2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations

Cloud Adaptation & Application (Re-)Distribution: Bridging the two Perspectives

Santiago Gómez Sáez, Vasilios Andrikopoulos
Institute of Architecture of Application Systems,
University of Stuttgart
Stuttgart, Germany
{gomez-saez, andrikopoulos}
@iaas.uni-stuttgart.de

Florian Wessling, Clarissa Cassales Marquezan
Software Systems Engineering,
paluno, University of Duisburg-Essen
Essen, Germany
{florian.wessling, clarissa.marquezan}
@paluno.uni-due.de

Abstract

Cloud developers have to make several decisions when running their application in a cloud environment that may lead to conflicting objectives, inefficient deployment, and inappropriate or not existing adaptation strategies. Proper decision-support tools and processes are therefore needed to make cloud developers aware of the issues that need to be considered when deploying and running applications in the Cloud. Current decision support tools for cloud developers do not provide a structured and organized process in which the cloud developers can systematically check their choices when planning the deployment, execution, and adaptation of applications in the Cloud. In this paper, we combine two previous works and introduce an approach for identifying the options for (re-)deploying application in cloud providers infrastructures and the possible strategies of adaptation that can be used by the deployed application at runtime. The key contribution is a support process that synthesizes the two approaches. We also describe a case study where our support process is applied and we indicate the alternatives for application (re-)deployment and adaptation.

1. Introduction

In the recent years the Cloud computing model has been largely adopted by industry and investigated by researchers. Even without considering the different types of cloud service offerings currently available, cloud application developers are faced with a large amount of options and decisions to be made. This includes addressing a multitude of issues such as the partial versus complete migration of the application to the Cloud model, various application deployment options, appropriate adaptation strategies at runtime, and selection among the same type service offered by different cloud providers. Conflicting objectives, inappropriate deployment decisions, and adaptation strategies that fail to change during runtime may result in sub-optimal application performance and unnecessary incurred costs. Appropriate decision support tools and processes are therefore needed to make cloud application developers, which are the main stakeholders of this work, aware of these issues.

Current decision support tools for cloud application developers focus on the migration of the application to the Cloud environment [1], estimation of the application load [2] or the costs when deploying the application [3], [4], among others. However, the proposed solutions do not provide a structured and organized process in which the cloud application developers can systematically check their choices when planning the deployment and execution of applications in the cloud. In previous works [5], [6], we drew attention to the impact and dependencies of such developer choices, but focused either on (re)deployment [5] or runtime adaptation strategies [6] for cloud applications.

In this paper, we combine the previous works and we introduce an approach for identifying the options for (re-)deploying applications in a cloud provider and the possible strategies of adaptation that can be used by the deployed application at runtime. The goal of this approach is to support the cloud application developers to become aware of the type of choices to be made when planning

978-1-4799-5467-4/14 $31.00 © 2014 IEEE
DOI 10.1109/EDOCW.2014.33

and executing applications using cloud offers at various service delivery models. This work considers that the cloud application developer has already decided to migrate the application to the cloud. The key contribution of this paper is a process that formalizes the proposed approach. In this paper, we also conduct a case study based on the Apache Olio application[1]. Using this example, we show how our process is applied, and the results of options to support the cloud application developer that can be generated by our process.

The remainder of the paper is organized as follows. Section 2 summarizes the key ideas from our previous work. Section 3 introduces our approach. In Section 4, we present the case study and discuss the use of our approach. The related work is summarized in Section 5. Finally, Section 6 describes our conclusions and future work.

2. Background

Optimal Application Distribution. The exponential growth of cloud service offerings in the last years has increased the number of alternatives for engineering and re-engineering applications to be partially or completely run in a cloud environment [1]. Towards ensuring portability and interoperability among different Cloud providers, initiatives like the TOSCA standard [7] or Cloud Blueprints [8] enable the application developer to enhance the application design by providing the means to model which cloud offering is used to host which parts of the application stack. The existence of multiple migration types and cloud offerings broadens the application design alternatives space that developers must take into account when utilizing one or multiple cloud services. However, such space can be narrowed by specifying non-functional aspects.

Therefore, in [5] we propose a technology-agnostic formal framework towards deriving the optimal distribution of the application across cloud offerings in a flexible manner. We propose to model the application topology as a typed topology graph model, which then can be partitioned into a graph model that depicts the application specific and non-application specific (and possibly reusable) components of the application stack. Application specific components are unique and specific for each application, e.g., the front- and back-end components of a Web application, while the application non-specific components can be, e.g., middleware components like an Apache Web server or cloud offerings. By analyzing the typed topology graph model, a set of *viable* topologies describing alternative application deployment scenarios can be inferred.

The inference of multiple viable topologies introduces a multi-dimensional problem related to evaluating and deciding among such alternatives. Cloud providers nowadays focus on targeting one dimension, and provide proprietary tools that calculate and analyze the monetary cost when using their offered services, such as the Amazon Simple Monthly Calculator[2], and provide configuration samples for different types of applications and resources demands. To quantitatively evaluate these multiple dimensions, in [5] we propose the usage of *utility functions* towards building a ranked set of alternative viable topologies and identifying the optimal application distribution focusing on non-functional aspects, e.g., monetary cost, performance, etc. Optimizing the application distribution towards balancing the performance-cost trade-off must be considered as a long-term collaborative task which focuses on the one hand on the evolutionary aspect of the application workload, and on the other hand ensures that the triggered resource adaptations comply with expected service objectives. Focusing on three-layered applications, in [9] we identified the need to partially or completely distribute the application layers among multiple cloud offerings to cope with application workloads fluctuations. A significant performance improvement of the application database layer was observed when migrating its data to IaaS or DBaaS solutions, showing the latter to have the most improved performance for different workload characteristics.

Identifying Adaptation Strategies. In general there are several entities in a cloud environment that can trigger and be influenced by adaptation actions. These adaptation actions may interfere due to direct or indirect relationships among entities. Indeed, runtime adaptation can be triggered by the cloud application itself as well as by the infrastructure. Considering the point of view of the cloud application, the actual adaptation action can

1. Apache Olio Application v2.0: http://incubator.apache.org/olio/index.data/Olio_Overview_long.pdf

2. Amazon Simple Monthly Calculator: http://calculator.s3.amazonaws.com/index.html

vary greatly depending on the envisioned application distribution. For instance, it is possible to use a pool of virtual machines between which a load balancer distributes the incoming traffic [10]. Another approach [11] establishes a framework that abstracts from the application logic and manages its deployment inside the reserved cloud resources. The specification of adaptation triggers and actions is also supported by commercial solutions such as Google Compute Engine, Windows Azure and Amazon EC2, which allow the creation of rules or policies for load balancing or auto-scaling features. For the infrastructure point of view, adaptation, such as redeploying and relocating virtual resources between physical resources can also be triggered when e.g., SLA QoS requirements are violated.

As discussed above, a cloud environment is a complex system made up of many different entities with different capabilities for changing and adapting. The specific adaptation capabilities of these entities can be exploited to support the developers on identifying adaptation strategies tailored to their application focus. This is only possible when adaptation actions and their interrelations are identified. This means that the range of options that can be used for engineering cloud application adaptations is much broader than just elasticity. In [6] a conceptual model was introduced to structure the dependencies among the cloud entities with adaptation capabilities. The goal of this model is to make developers aware of the possible adaptation options. The model consists of four layers characterizing the entities of the cloud environment: The *Physical Layer* represents the data centers and physical infrastructure. The entities associated with the virtual resources are part of the *Virtualization Layer*. The *Logical Application Architecture Layer* comprises the entities associated with the topology and structure of the application and are typically hosted inside the virtual resources. Finally, the entities related to the core application logic implementation are enclosed in the *Application Business Logic Layer*. In [6] we present a step towards the identification of potential adaptation options in the cloud environment.

3. Approach

In the following we discuss how the approaches discussed in the previous section can be synthesized into a support process, starting with its requirements.

3.1. Requirements

The functional and non-functional requirements presented in this section aim to provide support for analyzing the application functional and non-functional aspects, enable the dynamic (re-)distribution of application in the Cloud, and trigger the dynamic adaptation of Cloud resources utilized by such application.

Functional Requirements.

- FR_1 *Top-down and bottom-up application evaluation*: The process must support the analysis and evaluation of the application topology alternatives both based on previous knowledge and empirical analysis, and during the application production phase, by e.g., monitoring.
- FR_2 *Enrichment of the Topology Specification*: The process has to support the definition of application topologies in various formats such as TOSCA [7] or Blueprints [8] and must consider non-functional aspects specified as extensions of the previous formats.
- FR_3 *Management and Configuration*: Any tool supporting such a process must provide management and configuration capabilities for cloud services from different providers covering all cloud service and delivery models. Furthermore, it must provide the means to support the inference of multiple alternative viable topology instances which specify the usage of such services under the different deployment models.
- FR_4 *Support of Different Migration Types*: In order to (re-)distribute an application the process has to support all Cloud native and non Cloud native application migration types identified in [1]: replacement, partial migration, whole software stack migration, and cloudification.
- FR_5 *Characterization of Adaptation Strategies*: The process must support the characterization of different adaptation actions that can be used during runtime for a given application. These different types of adaptation are related to the distinct mechanisms for enforcing changes spread among the cloud layers.
- FR_6 *Independence from Architectural Paradigm*: Independence from the architecture paradigm the application to be (re-)distributed is based on, e.g., Service-Oriented Architecture [13] or three-layered architecture [14], must be supported.
- FR_7 *Support & Reaction to Application Resources Demand Evolution*: As the workload of an

application is subject to fluctuations over time, the process must support the identification of these fluctuations and react by automatically and dynamically adapting the cloud resources based on the resource adaptation constraints specified by the developer, or by proposing an alternative application distribution.

FR_8 *Support of Hardware, Software, and Application Characteristics*: The process must consider such characteristics for optimizing the overall application performance.

FR_9 *Creation of Workload Behavior Model*: The process has to support the derivation of application workloads with different behaviors and must provide fitting capabilities in order to create the workload behavior model, e.g., using probability and statistical techniques [15].

FR_{10} *Multi-dimensional Evaluation and Analysis*: different parameters must be taken into consideration for (re-)distributing an application. The process must be aware of the developer objectives, and provide the means to empirically target a multi-dimension analysis, e.g., focusing on cost, performance, consistency, etc.

FR_{11} *Application (Re-)Distribution & Adaptation Utility Calculation*: The process must provide the means to calculate and analyze the value perceived by the developer of (re-)distributing the application. Such calculations and analysis must take into consideration the objectives of two parties involved in the application distribution, which can be expressed as developer and cloud provider preferences.

Non-functional Requirements.

NFR_1 *Security*: (Re-)distribution and (re-)configuration of applications requires root access and administrative rights to the application. Any tool supporting the process should therefore enforce user-wide security policies and incorporate necessary authorization, authentication, integrity, and confidentiality mechanisms.

NFR_2 *Extensibility*: The methodology should be extensible, e.g., in order to incorporate further provisioning, deployment, and configuration approaches and technologies for re-distribution and runtime adaptation of the application.

NFR_3 *Reusability*: The application topology alternative set analysis, application behavior evolution observation, application (re-)distribution and adaptation mechanisms as well as underlying concepts should not be solution-specific and technology dependent.

3.2. Process

In this section we propose a process-based realization approach that fulfills the requirements identified in the previous section. As depicted in Figure 1, three main participants can be identified in our process: the *Application Developer*, the *Distribution Support System*, and the *Infrastructure Management System*. For the purpose of simplifying the textual description of the process, we describe in this section the sequence of mandatory tasks of each participant separately, and identify the interactions among them. In all participants' perspectives the existence of an explicit specification of a synchronized loop among them can be observed. Such loop is introduced in [9] as the *Collaborative Loop*, and it focuses on the performance evolution aspect of the application. In this work we aim to broaden the dimensional analysis of such collaboration by incorporating a multi-dimensional analysis from both cloud provider and consumer perspectives. Our goal is to achieve an optimal and adaptation-aware distribution of the application that takes into account the interests of both.

Application Developer. The *Application Developer* mandatory functionalities are the ones related to designing the application architecture and implementing its components, defining the application topology model, describing the expected runtime adaptability properties of the application, and selecting the application distribution alternatives offered by a distribution support system, as discussed in [5]. Optional but recommended functionalities entail the specification of application non-functional aspects, e.g., expected performance and monetary cost, workload behavior and variation (if available), required deployment regions, etc.

The *Model & Enrich Application Topology* task consists of modeling and specifying the α-topology described in [5]. Such topology may be enriched with non-functional aspects information which may be specific to each application profile. The *Adaptability Properties Specification* task aims to explicitly make the application developer aware of automatic resources adaptation supported by the cloud offering. For example, the Amazon AWS Elastic Beanstalk

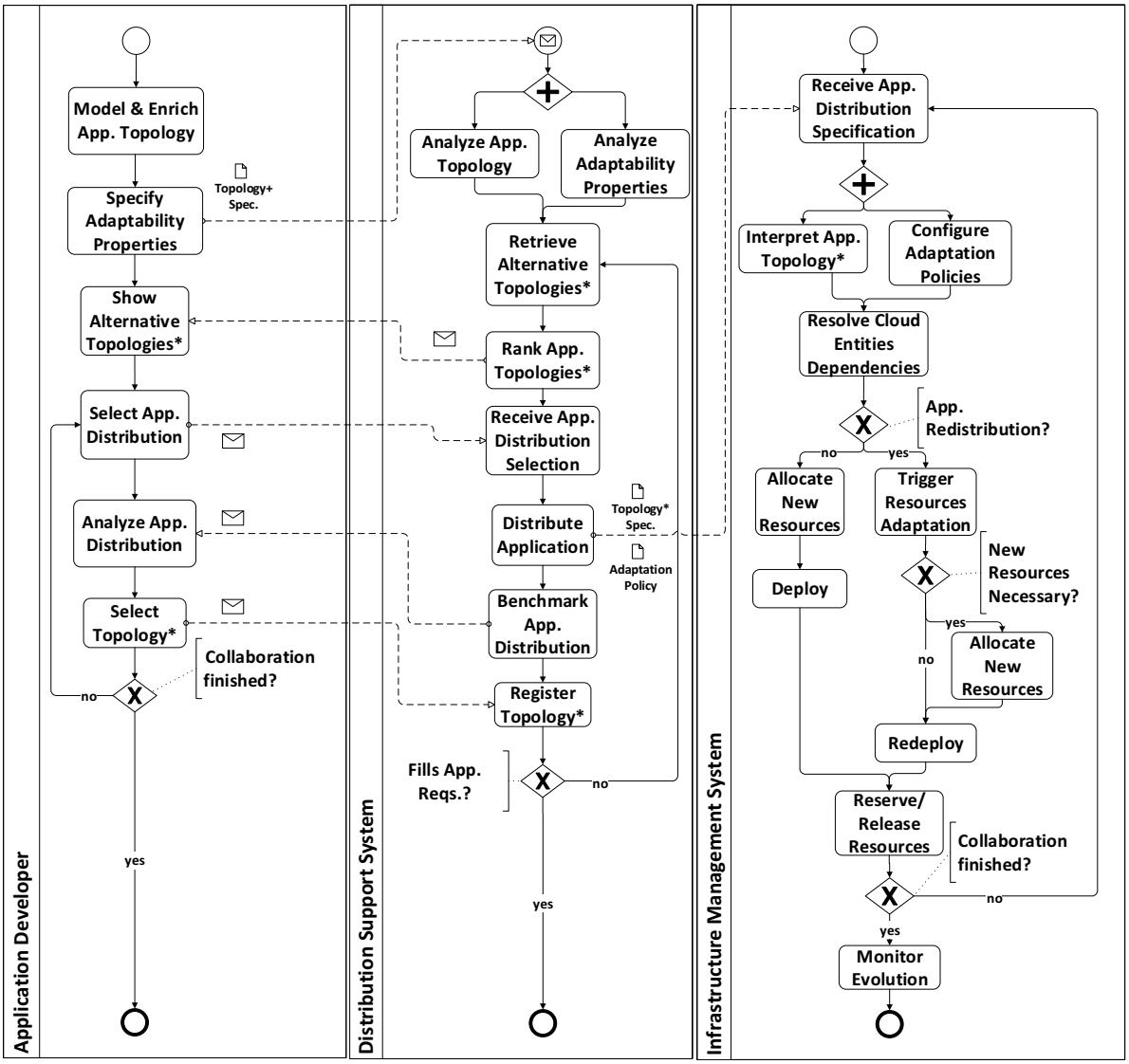

Figure 1: Application Distribution Support Process

allows developers to predefine load balancing and auto scaling properties. In this task, the adaptation model has to be specified describing which kind of changes can be executed during the execution of an application in the cloud. Furthermore, as identified in [6], cloud application adaptation is not necessarily limited to auto-scaling. The adaptation model therefore captures the different types of adaptations based on both application and cloud infrastructure support for changes. Consequently to facilitating the application description in terms of its topology and adaptability properties, a ranked set of concrete application distribution alternatives is proposed to the developer, depicted as *Topologies** in Figure 1. Such alternatives may contain multiple cloud services within one or multiple cloud providers. After visualizing and analyzing the different application distribution alternatives (*Show Alternative Topologies**), the application developer selects the most suitable concrete application distribution and adaptation options (*Select Application Distribution*).

The selected alternative is then used by the Distribution Support System participant to instantiate the application in the cloud environment and evaluate its performance. The *Analyze Application Distribution* task consists of retrieving, interpreting,

and analyzing the results obtained from this application evaluation. This information can then be visualized by the Application Developer participant. The results from the aforementioned task are then under the responsibility of the *Topology* Selection* task. In case the topology parameters offered suit the expected behavior and results, such a topology can be registered as a suitable topology for a pre-defined set of functional and non-functional aspects for future usage, e.g., to trigger a dynamic and transparent application redistribution to meet application workload demands.

Distribution Support System. This system provides the application developer with the necessary artifacts to generate application distribution alternatives, rank them based on the functional requirements, non-functional aspects, and dynamic adaptations that cope with the properties predefined by the developer. Furthermore, the system aims to provide the means for application developers to analyze the empirical evaluation of the proposed application distributions and adaptation alternatives indicated by the support system.

In a first step, the enriched application topology and the adaptability properties defined by the developer are analyzed in order to infer alternative topologies and adaptation options sets. The *Retrieve Alternative Topologies** task retrieves the available cloud services from a cloud offerings knowledge base provided by this system or by external cloud providers. The system must explore the alternatives space, infer, and narrow the distribution alternatives in order to provide a set constituted by alternative topologies which fit the requirements and properties indicated by the developer. The set of alternative topologies has also a set of adaptation options associated with it that can be performed given the chosen topology and respecting the pre-define properties established by the developers. Consequently, the *Rank Application Topologies* task is responsible for ranking the different alternatives that constitute the alternative topologies set. Such a ranking must be performed according to the developer's preferences, e.g., using a utility-based calculation approach. The *Distribute Application* task realizes the actual distribution of the application. This task depends on the deployment engine capabilities supported by the the Infrastructure Management System. It must be able to materialize the topology specification provided by the previous task, as well as the automatic adaptation defined by the developer properties, e.g., using a policy attachment-based approach [16].

The multi-dimensional evaluation of the application (re-)distribution is performed by the *Benchmark Application Distribution* task. Such evaluation can be performed using existing solutions to analyze the different problem dimensions, e.g., by using benchmarking solutions or simulation frameworks that focus on performance [17], [18], or monetary costs using a cost calculator, etc. Alternative topologies, adaptation properties, and evaluation results can then be registered in the system towards creating a knowledge base for future use.

Infrastructure Management System. This is the intermediary support system used for managing one or multiple cloud infrastructures. Therefore, it must natively support managing the pulling, release, and adaptation of cloud entity instances considering the adaptation properties defined by the developer and the cloud entities dependencies resolution.

The *Interpret Application Topology** task interprets the application topology specification and generates an intermediate domain specific format for interacting with the application (re-)deployment engine. In the parallel step *Configure Adaptation Policies*, the system derives the adaptation properties defined by the developer and creates application specific adaptation policies for managing the underlying resources dynamic adaptations. Consequently to creating the means to interpret the application distribution and adaptations specified by the developer, the *Resolve Cloud Entities Dependencies* analyzes the application topology alternative instance, derives, and resolves the cloud entities dependencies. The support system identifies the dependencies at the different layers (Application Business Logic Layer, etc.) and the possible runtime adaptations. If the operation of the application does not conform with the specified properties, adaptations will be triggered and resources at different levels can be used for such adaptation. If the application has previously been deployed (application (re-)distribution case), the underlying cloud entities instances are adapted, and potential new cloud entity instances are allocated as task *Trigger Resources Adaptation* depicts. If the application has not been previously deployed, then underlying cloud entity instances must be allocated in the task *Allocate New Resources*.

Consequently to the (re-)distribution conditional analysis, in the *Reserve / Release Resources* task,

the cloud entity instances from the infrastructure must be pulled and reserved for (re-)distributing the application. The same must be taken into consideration when releasing cloud entity instances that were previously used by a concrete application distribution. If the collaboration is finished, i.e. no further application redistribution and evaluation is necessary in the short term, then the system monitors the application performance and behavior evolution over time. Relevant metrics are stored towards deriving behavioral and performance patterns, e.g., based on cost, QoS, application workload variations, triggered cloud entity adaptations, etc.

4. Evaluation

In this section, we present the evaluation of the previously presented process-based approach by means of a case study based on an existing application, the Apache Olio v0.2. This application emulates a Web 2.0 social event application developed in various programming languages for purposes of evaluating the performance of web technologies. In the scope of this work we use the PHP application release, whose architecture is constituted by a wide landscape of components and technologies, such as a MySQL database system, MemCached for caching purposes, a distributed storage system, and an external emulated service. Such variety enables the exploration of multiple cloud offerings and deployment scenarios, and the derivation multiple topology alternatives.

Following the *Application Cloud Distribution Support Process*, and after modeling and enriching the application topology with functional and non-functional aspects, the multiple viable alternative topologies denoted with dashed lines in the Figure 2 are derived. For this purpose, an interaction with a knowledge system capable of maintaining a registration of available cloud offerings is required.

The alternative topologies space must be analyzed by the developer, and therefore a ranked list according to, e.g., developer preferences, must be generated applying, e.g., utility-based calculations over the non-functional aspects of the application. For the sake of simplicity, we assume that the application developer selects the topology alternative constituted by the following components, cloud offerings and relations: 1) The Apache Olio Frontend is hosted on-premise, 2) the Geocoder emulator is deployed in an AWS EC2 m1.medium VM instance, 3) an AWS ElastiCache instance is used to mitigate the web pages retrieval from the application database migrated to 4) an AWS RDS db.m1.large MySQL database instance.

When the developer selects a topology alternative, the distributed deployment of the application in the Cloud is triggered. This list of alternatives is illustrated in Figure 2, where the dashed circles in this figure indicate the alternatives of topologies for the specific application. Together with alternative topology nodes, we also define adaptations that can be activated for this example. Two types of adaptation elements can enrich the application topology model. First, we define the *Elastic Adaptation-aware Specification* document, which is used by the *Cloud Platform Management Support* participant in our process to configure the elastic adaptation actions over the nodes and alternative nodes of the application topology. In addition, we also define the *Intra-Node Adaptation-aware Specification* document. This document contains adaptation actions that are not directly related to elasticity, but enable changes inside the application node that can appease problems experienced by the application, before going for elastic solutions. Examples of such intra-node adaptation options would be the reconfiguration of number of connections in the MySQL server. The Infrastructure Management System then interprets the topology and configures the adaptation properties and constraints for the application and its underlying resources. Focusing on the provided example, for the AWS ElastiCache offering, the elasticity features can be configured, while for the AWS RDS database instance a high availability can be enabled through the usage of the multi availability-zones deployment features.

The allocation and adaptation of new provisioned or adapted resources, respectively, can be performed through the usage of dynamic provisioning and deployment of cloud services. After the complete deployment of the application stack in a distributed manner, the infrastructure must be able to provide monitoring capabilities in order to retrieve and evaluate the evolution of the application behavior focusing on the specified non-functional aspects, e.g., performance in terms of transactions per second. The resource provisioning and application deployment phases are followed by the evaluation of the application distribution using benchmarking techniques. The *Application Distribution Support* system must maintain and

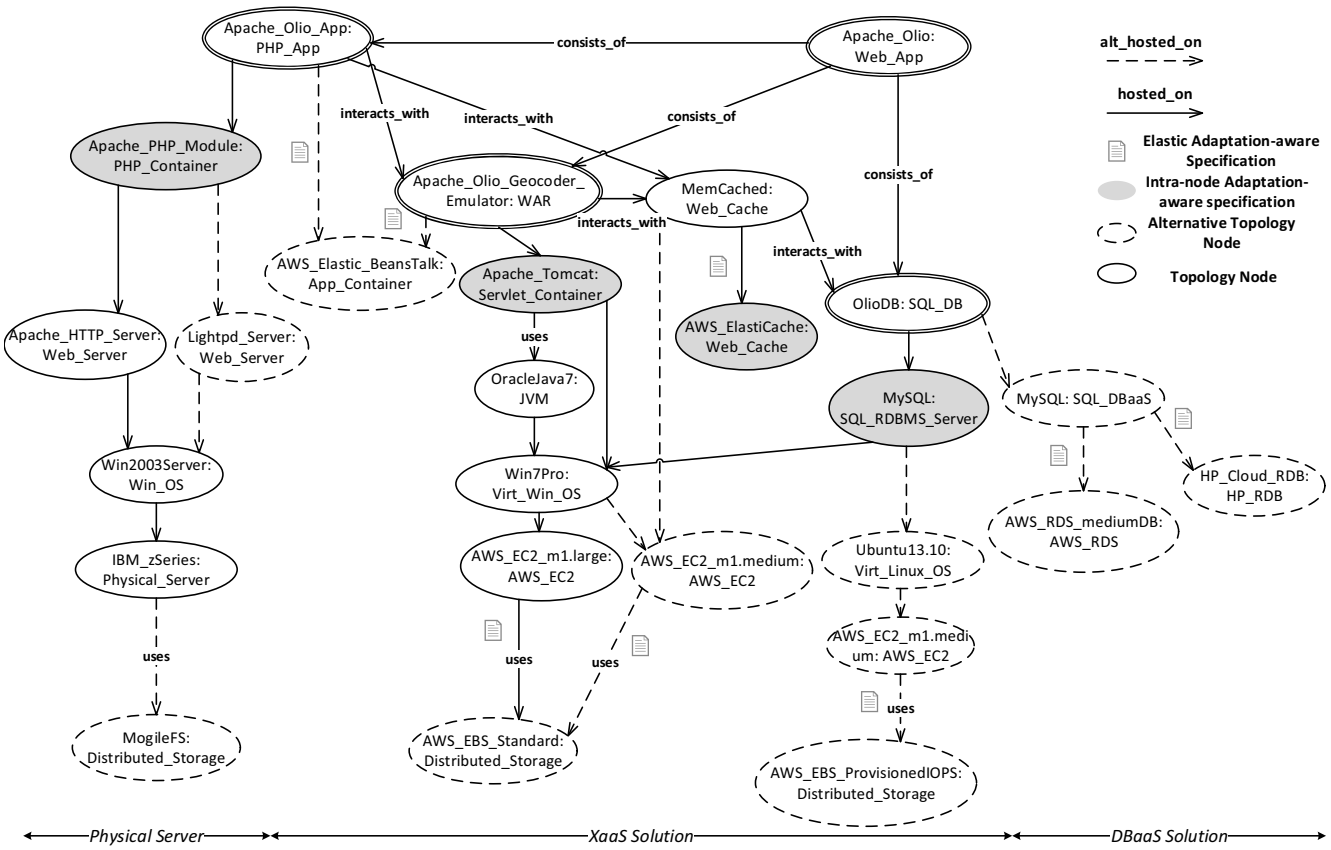

Figure 2: Apache Olio v0.2 Application Topology & Alternatives

provide a pool of heterogeneous benchmarks capable of covering the different problem dimensions which must be evaluated and analyzed, e.g., the number of transactions per second using the Rain workload generator and the Faban Benchmark as discussed in [9]. Such evaluation facilitates and provides the means to empirically and quantitatively analyze the application distribution from the developer's perspective. The realization of such benchmarking pool and empirical evaluation are future work.

5. Related Work

The creation of the viable topology alternatives set by exploring the available cloud offerings and focusing on a set of dimensions usually involving operational expenses has been addressed in other investigations. Approaches such as Policy4TOSCA [16] enable the policy-based description of application non-functional aspects. However, such requirements describe properties or configuration parameters in the TOSCA specification which are applied to the topology instance predefined by the developer, rather than used to derive the topology alternatives that meet such requirements. The MOCCA framework focuses on optimizing the application topology by introducing variability points in the topology model and using optimization techniques to find the most suitable cloud offerings [19]. The CloudMig approach builds on an initial topology [20] of the application which is adapted through model transformation based on existing cloud offerings. The approach in [21] uses a Palladio-based application topology model in order to distribute an application across different cloud providers aiming at optimizing for availability and operational expenses. The comparison and ranking of cloud offerings focusing on non-functional aspects is supported through simulation techniques in the SMICloud Framework [22]. The analysis of cloud deployment options for migrating software to the Cloud is also addressed in CDOSIM [23] through simulation. These approaches do not consider the proposed dynamic adaptation features, but can be extended and used as part of the process implementation.

It terms of cloud adaptation, approaches such as [24] deal with the dynamic adaptation of the infrastructure resources to both comply with the application topology and ensure the SLA. Islam et al. deliver a method for the consumer to measure elasticity properties of different cloud platforms [25]. An experimental analysis of scalability and performance was conducted by Jayasinghe et al. [26]. They focus on the migration of n-tier applications to IaaS clouds and employed the RUBBoS benchmark to compare the performance of different cloud environments (e.g., amount of concurrent users vs. throughput (requests per second), CPU utilization and response time). The SMICloud Framework [22] also considers the deployment and adaptation in terms of elasticity but ignores the developer in the decision making process. For identifying options of adaptation this approach can be extended to take the developers and their desired application focus into account. Lee et al. define a quality model in order to evaluate Sofware-as-a-Service in cloud environments [27]. The authors define metrics for quality attributes such as *efficiency*, *reliability* and *availability*. These metrics could be used to determine options of adaptation by triggering actions when a certain threshold is reached. Suleiman et al. [3] focus on the economics and elasticity challenges of computing resources in public cloud infrastructures. They identified metrics in the literature which describe bottleneck points in terms of capacity and performance.

The approaches mentioned above do not provide any models that can be immediately used by the developer to represent the possible options for adaptation. In general they only focus on elasticity as the one type adaptation and no other types are mentioned (e.g., considering the reconfiguration of a load balancer queuing algorithm). Nonetheless, the metrics presented can be used as a base for triggering and determining specific adaptation actions. Finally, in the context of the SLA@SOI research project protocols and engines were developed which provide mechanisms for adaptation that are triggered by SLA violations [28]. A modelling language has been created to express non-functional requirements and re-negotiation. The triggering of this re-negotiation resembles some type of adaptation but it does not specify which actions should be performed. This deficit is addressed by our approach.

6. Conclusions and Future Work

In this paper we introduced a detailed support process that allows cloud application developers to make more informed decisions about deployment and adaptation aspects of their applications that builds on bridging and synthesizing previous works. For this purpose, we defined a process with three main participants. The Application Developer is involved in the tasks associated with enriching the application specification and choosing among the offered options of distribution and adaptation. The Application Distribution Support System entails the alternatives generating options for distribution and adaptation based on the enriched specification. The Infrastructure Management System is associated with the tasks of interacting with cloud providers, and the effective distribution and adaptation of the application in the cloud environment.

Future work focuses on fleshing out the individual tasks identified in the process and connecting them with the specific techniques and tools that can be used for their realization. Consequently, the goal moves to developing a toolkit that will allow our proposal to be evaluated on the field based on real world applications.

ACKNOWLEDGEMENTS

This work is funded by the EU FP7 projects CloudWave (610802) and ALLOW Ensembles (600792).

References

[1] V. Andrikopoulos, T. Binz, F. Leymann, and S. Strauch, "How to Adapt Applications for the Cloud Environment," *Computing*, vol. 95, no. 6, pp. 493–535, 2013.

[2] A. Bankole and S. Ajila, "Cloud client prediction models for cloud resource provisioning in a multitier web application environment," in *Proceedings of SOSE'13*, March 2013, pp. 156–161.

[3] B. Suleiman, S. Sakr, R. Jeffery, and A. Liu, "On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure," *Journal of Internet Services and Applications*, vol. 3, no. 2, pp. 173–193, 2012.

[4] S. H. Liew and Y.-Y. Su, "Cloudguide: Helping users estimate cloud deployment cost and performance for legacy web applications," in *Proceedings of CloudCom'12*, Dec 2012, pp. 90–98.

[5] V. Andrikopoulos, S. Gómez Sáez, F. Leymann, and J. Wettinger, "Optimal Distribution of Applications in the Cloud," in *Proceedings of CAiSE'14*. Springer, June 2014, *(to appear)*.

[6] Clarissa Cassales Marquezan *et al.*, "Towards exploiting the full adaptation potential of cloud applications," in *Proceedings of PESOS'14*, 2014, pp. 48–57.

[7] T. Binz, G. Breiter, F. Leymann, and T. Spatzier, "Portable Cloud Services Using TOSCA," *Internet Computing, IEEE*, vol. 16, no. 3, pp. 80–85, 2012.

[8] M. P. Papazoglou and W. van den Heuvel, "Blueprinting the cloud," *Internet Computing*, vol. 15, no. 6, pp. 74–79, 2011.

[9] S. Gómez Sáez, V. Andrikopoulos, F. Leymann, and S. Strauch, "Towards Dynamic Application Distribution Support for Performance Optimization in the Cloud," in *Proceedings of CLOUD'14*, June 2014, *(to appear)*.

[10] M. Miglierina, G. P. Gibilisco, D. Ardagna, and E. D. Nitto, "Model based control for multi-cloud applications," in *Proceedings of MiSE'13*, 2013, pp. 37–43.

[11] P. Leitner et al., "Cloudscale: A novel middleware for building transparently scaling cloud applications," in *Proceedings of the SAC '12*, 2012, pp. 434–440.

[12] M. Papazoglou and W. van den Heuvel, "Blueprinting the Cloud," *Internet Computing, IEEE*, vol. 15, no. 6, pp. 74–79, 2011.

[13] F. Curbera *et al.*, *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall International, 2005.

[14] M. Fowler, *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2002.

[15] B. J. Watson *et al.*, "Probabilistic Performance Modeling of Virtualized Resource Allocation," in *Proceedings of ICAC'10*, 2010.

[16] T. Waizenegger *et al.*, "Policy4TOSCA: A Policy-Aware Cloud Service Provisioning Approach to Enable Secure Cloud Computing," in *OTM'13 Conferences*. Springer Berlin Heidelberg, September 2013, pp. 360–376.

[17] A. Beitch *et al.*, "Rain: A Workload Generation Toolkit for Cloud Computing Applications," University of California, Tech. Rep. UCB/EECS-2010-14, 2010.

[18] W. Sobel *et al.*, "Cloudstone: Multi-platform, Multi-language Benchmark and Measurement Tools for Web 2.0."

[19] F. Leymann *et al.*, "Moving Applications to the Cloud: An Approach based on Application Model Enrichment," *IJCIS*, vol. 20, no. 3, pp. 307–356, October 2011.

[20] S. Frey and W. Hasselbring, "The cloudmig approach: Model-based migration of software systems to cloud-optimized applications," *International Journal on Advances in Software*, vol. 4, no. 3 and 4, pp. 342–353, 2011.

[21] M. Miglierina, G. Gibilisco, D. Ardagna, and E. Di Nitto, "Model based control for multi-cloud applications," in *Proceedings of MiSE'13*, 2013, pp. 37–43.

[22] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Gener. Comput. Syst.*, vol. 29, no. 4, pp. 1012–1023.

[23] F. Fittkau, S. Frey, and W. Hasselbring, "Cdosim: Simulating cloud deployment options for software migration support," in *Proceedings of MESOCA'12*. IEEE, 2012, pp. 37–46.

[24] A.-F. Antonescu, P. Robinson, and T. Braun, "Dynamic topology orchestration for distributed cloud-based applications," in *NCCA*, 2012, pp. 116–123.

[25] S. Islam, K. Lee, A. Fekete, and A. Liu, "How a consumer can measure elasticity for cloud platforms," in *Proceedings of ICPE'12*. ACM, 2012, pp. 85–96.

[26] D. Jayasinghe *et al.*, "Variations in performance and scalability when migrating n-tier applications to different clouds," in *Proceedings of CLOUD'11*. IEEE Computer Society, 2011, pp. 73–80.

[27] J. Y. Lee, J. W. Lee, D. W. Cheun, and S. D. Kim, "A quality model for evaluating software-as-a-service in cloud computing," in *Proceedings of SERA'09*, pp. 261–266.

[28] Wieder, P., Butler, J.M., Theilmann, W., Yahyapour, R., *Service Level Agreements for Cloud Computing*. Springer.

All links were last followed on June 19, 2014.