Consolidation of Performance and Workload Models in Evolving Cloud Application Topologies

Santiago Gómez Sáez, Vasilios Andrikopoulos and Frank Leymann

Institute of Architecture of Application Systems, University of Stuttgart, Universitätsstr. 38, Stuttgart, Germany

Keywords: Cloud Application Topology, Cloud Application Distribution, Cloud Application Performance Engineering, Cloud Application Workload.

Abstract: The increase of available Cloud services and providers has contributed to accelerate the development and has broaden the possibilities for building and provisioning Cloud applications in heterogeneous Cloud environments. The necessity for satisfying business and operational requirements in an agile and rapid manner has created the need for adapting traditional methods and tooling support for building and provisioning Cloud applications. Focusing on the application’s performance and its evolution, we observe a lack of support for specifying, capturing, analyzing, and reasoning on the impact of using different Cloud services and configurations. This paper bridges such a gap by proposing the conceptual and tooling support to enhance Cloud application topology models to capture and analyze the evolution of the application’s performance. The tooling support is built upon an existing modeling environment, which is subsequently evaluated using the MediaWiki (Wikipedia) application and its realistic workload.

1 INTRODUCTION

The existence of a wide technological landscape offered in the Everything-as-a-Service (*aaS) model has contributed to an increase of applications partially or completely running or built in the Cloud, potentially as a composition of Cloud services (Andrikopoulos et al., 2013). The adoption of continuous software delivery models, such as DevOps, aim at offering flexibility and agility for a quick response to market changes. The DevOps emergence boosted efforts in research and industry towards developing concepts and tools to assist application developers to develop, provision, and (re)deploy cloud applications in a simplified, interoperable, and agile manner.

Standards like TOSCA\(^1\) allow the automated and interoperable provisioning and configuration of Cloud services to host the different application components. However, there exists a lack of native support for assisting application developers in the selection and configuration of appropriate Cloud services based on a set of business and operational objectives. When focusing on the performance of the application, such an analysis becomes even more complex and wider, as (i) the fluctuation of the application workload has an impact on the resources demand and QoS, and (ii) the existence of multiple applications running on the same physical environment of a provider has an unpredictable impact on the offered performance (Gómez Sáez et al., 2015). Towards narrowing such a gap, in (Gómez Sáez et al., 2014) we proposed a process-based approach aimed at assisting application developers to efficiently (re)distribute their application components spanning multiple Cloud offerings while focusing on fluctuating and evolving workloads and performance demands. This work materializes the vision described in (Gómez Sáez et al., 2014) by developing the concepts for consolidating performance aspects in Cloud application topologies. The main contributions of this work are:

1. the derivation of a life cycle targeting the specification, analysis, and adaptation of Cloud applications for evolving workloads and performance requirements,

2. the development of the foundations to enrich Cloud application topology models with evolving business and operational requirements, and workload behavioral models,

3. its corresponding tooling support built atop the

\(^1\)Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0: http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html
TOSCA\(^2\) and OpenTOSCA\(^3\) specification and ecosystem, respectively, and
4. the evaluation of the proposed approach using the
well known MediaWiki\(^4\) application and its realistic
workload.

The remainder of the paper is structured as follows. Section 2 presents the background this work builds upon. The proposed life cycle for application (re)distribution, and the meta-models for enriching Cloud application topologies with evolving performance and workload models are presented in Section 3. The tooling support is presented in Section 4, which is then evaluated in Section 5. Section 6 presents related works, and Section 7 conclusions and future work.

2 MOTIVATION & BACKGROUND

Figure 1 depicts multiple viable distributions of the two-tiered PHP-based MediaWiki\(^5\) application. The MediaWiki topology breaks the application stack in two main component groups: (i) application specific components, i.e. each application tier description, and (ii) the application independent sub-topology/ies (i.e. the \(\alpha\)- and \(\gamma\)-topologies in the terminology of (Andrikopoulos et al., 2014a), respectively). The available number and nature of Cloud offerings allows for a partial or complete distribution of the application components, therefore building a wide spectrum of alternative \(\mu\)-topologies (Andrikopoulos et al., 2014a).

For instance, it is possible to outsource the MediaWiki front-end tier to a VM-based offering, such as Amazon EC2\(^6\), and migrate its backend database tier to an off-premise EC2 VM or in an Amazon RDS\(^7\) database instance.

The existence of such a spectrum of alternative \(\mu\)-topologies together with the necessity to rapidly satisfy changing business and operational requirements introduces a multi-dimensional problem involving the analysis and revision of requirements and objectives. More specifically, it involves the evaluation of the trade-off between two or more dimensions, e.g. cost vs. performance, etc., during the modeling and production phases of the application. Towards bridging such a gap, in (Gómez Sáez et al., 2014) and (Andrikopoulos et al., 2014a) we proposed a methodology to distribute and redistribute their application components spaced among multiple Clouds to cope with changing business and operational requirements, and fluctuating workloads.

The Performance Aware Cloud (Re)Distribution Process depicted in Figure 2 consists of several tasks: (i) modeling the application topology, (ii) enriching such topology with business and operational requirements, and the workload characteristics, which are then (iii) processed and analyzed to subsequently (iv) discover, construct, and evaluate alternative \(\mu\)-topologies. The (v) deployment and production phase of the application assists in building the necessary knowledge to (vi) analyze the evolution of the appli-
cation performance demands and workload behavior through monitoring techniques. The Collaborative Loop gears towards supporting the (re-)distribution of the application over time to rapidly react to changing requirements and fluctuating workloads. This research paper provides the conceptual and tooling support towards supporting the Modeling and Enrichment tasks in Figure 2.

3 PERFORMANCE-AWARE MODELING & ENRICHMENT OF CLOUD APPLICATIONS

This section presents two fundamental aspects that must be taken into consideration for achieving an agile (re)distribution of Cloud applications spanning multiple Clouds: (i) the development of a life cycle for selecting and configuring Cloud resources to satisfy application business and operational requirements, and (ii) the derivation of the necessary foundations in the Modeling and Enrichment tasks depicted of the Performance Aware Cloud Application (Re)Distribution Process to support the different phases of the life cycle.

3.1 Life Cycle

The specification and analysis of the application business and operational performance require to consider two aspects: (i) the difference between the required and offered performance, and (ii) the evolution of the application workload behavior. According to several investigations (Bahga and Madisetti, 2011; Gmach et al., 2007; John et al., 1998; Mian et al., 2013; Watson et al., 2010), two approaches can be derived for designing and provisioning adaptable Cloud applications w.r.t. changing business and operational requirements, and fluctuating workloads: top-down and bottom-up. The life cycle depicted in Figure 3 supports the (re)distribution of applications w.r.t. their business and operational requirements. A first phase consists of the KPI Specification, by means of defining and specifying the business and operational requirements. Together with the specification and analysis of the application workload in the Workload Model Derivation phase, the resources can be selected and configured in the Resources Selection & Configuration phase. However, the previous phases do not directly enable the analysis of the workload fluctuation. Towards such a goal, the Workload Evolution & Characterization phase allows to analyze the workload evolution during the application’s production phase, using, e.g. monitoring techniques.

The execution of the previous phases in the top-down approach empowers the allocation of resources to satisfy business and operational requirements. The bottom-up approach builds on the adaptation of resources and in the refinement of business and operational requirements. The bottom-up approach mainly builds towards deriving the optimal resource allocation and configuration w.r.t. the application performance (Mian et al., 2013). In this work we build towards the consolidation of the top-down and bottom-up application workload analysis approaches over time in order to proactively satisfy application demands by dynamically (re-)adapting its topology.

3.2 KPI Requirements Specification

The specification of business and operational requirements typically relate to the application’s business KPIs, e.g. expected profit, maximum accepted latency to ensure a user’s satisfaction level, etc. The meta-model depicted in Figure 4 establishes the placeholders for the specification and the analysis of adaptive performance requirements in the KPI Specification phase of the Application (Re)Distribution Life Cycle. Performance Requirements can be partitioned in two main correlated groups (see Figure 4): Operational Requirements and Business Requirements. Operational Requirements relate to the provisioning and configuration of resources, etc., while the Business Requirements relate to the business objectives, e.g. expected revenue per user, maximum expenditure, etc. Both Business and Operational Requirements have one overlapping characteristic: Metrics can be used to quantitatively analyze and evaluate the fulfillment of such requirements (see Figure 4).

In order to simplify the selection of performance metrics, these can be classified and organized in the proposed meta-model within different Metric Cate-
For such a purpose, in this section we identify the necessary artifacts and we derive a the meta-model for enhancing Cloud application topologies with workload models. For the scope of this work, we define the application’s workload as the description of a set of business transactions which are probabilistically distributed for a time interval, have an impact on the application state, and define the behavioral characteristics of its corresponding users.

The Workload Behavior Specification meta-model depicted in Figure 5 builds upon the workload description presented in (Van Hoorn et al., 2008), which we enhance for composite Cloud applications. Such meta-model defines the placeholders for specifying or building workload behavioral models in the Workload Model Derivation and Workload Evolution & Characterization phases of the Application (Re)Distribution Life Cycle (see Figure 3).

The application’s Workload Profile consists of a set of Workload samples, each comprising its Usage Profile, its Workload Mix, and its Behavioral Model. The workload’s Usage Profile describes the evolution of the application’s end Users behavior, in terms of their Arrival Rate and the specification of concurrent or non-concurrent requests. Every User executes a set of Business Transactions sent over a specific transport protocol supported by the application’s Component Protocol. The set of transactions performed on
the application are distributed within a **Workload Mix**, based on, e.g. popularity, probability of occurrence, etc., each having an impact on every **Application State** of the **Application Model**. For example, transitions between **Application States** originate in the execution of the user’s requests, e.g. log-in operations, Wiki page search, etc. The distribution of requests within the workload mix and the distribution of users over time define a **Behavioral Model**, i.e. a statistical model representing the behavioral characteristics of users and requests, which can be leveraged in order to drive statistical analyses, categorizations, or estimations.

### 3.4 Cloud Resources Selection & Reconfiguration

In this section we investigate how Cloud application topology models can be enhanced with the previously depicted knowledge. Since Cloud application topologies describe the set of components, services, and the relationships among them, the enrichment support must take into consideration the different levels where Cloud application topology models can be enriched. The remaining of this section outlines in a fined granular manner such possibilities.

Due to the generic nature of GENTL among different Cloud application topology languages, we use it as the basis for analyzing the modeling and enrichment points of Cloud application topology models (An- drikopoulos et al., 2014b) (see Figure 6). KPI requirements and workload behavioral characteristics can be fundamentally specified in two granular ways. A fined granular description consists of decomposing and describing the performance requirements and workload characteristics, respectively, e.g. on the topology **Component** or **Relationship** levels. For instance, the usage of different storage services to store application data, such as AWS S3\(^8\) or AWS RDS, may require a fined granular description of the workload operations for each topology **Component** independently, due to the different nature of such data services. **Relations** among the topology components depicted as **Connectors** have a significant impact in the overall performance of applications.

---

\(8\) AWS S3: http://aws.amazon.com/s3
applications, as they are usually impacted by the network configuration and characteristics. The specification of performance information in sub-topologies enables to group reusable Cloud components and services with common KPI requirements and workload behavioral characteristics.

The enhancement of Cloud application topologies with KPI requirements and workload information can be leveraged during the design, provisioning, and production phases of the application for predicting and analyzing the performance of Cloud offerings. Such information can be leveraged in future decision making tasks towards assisting application developers to select, configure, and dynamically adapt Cloud resources.

4 ARCHITECTURE & IMPLEMENTATION

In the following we discuss the tooling support to support Cloud application developers in the Modeling and Enrichment tasks of performance-aware Cloud application topologies. Due to the high adoption of the TOSCA standard in both the industry and research domain, we build the technological support atop of the TOSCA specification and its corresponding modeling environment OpenTOSCA Winery (Kopp et al., 2013).

PERFinery - an OpenTOSCA Winery Extension

PERFinery is a TOSCA-based modeling environment geared towards the enrichment of TOSCA-based Cloud application topologies with evolutionary performance requirements and workload models.

The specification of non-functional requirements of Cloud applications is supported in TOSCA by attaching custom policies to the application topology, typically conforming to the Policy4TOSCA definition (Waizenegger et al., 2013). Policy4TOSCA enables the definition of Policy Types and Policy Templates comprising actions which must be performed at concrete phases of the application life cycle and on specific layers of the application. However, Policy4TOSCA lacks of tooling support and does not capture evolving performance information, such as the influence of workloads on the fulfillment of KPI requirements. The topology enrichment support in this work empowers the current TOSCA policy definition support by enabling the graphical creation of custom Policies comprising the different measurable and analyzable application requirements.

Figure 7 depicts the architecture of PERFinery, and highlights the components that have been extended in Winery. The Topology Elements Manager, Topology and Plan Modeler, and the Repositories are the major components (see Figure 7). The Topology Elements

PERFinery Modeling Environment: http://www.iaas.uni-stuttgart.de/PERFinery/
Manager enables the administration and management of reusable artifacts among multiple topology templates, e.g. Node Types, Relationship Types, or Deployment Artifacts. The Topology Modeler provides the modeling artifacts towards designing and visualizing Cloud application topologies. PERFinery comprises repositories for storing Cloud application topology models and reusable artifacts. The modeling elements are provided through the Topology Modeler and Element Manager client-side GUIs developed using Java Web technologies and HTML5 (see Figure 7). Moreover, a REST interface is also offered atop of the Topology Repository for persisting and retrieving TOSCA artifacts, e.g. CSAR packages. The generated TOSCA and Policy4TOSCA templates, and workload specifications, as well as the corresponding CSAR packages, are persisted in separate repositories. The persistence of workload models is driven and persisted in an independent Workload Repository. Generated workloads can be used towards enriching modeled application topologies with their behavioral characteristics, e.g. user arrival rate, transactions’ mix and distribution of requests, etc.

Application topology and workload models can be created, viewed, and modified by navigating through the repository and the different sections that PERFinery offers. For instance, Figure 8 depicts the topology model created for the MediaWiki application used for the motivation of this work, which is subsequently enhanced with performance information.

5 EVALUATION

We evaluate our approach using the MediaWiki (Wikipedia) application and its realistic workload provided in (Urdaneta et al., 2009). The MediaWiki front-end tier encapsulates the presentation and business logic layers, while its back-end provides its persistency mechanisms. We used the workload provided in Wikibench\(^\text{10}\) for describing the workload. Figure 8 shows the view for building custom performance Policy Profiles.

\(^{10}\text{Wikibench Project: http://www.wikibench.eu/}\)

W.r.t. the workload specification, a first step consisted of sampling 200K HTTP requests of the original WikiBench workload describing the characteristics depicted in Table 1. However, such a workload sample is not distributed among the different users. For this purpose, the workload sample is referenced in PERFinery together with the specification of its behavioral characteristics, as depicted in Figure 9, respectively. More specifically, the workload model is specified in CSV format and comprises multiple users arriving at different time intervals and executing a set of requests mixes defined in the workload sample.

The extensions realized in Winery serve as the basis for enriching TOSCA topologies with performance information. More specifically, it provides Cloud application developers with the means to graphically include QoS information as TOSCA policies and workload descriptors.

<table>
<thead>
<tr>
<th>Operation Type</th>
<th>#Requests</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>199925</td>
<td>99.96%</td>
</tr>
<tr>
<td>Write</td>
<td>75</td>
<td>0.04%</td>
</tr>
<tr>
<td>Image Retrieval</td>
<td>8971</td>
<td>4.49%</td>
</tr>
<tr>
<td>Skins Retrieval</td>
<td>66926</td>
<td>33.46%</td>
</tr>
<tr>
<td>Wiki Pages Retrieval</td>
<td>106347</td>
<td>53.17%</td>
</tr>
<tr>
<td>Others</td>
<td>7634</td>
<td>3.81%</td>
</tr>
</tbody>
</table>

Table 1: Workload Characteristics.

6 RELATED WORK

Most analytical approaches and frameworks in the literature focus on combining operational expenses analysis with one or more dimensions pertaining to performance as part of their mission to support Cloud application developers.

DADL is presented in (Mirkovic et al., 2010) as a language to describe the architecture, behavior and needs of a distributed application to be deployed in the Cloud, as well as describing available Cloud offerings for matching purposes. Similarly, in (Antonescu et al., 2012), the authors propose a policy and action-based approach that matches and dynamically adapts the allocation of infrastructure resources to an application topology in order to ensure SLAs. The CloudMig (Frey and Hasselbring, 2011) approach builds...
on an initial topology and utilization model of the application that is mapped or adapted through model transformation in order to optimize the distribution of the application across Cloud offerings. The MODA-Cloud (di Nitto et al., 2013) and CloudML (Brandtzæg et al., 2012) projects focus on providing a multi-dimensional early design support of applications by applying model transformation techniques and code generation for multi-cloud applications. Further multi-cloud application distribution approaches are targeted by the SeaClouds EU Project\footnote{SeaClouds EU Project: http://www.seaclouds-project.eu/project.html}, by means of providing a Cloud Service Orchestrator capable of provisioning and managing application components spanned among multiple Cloud environments (Brogi et al., 2014). The CACTOS EU Project\footnote{CACTOS EU Project: http://www.cactosfp7.eu/} is possibly the closest approach to the fundamentals developed as part of this work. The CACTOS environment aims at fitting resources within a provider for diverse application workloads. However, all previous approaches introduce complex tasks, e.g. creation of simulation models, which often require the intervention of domain experts, causing an overhead in the development and (re)deployment tasks of applications. Moreover, the relationship of topology models with varying application workloads is not yet fully covered.

Further optimization of distribution of applications like the Palladio-based approach discussed in (Miglierina et al., 2013) aims at optimizing for availability and operational expenses. Moreover, optimization mechanisms are based on simulation techniques requiring the definition of their corresponding models. The MOCCA framework (Leymann et al., 2011) deals with the same problem by introducing variability points in the application topology in order to cope with possible alternative deployment topologies. CMotion (Binz et al., 2011) uses an approach based on topology modeling, generation of alternative topologies, and consequent evaluation and selection of one of those alter-
natives based on multiple criteria. The work in (Andrikopoulos et al., 2014a) uses the notion of typed graphs for similar purposes and proposes a formal framework to support this effort. In a similar approach, MADCAT (Inzinger et al., 2014) incorporates to the topology model scalability elements, and refines the topology model from a high-level application topology to a ready for deployment one.

The vision this paper pursues aims at leveraging existing non-functional requirement specification approaches, such as the ones previously discussed, towards providing the conceptual foundations to specify the performance aspects and analyze the impact of fluctuating workloads for various Cloud application distribution and configuration alternatives in a simplified manner by enriching Cloud application topology models.

7 CONCLUSIONS AND FUTURE WORK

The heterogeneity of available Cloud services has become a challenge for application developers when considering a rapid and efficient selection and configuration of Cloud offerings. Application components can be distributed or replaced by different Cloud services, potentially spanned among multiple Clouds. Focusing on the business and operational performance of Cloud applications, there currently exists a lack of modeling and decision making support for capturing, analyzing, and assessing when migrating, configuring, and utilizing different Cloud services under fluctuating workloads and intermittent QoS levels.

The assessment of, and guidance in the distribution of multi-Cloud applications is the core motivation behind this work. We build towards enabling the efficient (re-)distribution of Cloud applications by means of selecting and configuring Cloud offerings to cope with fluctuating workloads and evolving performance demands. The first step towards such a goal is covered in this work by providing the means to support this effort. In a similar approach, MADCAT (Inzinger et al., 2014) incorporates to the topology model scalability elements, and refines the topology model from a high-level application topology to a ready for deployment one.

The vision this paper pursues aims at leveraging existing non-functional requirement specification approaches, such as the ones previously discussed, towards providing the conceptual foundations to specify the performance aspects and analyze the impact of fluctuating workloads for various Cloud application distribution and configuration alternatives in a simplified manner by enriching Cloud application topology models.

Acknowledgements

This work is partially funded by the FP7 EU-FET project 600792 ALLOW Ensembles and the German 610872 DFG Project SitOPT.

References


