt a model based design approach in which a set of models describing the application under development are transformed in automatic or semi-automatic ways and analyzed in order to predict the expected application behavior in terms of costs and QoS. All the model-to-model transformations used by SPACE4Cloud^{Dev} have been fully automatized and rely on state of the art approaches for the generation of performance models from design model [6]. This automation allows software architects to exploit the benefit of performance predictions, without the need to know details of the advanced performance models used in the analysis.

Performance analysis is based on Layered Queuing Networks (LQNs) [7]. State of the art solution engines such as LINE [8] or LQNS [9] can be used to evaluate the performance model and derive the QoS metrics. The designed solution is also evaluated in term of costs in order to provide an estimation of the daily cost involved in using the proposed cloud infrastructure.

[1] www.modaclouds.eu
[2] github.com/deib-polimi/modaclouds-space4cloud
[3] github.com/deib-polimi/modaclouds-autoscalingReasoner

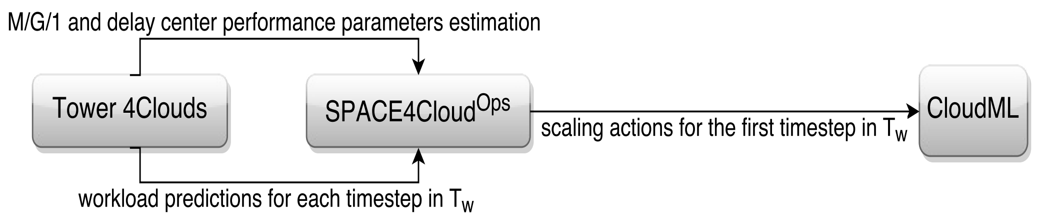
M/G/1 and delay center performance parameters estimation
Tower 4Clouds
SPACE4Cloud^Ops
scaling actions for the first timestep in T_W
CloudML
workload predictions for each timestep in T_W

Fig. 1: SPACE4Cloud^{Ops} block diagram

Finally, SPACE4Cloud^{Dev} enables application developers to automatically explore a wide range of deployment decisions delegating the generation of deployment alternatives and their evaluation to the tool.

To generate promising candidate deployment alternatives SPACE4Cloud^{Dev} uses an advanced heuristic approach. Then it exploits its assessment functionality to evaluate the performance of the generated alternatives. The user can drive the optimization by providing constraints that the final solution has to fulfill. Constraints can predicate on the structure of the solution specifying, for example, the set of cloud services to take into consideration or minimum requirements that services has to provide to host application components (e.g., the minimum memory capacity of a VM). Constraints can also specify the expected QoS of the application in terms of average and percentile of the response time for any of the modeled functionality of the system. The optimization procedure will then discard all the solutions that do not meet these requirements and provide to the user a feasible solution that minimize the cost of the deployment. More details on the optimization approach used by SPACE4Cloud^{Dev} can be found in [10].

The deployment solution derived by the optimization process can be exported back in the initial model in order to be explored or changed by the developer and eventually deployed and managed at runtime. A set of performance metrics derived by the analysis of the system is also carried over at runtime and used by SPACE4Cloud^{Ops}.

B. Runtime: SPACE4Cloud^{Ops}

SPACE4Cloud^{Ops} continues the process of quality assurance at runtime, i.e., during operations. Once the initial optimal solution, derived by SPACE4Cloud^{Dev}, has been deployed into the runtime platform, SPACE4Cloud^{Ops} starts adapting the underlying infrastructure, with the aim to fulfill application SLAs.

The adaptation mechanism is based on the solution of a mathematical problem that determine the optimimal resource allocation.

SPACE4Cloud^{Ops} enacts a receding horizon control [11] over a 5-10 minutes time scale. Periodically, given as input a prediction of the incoming workload for the next n_w time slots belonging to a time window $\mathcal{T}_w$, the optimal solution (in terms of number of VMs to run) for each timestep in $\mathcal{T}_w$ is determined; then only the decisions for the first time step are enacted within the running system.

At runtime, as a first approximation, each application hosted in a VM is modelled with a simplified performance model, which includes an M/G/1 queue in tandem with a delay center. The delay center is used to model network and/or protocol delays introduced in establishing connections, etc. Performance parameters need to be continuously updated at runtime (see [12] for further details) in order to capture transient behavior of VMs network and I/O interference [13], and performance variability of the Cloud provider over time [14]. This simplified performance model is used since alternative runtime adaptation decisions need to be evaluated very quickly according to a 5-10 minutes time scale.

Finally, since we deal with a finer grained time scale, we allocate the cost of an instance as soon as it is required and we consider a VM as freely available until the end of the hour, following the common pay by the hour pricing model. At the beginning of the following hour, if we do not need this instance again, it will be released; otherwise, it will remain available and we will be charged again. Overall our approach is formulated as a Mixed Integer Linear Problem (MILP), which can be efficiently solved by commercial solvers. Details about our problem formulation can be found in [15].

SPACE4Cloud^{Ops} employs this optimization model in order to control the execution infrastructure by means of scaling actions. At each time step (marked by a clock element) an appropriate monitoring system provides new workload predictions for each time slot in the current time window $\mathcal{T}_w$ and new estimates for the performance parameters (see Figure 1). Indeed, these parameters have to be continuously updated at runtime since they catch and describe transient behavior of Cloud environments. The last is not an trivial requirement, since the parameters to update may not be directly measurable, like the workload predictions, or at least very difficult to measure, like the resource demands and it may be preferable to adopt statistical methods in order to derive them. This discussion underlines the need of a specifically designed monitoring system, which will be further discussed in the next Section. The optimizer component feeds the optimization model using the monitored parameters and the current application state expressed in terms of allocated VMs that are available for free for each future time step in the considered time window.

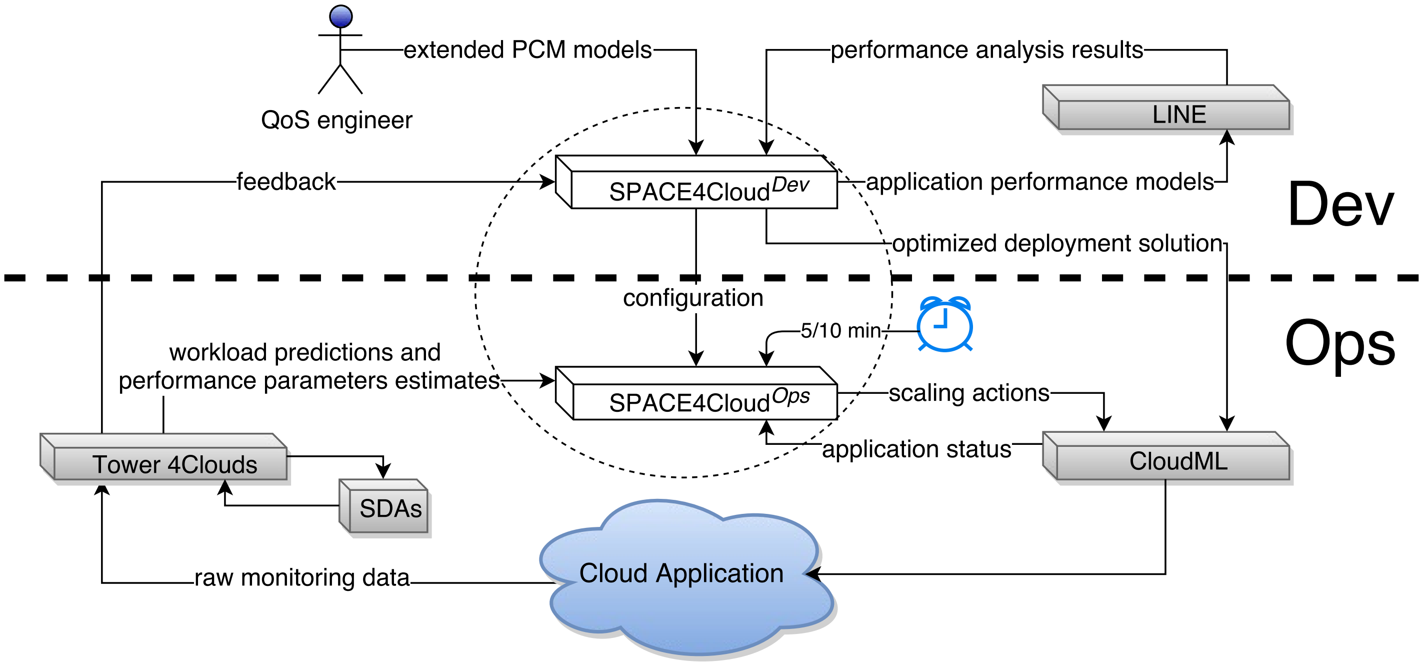

Fig. 2: SPACE4Cloud working environment

In the next Section we are going to describe in more detail the required environment in order to employ the SPACE4Cloud framework, focusing on each component with which it interacts.

III. MODACLOUDS DEVOPS PLATFORM

In previous sections we provided some hints about the operating environment required by SPACE4Cloud, in terms of specific components with which it needs to interact. In particular SPACE4Cloud^{Dev} requires an LQN solver performing a performance analysis of the application, while SPACE4Cloud^{Ops} requires a monitoring system able to provide all the necessary metrics and an IaaS interface enabling the provisioning of new Cloud resources and providing access to runtime deployment information. In this section we briefly introduce our choices for each of these components. All of them have been developed within the MODAClouds project and are part of the MODAClouds Platform. Figure 2 shows the whole architecture surrounding the SPACE4Cloud DevOps environment.

LINE (Layered QueueIng Network Engine) [8], an open-source tool for the performance analysis of Cloud applications, is the adopted LQN solver. LINE is able to compute both average performance metrics and response time distributions, which can be used to assess percentile Service-Level Objectives. The integration of LINE with SPACE4Cloud^{Dev} allows to bring the low level analysis of LQN model to the higher level design model.

Regarding the monitoring system, we employ *Tower 4Clouds* [16], which is specifically designed to support SPACE4Cloud^{Ops}, providing all the metrics for keeping the optimization model up to date with the respect to the runtime conditions. *Tower 4Clouds* is basically constituted by a set of distributed data collectors and a central server. Data collectors are responsible for directly sense metrics coming from applications or infrastructural resources and to submit them to the central server, which then takes care of processing these raw data. On the server side we can find a stream reasoning engine based on C-SPARQL language (Continous-SPARQL [17]), which allows to query the incoming and flowing stream of metrics formatted in RDF.

It is possible to install queries on the available streams of metrics by means of monitoring rules, describing how incoming data have to be processed, what conditions have to be verified, and what output should be produced. Components can be attached to consume the output of a monitoring rule by means of Metric Observers. In this way components can be automatically notified about rules evaluation results.

Tower 4Clouds offers a specific set of Data Collectors that successfully match the requirements of SPACE4Cloud^{Ops}; these are named Statistical Data Analyzers (SDAs). SDAs extract hidden information from monitoring data using statistical and predictive methods. In particular, SPACE4Cloud^{Ops} leverages **Estimation** and **Forecasting** SDAs. Estimation SDAs employee multiple estimation methods, like UBR (Utilization-based regression), CI (Complete Information) [16] and ERPS (Extended Regression for Processor Sharing resources) [18], allowing to obtain estimates for the performance parameters that SPACE4Cloud^{Ops} has to keep up to date in the optimization model, like the services demands and delays. Forecasting SDAs implement a number of forecasting methods including both time series forecasting and machine learning based forecasting algorithms. For the time series forecasting, included methods are Autoregressive model (AR) and Autoregressive integrated moving average model (ARIMA) [16]. In particular, in our context we adopt the ARIMA model for the workload forecasting.

For what concern the IaaS interface, we adopted *CloudML* [19], another component developed in the context of the MODAClouds project. CloudML provides a domain specific modelling language along with a runtime environment in order to allow from one side to model the provisioning and deployment of a multi-Cloud application, from the other to automate the deployment and to facilitate the runtime management, in terms of adaptation or reconfiguration actions of multi-Cloud applications. In this way, CloudML supports the DevOps methodology, achieving better delivery life-cycle by integrating in a single framework both development and operation activities. CloudML is inspired by component-based approaches that facilitate separation of concerns and reusability. In this respect, deployment models can be regarded as assemblies of components exposing ports (or interfaces), and bindings between these ports. The runtime environment is accessible as a service through a WebSocket interface.

IV. Experimental Results

In this section we describe some of the experiments performed to prove the soundness of our approach. While describing these experiments we will also walk through the workflow of the entire SPACE4Cloud environment in order to better underline how our approach fits the DevOps paradigm.

Our analysis is based on the Meeting in the Cloud (MiC) application. MiC is a web application that implements a social network, which allow users to create a profile by specifying, during the registration phase, his/her topics of interest and share them with other users. For each preferred topic, the user has to answer some appreciation questions with a grade from 1 to 5. Based on these grades the system calculates the Pearson coefficient in order to identify which other users are most similar to the newcomer. After the registration process, the user can log into the MiC application and interact with the suggested Best Contacts by writing and reading posts on selected topics. Users can also change their interests refining their profiles; in this case the system reacts by re-evaluating the similarity and updating the list of recommended contacts. Embracing a model-based approach, the first step is to model our application and the operational environment at design time exploiting the extended PCM format. To this aim, the MiC case study application has been partially modeled and a single tier deployment has been considered.

Moreover, just three type of requests have been modeled among those exposed by MiC, *register*, *selectTopics* and *saveAnswers*; the expected user workflow consists in performing these operations in sequence completing the registration process. Finally, we assume for simplicity the application to face significant workload variations only during the four central hours of the day, from 10 a.m. to 2 p.m. An increase/decrease of the workload intensity is assumed to happen every half an hour.

The resulting models need then to be correctly characterized from the performance point of view. To this aim, initial estimates of service times are derived by performing some preliminary experiments on a prototype environment and relying on state-of-the-art parameter estimation techniques [12].

Along this path we could properly describe the application at hand and the runtime environment and feed with the final models SPACE4Cloud^{Dev}. The output is an initial deployment solution, a set of hourly thresholds over the average requests response time and a set of adaptation policies over 24 hours, expressed in terms of number of VMs per hour to be instantiated.

At this point SPACE4Cloud^{Dev} automatically deploys the application in the MODAClouds runtime environment using *CloudML*, and sends the required configuration to SPACE4Cloud^{Ops}, which starts creating the optimization model and enacting the runtime control loop with a 5 minute timescale against a synthetic workload generated by Apache JMeter[4]. A timescale of 5 minutes has been set up since it has been proved to provide better performance [15].

[4]jmeter.apache.org

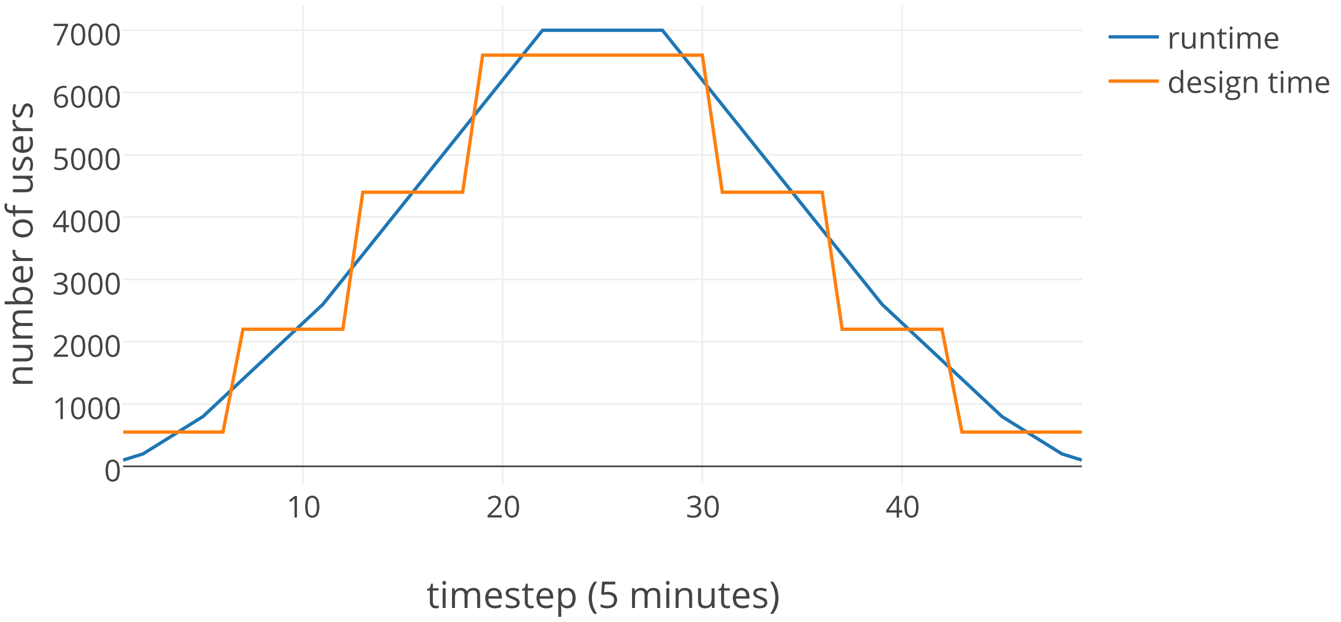

Fig. 3: Comparison between design time and runtime workloads.

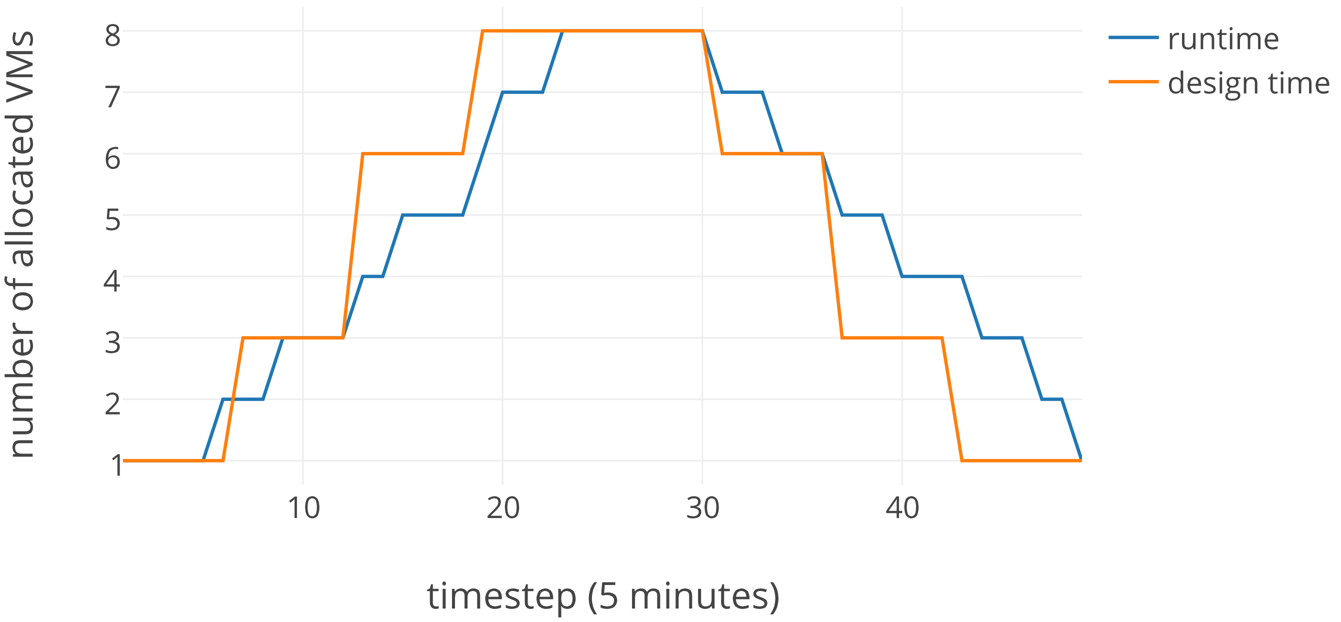

Fig. 4: Comparison between design time and runtime allocation plans

According to the design time assumptions, at runtime we injected a workload lasting for four hours but now varying every 5 minutes. The injected workload is basically a ramp up with a maximum number of users equal to 7,000, followed by a ramp down. The average number of users for each half an hour is such to be the same expected in the design time analysis. Figure 3 shows a comparison between the workload expected at design time and the one injected at runtime. We can notice that the design time expected workload varies every half an hour or every 6 timesteps lasting for 5 minutes, while the runtime worklaod shows an higher variability. SPACE4Cloud^{Ops} have to be able to capture this higher variability of the injected workload with the respect to the one expected at design time. Figure 4 shows a comparison between the allocation plan predicted at design time and the real allocation plan enacted at runtime. The number of running VMs varies in the range [1, 8] following the shape of the workload, proving the effectiveness of the autoscaling and adaptive mechanisms provided by SPACE4Cloud^{Ops}. It is worth to be noticed that the runtime allocation plan appears to be temporally shifted ahead with the respect to the design time plan, in which when a new VM is required it is assumed to become available instantly. This could be explained by the combination of two factors. During the ramp up phase a delay is introduced resulting by the sum of various delaying factors, i.e. the time required to have a new instance up and running or the time required by the load balancer to start splitting the workload considering also the new instantiated VM. On the other side, during the ramp down phase, if a VM instance result to be not required anymore, but it is still available for free until the end of the hour, it will be left running since we have already been already charged for it.

V. Related Work

In [20] we presented SPACE4Clouds for the first time as a design time tool for system performance and cost evaluation of Cloud systems. Here we propose SPACE4Clouds as a DevOps environment, in which what we presented in [20] becomes now SPACE4Cloud^{Dev}, or the Dev tool, which has been now extended with an additional module, in order to be coupled and connected to SPACE4Cloud^{Ops}, the runtime QoS management system, in a DevOps flavor.

In a DevOps approach, given the stress on collaboration among Dev ond Ops teams, the ability to predict and assess QoS characteristics of a complex system acquires paramount importance. An effective way of dealing with this necessity is to integrate prediction techniques for non-functional properties into the software development process via appropriate workflows and tools. In this approach, adopted in our solution, the model of the software architecture is annotated with non-functional elements that are used to predict the performance of the application. The output of this analysis can be used at design-time as feedback to modify the architecture and/or the deployment and at runtime to improve the application adaptability with the aim of meeting some desirable requirements.

In the Model-Driven Quality Prediction (MDQP) research area, the application is modeled by means of UML models; The Object Management Group (OMG) introduced for this purpose two UML profiles specially tailored to model QoS, called *Schedulability, Performance and Time* (SPT) [21] and *Modeling and Analysis of Real-Time and Embedded Systems* MARTE [22]. These profiles allow to express some performance characteristics but still lack the proper support to model the heterogeneity of the Cloud infrastructure. This approach of extending UML is not the only one that deals with the analysis of non-functional properties of software systems, Becker et al. developed the PCM [23], a language that can be used to model an application and its non-functional properties. From the model description a LQN model to estimate the performance of the system can be automatically derived. Many meta-models have been built to support performance prediction, some surveys of these models, their capability and their applicability to different scenarios can be found in [24], [25], [26].

To cope with the fact that the space of design alternatives can be very large, solutions able to help the user in this task have been proposed. In [27], Li et al. presented the Automated Quality-driven Optimization of Software Architecture (AQOSA) toolkit that implements some advanced evolutionary optimization algorithms for multi-objective problems. The toolkit integrates modeling technologies with performance evaluation tools in order to evaluate the goodness of generated solution according to a cost function. Aleti et al. proposed a similar approach in [28]; they presented the ArcheOpterix framework that exploits evolutionary algorithms to derive Pareto optimal component deployment decisions with respect to multiple quality criteria. A multi-objective optimization strategy on top of ArcheOpterix is implemented in [29] to find trade-off between reliability and energy consumption in embedded systems. PerOpteryx [30] use a similar approach to optimize software architectures modeled with the Palladio framework [23] according to performance, reliability and cost. Other approaches combine analytic optimization techniques with evolutionary algorithms to find Pareto optimal solutions, an example of this is presented in [31] with a particular focus on availability, performance and cost. The main limitations of these approaches is the fact that the use of a genetic algorithm requires the evaluation of a huge number of candidate solutions, or individuals. In many situations the evaluation of such a big set of solutions is not feasible in reasonable time so the optimization might require hours or even days. A tabu search (TS) heuristic has been used in [32] to derive component allocation under availability constraints in the context of embedded systems. Finally, Frey et al. [33] proposed a combined metaheuristic-simulation approach based on a genetic algorithm to derive deployment architecture and runtime reconfiguration rules while moving a legacy application to the Cloud environment.

As far as the Operations side is concerned, many Cloud providers offer purely reactive auto-scaling using threshold-based rules. The scaling decisions are triggered based on some performance metrics and predefined thresholds [34]. This approach has become rather popular due to its simplicity. Nonetheless, many studies concerning the resource allocation problem under cost and execution time constraints have been published. A recent review on auto-scaling techniques can be found in [35]. The work presented in [36] proposed a framework supporting dynamic capacity allocation to adapt to workload fluctuations. The model formulated represents a Cloud center with a $M/M/C/C$ queuing system, with different priority classes. A multi-dimensional resource allocation with SLA guarantees problem also at data-center level is presented in [37].

In [38] the VM placement problem is solved at multiple time scales through a hierarchical optimization framework. The paper [39] aims at optimizing the cost and the utilization of a set of applications through vertical auto-scaling mechanism, i.e., proposing a new set of instances to minimize cost and maximize utilization, or increase performance efficiency.

An online VM provisioning is proposed in [40]. The solution solves allocation problems with partial information, calculating allocation and revenues as the customers arrive at the system and place their requests.

Finally, in [41], a control method based on reinforcement learning (RL) is presented. This technique aims at automating the scaling process without leveraging any a priori knowledge on the running application.

VI. Conclusions

In this work we presented an integrated DevOps environment for design-time modeling and optimization, and runtime control of Cloud applications. The aim of the tool is to minimize the execution cost of Cloud applications providing QoS guarantees by design. The most distinguished characteristic of the Cloud, such as variable workload, congestion due to multi-tenancy and performance variability have been included in the modeling of the application considered in the QoS evaluation.

Future work will be devoted to extend the runtime adaptive actions to PaaS platforms and multi-Cloud applications, both already managed at design-time. Moreover, a feedback mechanism will be implemented in MODAClouds runtime environment to provide a better estimation of design-time parameters [42]. In this way the user, dealing with an application model more coherent with the reality, can further improve the deployment discovering and solving possible performance bottleneck. This addition will provide feebacks from the operations to the development phase bringing us a step closer to a full adoption of the DevOps methodology. Finally, the tool will be extended for design-time modeling and optimization of data intensive applications.

Acknowledgements

The research reported in this article is partially supported by the European Commission grant no. FP7-ICT-2011-8-318484 (MODAClouds).

References

[1] Dev2ops.org, "What is devops?." http://dev2ops.org/2010/02/what-is-devops/.
[2] M. Loukides, *What is DevOps?* " O'Reilly Media, Inc.", 2012.
[3] J. Roche, "Adopting devops practices in quality assurance," *Communications of the ACM*, vol. 56, no. 11, pp. 38–43, 2013.
[4] M. Httermann, *DevOps for developers*. Apress, 2012.
[5] D. Ardagna, M. Ciavotta, M. Miglierina, G. Gibilisco, G. Casale, J. Pérez, F. D'Andria, and R. S. González, "MODAClouds D5.2.2 - MODACloudML QoS abstractions and prediction models specification," 2014.
[6] H. Koziolek and R. Reussner, "A model transformation from the palladio component model to layered queueing networks," in *Performance Evaluation: Metrics, Models and Benchmarks* (S. Kounev, I. Gorton, and K. Sachs, eds.), vol. 5119 of *Lecture Notes in Computer Science*, pp. 58–78, Springer Berlin Heidelberg, 2008.
[7] J. A. Rolia and K. C. Sevcik, "The method of layers," *IEEE Trans. Softw. Eng.*, vol. 21, pp. 689–700, Aug. 1995.
[8] J. Perez and G. Casale, "Assessing sla compliance from palladio component models," in *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2013 15th International Symposium on*, pp. 409–416, Sept 2013.
[9] G. Franks, T. Al-Omari, M. Woodside, O. Das, and S. Derisavi, "Enhanced modeling and solution of layered queueing networks," *Software Engineering, IEEE Transactions on*, vol. 35, pp. 148–161, March 2009.
[10] D. Ardagna, G. Gibilisco, M. Ciavotta, and A. Lavrentev, "A multi-model optimization framework for the model driven design of cloud applications," in *Search-Based Software Engineering* (C. Le Goues and S. Yoo, eds.), vol. 8636 of *Lecture Notes in Computer Science*, pp. 61–76, Springer International Publishing, 2014.
[11] Wook Hyun Kwon, Soo H. Han, *Receding Horizon Predictive Control: Model Predictive Control for State Models.* Springer, 2005.
[12] L. Zhang, X. Meng, S. Meng, and J. Tan, "K-scope: Online performance tracking for dynamic cloud applications," in *ICAC 2013*.
[13] Y. Mei, L. Liu, X. Pu, S. Sivathanu, and X. Dong, "Performance analysis of network i/o workloads in virtualized data centers," *Services Computing, IEEE Transactions on*, vol. 6, no. 1, pp. 48–63, 2013.
[14] Alistair Croll, "Cloud performance from the end user perspective." http://www.bitcurrent.com/download/cloud-performance-from-the-end-user-perspective/.
[15] D. Ardagna, M. Ciavotta, and R. Lancellotti, "A receding horizon approach for the runtime management of iaas cloud systems," SYNASC 2014.
[16] G. Iuhasz, S. Panica, G. Casale, W. Wang, P. Jamshidi, D. Ardagna, M. Ciavotta, D. Whigham, N. Ferry, and R. S. González, "MODAClouds D6.4.2 - runtime environment final release," 2015.
[17] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus, "C-sparql: A continuous query language for rdf data streams," *International Journal of Semantic Computing*, vol. 04, no. 01, pp. 3–25, 2010.
[18] J. Perez, G. Casale, and S. Pacheco-Sanchez, "Estimating computational requirements in multi-threaded applications," *Software Engineering, IEEE Transactions on*, vol. 41, pp. 264–278, March 2015.
[19] N. Ferry, G. Brataas, A. Rossini, F. Chauvel, and A. Solberg, "Towards Bridging the Gap Between Scalability and Elasticity," in *CLOSER 2014: 4th International Conference on Cloud Computing and Services Science – Special Session on Multi-Clouds*, pp. 746–751, SCITEPRESS, 2014.
[20] D. Franceschelli, D. Ardagna, M. Ciavotta, and E. Di Nitto, "Space4cloud: A tool for system performance and costevaluation of cloud systems," in *Proceedings of the 2013 International Workshop on Multi-cloud Applications and Federated Clouds*, MultiCloud '13, (New York, NY, USA), pp. 27–34, ACM, 2013.
[21] OMG, *UML Profile for Schedulability, Performance, and Time Specification*, 2005.
[22] OMG, "A uml profile for marte: Modeling and analysis of real-time embedded systems," 2008.

[23] S. Becker, H. Koziolek, and R. Reussner, “The palladio component model for model-driven performance prediction,” *Journal of Systems and Software*, vol. 82, no. 1, pp. 3–22, 2009.

[24] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni, “Model-based performance prediction in software development: A survey,” *Soft. Eng., IEEE Trans.*, vol. 30, no. 5, pp. 295–310, 2004.

[25] H. Koziolek, “Performance evaluation of component-based software systems: A survey,” *Performance Evaluation*, vol. 67, no. 8, pp. 634–658, 2010.

[26] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya, “Software architecture optimization methods: A systematic literature review,” *Soft. Eng., IEEE Trans.*, vol. 39, no. 5, pp. 658 – 683, 2013.

[27] R. Li, R. Etemaadi, M. Emmerich, and M. Chaudron, “An evolutionary multiobjective optimization approach to component-based software architecture design,” in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pp. 432–439, June 2011.

[28] A. Aleti, S. Stefan Björnander, L. Grunske, and I. Meedeniya, “Archeopterix: An extendable tool for architecture optimization of aadl models,” in *MOMPES 2009*.

[29] I. Meedeniya, B. Buhnova, A. Aleti, and L. Grunske, “Architecture-driven reliability and energy optimization for complex embedded systems,” in *Research into Practice Reality and Gaps* (G. Heineman, J. Kofron, and F. Plasil, eds.), vol. 6093 of *Lecture Notes in Computer Science*, pp. 52–67, Springer Berlin Heidelberg, 2010.

[30] A. Koziolek, *Automated Improvement of Software Architecture Models for Performance and Other Quality Attributes*. PhD thesis, Germany, 2011.

[31] A. Koziolek, D. Ardagna, and R. Mirandola, “Hybrid multi-attribute qos optimization in component based software systems,” *Journal of Systems and Software*, vol. 86, no. 10, pp. 2542 – 2558, 2013.

[32] M. Ouzineb, M. Nourelfath, and M. Gendreau, “Tabu search for the redundancy allocation problem of homogenous series-parallel multi-state systems,” *Reliability Engineering & System Safety*, vol. 93, no. 8, pp. 1257–1272, 2008.

[33] S. Frey, F. Fittkau, and W. Hasselbring, “Search-based genetic optimization for deployment and reconfiguration of software in the cloud,” in *ICSE 2013*.

[34] “Amazon autoscaling service: https://aws.amazon.com/autoscaling/.”

[35] T. Lorido-Botran, J. Miguel-Alonso, and J. Lozano, “A review of auto-scaling techniques for elastic applications in cloud environments,” *Journal of Grid Computing*, vol. 12, no. 4, pp. 559–592, 2014.

[36] W. Ellens, M. Zivkovic, J. Akkerboom, R. Litjens, and H. van den Berg, “Performance of cloud computing centers with multiple priority classes,” in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pp. 245 –252, june 2012.

[37] H. Goudarzi and M. Pedram, “Multi-dimensional sla-based resource allocation for multi-tier cloud computing systems,” in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 324 –331, july 2011.

[38] B. Addis, D. Ardagna, B. Panicucci, M. S. Squillante, and L. Zhang, “A hierarchical approach for the resource management of very large cloud platforms,” *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 5, pp. 253–272, 2013.

[39] P. Kokkinos, T. Varvarigou, A. Kretsis, P. Soumplis, and E. Varvarigos, “Cost and utilization optimization of amazon ec2 instances,” in *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, pp. 518–525, June 2013.

[40] S. Zaman and D. Grosu, “An online mechanism for dynamic vm provisioning and allocation in clouds,” in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pp. 253–260, June 2012.

[41] P. Jamshidi, A. Sharifloo, C. Pahl, A. Metzger, and G. Estrada, “Self-Learning Cloud Controllers: Fuzzy Q-Learning for Knowledge Evolution,” *ArXiv e-prints*, July 2015.

[42] T.-F. Forti, G. Iuhasz, M. Neagul, G. Casale, J. F. Prez, and W. Wang, “MODAClouds D6.3.2 - Techniques for filling the gap between design and run-time Final version, year = 2015.”