Software Product Lines for Multi-Cloud Microservices-Based Applications

Gustavo Sousa, Walter Rudametkin, Laurence Duchien

Software Product Lines for Multi-Cloud Microservices-Based Applications

Gustavo Sousa, Walter Rudametkin, Laurence Duchien
Université de Lille / CRIStAL UMR 9189 / École Centrale de Lille / Inria
Lille, France
Email: firstname.lastname@inria.fr

Multi-cloud computing is the use of resources and services from multiple independent cloud providers [1]. It is used to avoid vendor lock-in, comply with location regulations, and optimize reliability, performance and costs. By combining private and public clouds from different providers, cloud customers can better exploit the cloud market to optimize their costs and quality of service while reducing vendor dependence.

Microservices architectural style [2] is increasingly used in cloud computing. It promotes decomposing applications into small services that can be independently scaled, thus optimizing resources usage. There is a synergy between microservices and multi-cloud computing. An application decomposed into small well-isolated microservices can be more easily deployed across different clouds. Consuming resources from multiple clouds improves reliability and scalability of microservices.

However, building multi-cloud systems is a very complex and time consuming task, which calls for automation and supporting tools. We propose an automated approach to build a multi-cloud environment for a microservices-based application, considering the following concerns.

Cloud providers heterogeneity: Cloud providers offer functionality at different levels of abstraction such as infrastructure (IaaS), platform (PaaS) and software (SaaS). Even at the same abstraction level, providers employ varying terminology and features that usually do not map directly those of competing providers, thus complicating their comparison.

Cloud providers variability: Cloud providers may have complex rules concerning their configuration. According to the provider, functionality may only be available in a given region or for a given user plan. Some features may have conflicts while other may have dependencies. These as well as other complex constraints found in cloud providers and should all be considered to obtain a proper configuration.

Multi-cloud requirements for microservices-based applications: Microservices can be developed by different teams, relying on different technologies and methodologies and may therefore require functionalities at different levels of abstraction. In a multi-cloud environment, microservices can be deployed across private and public clouds from different providers to implement scalability and redundancy mechanisms or to comply with location constraints.

Our approach to deal with this problem relies on the use of software product lines (SPLs) [3] and ontology reasoning to get from a high-level description of multi-cloud requirements to a selection of cloud providers and their configurations.

Multi-cloud application requirements are described in a high-level DSL. Developers can specify the functionality required by each service (e.g. application servers, databases, software platforms, scalability options, etc). They can also specify conditions such as that a database should be located in Europe or that a given service should be deployed to different providers in the same region, etc. Finally, they can define rules for scalability and redundancy within and across clouds.

Feature models [4] (FMs) are used to model variability in software product lines. In our case, a FM defines the features offered by a provider and how they are related. In a FM, we can specify constraints such as that a given database is only available in a certain region, that it conflicts with another database system, etc. A FM can be viewed as a tree of features that can be selected to build a cloud provider's configuration.

To build a valid multi-cloud environment we match application requirements against providers' offerings, described through FMs. A global cloud ontology is used to map equivalent concepts from requirements to offerings. For example, requirements can specify that a given service requires a relational database, stored in Europe with 10 GB of storage and 4GB of RAM. By reasoning over ontologies we are able to find that Google feature `Cloud SQL db-n1-standard-4` or Heroku feature `PostgreSQL Premium 2` can be used. Once we obtain a selection of features for each provider we use automated analysis [5] of FMs to find a valid and complete configuration for each selected cloud providers.

This approach allows for describing multi-cloud requirements for microservices-based applications in a provider independent way. By reasoning over ontologies we can deal with the heterogeneity across providers by identifying equivalent concepts. Modeling variability as FMs guarantees that the generated configurations are valid and follow providers' constraints and configuration options.

References

[1] N. Grozev *et al.*, "Inter-cloud architectures and application brokering: taxonomy and survey," *Software: Practice and Experience*, 2012.

[2] J. Lewis *et al.* (2014, Mars) Microservices: a definition of this new architectural term. [Online]. Available: http://martinfowler.com/articles/microservices.html

[3] K. Pohl *et al.*, *Software Product Line Engineering: Foundations, Principles and Techniques*, 1st ed. Springer Publishing Company, 2010.

[4] L. Chen *et al.*, "Variability management in software product lines: A systematic review," in *Proc. International Software Product Line Conference (SPLC'09)*, San Francisco, California, USA, Aug. 2009.

[5] D. Benavides *et al.*, "Automated analysis of feature models 20 years later: A literature review," *Information Systems*, 2010.