Chapter 11
Cloud Patterns

Teodor-Florin Fortiş and Nicolas Ferry

11.1 Introduction

A large number of design and architecture patterns have been identified during the last years, as the Cloud technologies were finding their path to maturity. In [1] Fehling et al., the authors expose a basic pattern-oriented view on Cloud computing, together with relevant patterns, view which is also applicable in the case of multi-Cloud applications.

Another set of more than forty patterns are included in the AWS Cloud Design patterns (CDP) [2], offering "*a collection of solutions and design ideas for using AWS Cloud technology to solve common systems design problems*".

In addition to the core set of Cloud design patterns, Erl et al. [3] propose a set of compound patterns, which, for most of them, are related to the essential characteristics of Cloud computing, such as Cloud bursting, elastic environment, multi-tenancy, Cloud deployment models, and others.

The IBM RedPaper [4] offers some insights on Pure Application Systems patterns and virtual application patterns (VAPs) which are "*a new Cloud deployment model that represents an evolution of the traditional topology patterns that are supported in virtual system patterns*". Finally, the Microsoft point of view on development of Cloud-hosted applications is covered by Homer et al. [5].

Complementary to the numerous design and architecture patterns that have already been described in the literature, a set of design heuristics or success factors was fully

T.-F. Fortiş (✉)
Institute e-Austria Timişoara and West University of Timişoara,
B-dul Vasile Pârvan 4, 300223 Timişoara, Romania
e-mail: fortis@info.uvt.ro

N. Ferry
Stiftelsen SINTEF, Postboks 4760, Sluppen, 7465 Trondheim, Norway
e-mail: nicolas.ferry@sintef.no

E. Di Nitto et al. (eds.), *Model-Driven Development and Operation of Multi-Cloud Applications*, PoliMI SpringerBriefs,
DOI 10.1007/978-3-319-46031-4_11

described in the context of the MODAClouds approach. This set will help mitigate various pitfalls when designing multi-Cloud applications.

11.2 Motivational Guidance

Important design heuristics and guidances have been identified as highly relevant for multi-Cloud applications, and especially in the context of MODAClouds.

Compute Partitioning

Compute partitioning is a design heuristic that helps building systems that can easily be maintained and deployed on Cloud platforms and infrastructures and advocates the utilization of patterns such as *loose coupling*, *compute partitioning*, *distributed applications* or *integration provider*. It allows application developers to efficiently exploit resources that can be provisioned with minimal effort. Particularly, as Cloud applications usually rely on multiple distributed resources, modularity and loose coupling become central for efficient exploitation of Cloud properties.

Thus, the separation of concern principle is essential in order to achieve the distribution of resources, as multi-Cloud application usually rely on resources possibly offered by multiple providers with their own specificities. This principle advocates decomposing and encapsulating the features of an application into modular and reusable blocks.

Based on the *computing partitioning guidance* [5] and using the *loose coupling* and *distribution application* patterns [1], the MODACloudML proposal is to decompose applications into logical components and help the user in allocating and reusing these components on Cloud resources.

Multiple Datacentre Deployment

Multiple datacentre deployment is one of the key factors that ensures successful deployments across multiple Cloud providers. This design heuristic relies on the *loose coupling* and *multiple datacentre deployment* patterns.

In the case of multi-Cloud applications, the providers of these applications will attempt to identify and exploit particularities of the underlying Cloud solutions in order to achieve an optimization of various characteristics (e.g., performance, availability, cost, etc.). Developers of such applications may therefore need novel design approaches in order to fully benefit from the varying sets of services that are supported by the different Cloud providers.

The approach considered in the case of MODAClouds consists in a separation of the design of the application from the technical specification of the underlying infrastructure as suggested by the MDA architecture. To achieve this separation, Cloud provider-independent models (CPIM) and Cloud provider-specific models (CPSM) are considered. The first ones enable the specification of Cloud provider-independent deployment scenarios in a Cloud agnostic way whilst the second allows selecting Cloud provider specific resources. CPIM should provide an appropriate

level of abstraction to allow the generation of CPSM, targeting various providers and being aware of their specificity at the same time. The identification of the right level of abstraction, as well as of the concepts that are relevant at the level of each of these models generates specific challenges in this scenario.

Instrumentation and Telemetry

Instrumentation and telemetry are key success factors in building feedback about the runtime performance of the system and its underlying platform and infrastructure. *Instrumentation and telemetry*, *loose coupling*, and *multiple datacentre deployment* are the most important patterns involved.

While in the case of a simple Cloud-application collecting some metrics related to the Cloud resources through provider's platform APIs may provide the right perspective on the behaviour of the application, this is not necessarily the case for multi-Cloud applications. Monitoring interfaces are likely to be incompatible and provider-specific, and therefore the monitoring activities could be subject to vendor lock-in. Moreover, it might not be enough to only monitor Cloud resource's usage in order to measure application's resource consumption and to provide efficient resource management activities.

Consequently, the MODAClouds approach supports this guidance and offer the means, at the level of the design-time platform and of the monitoring platform, to (i) allow the definition of monitoring rules at both the infrastructure and application levels in a provider-independent way, and (ii) enable the design of monitoring rules describing how incoming stream of data have to be processed, and what output should be produced when certain conditions have been verified.

11.3 MODAClouds-Specific Patterns

The guidance and design heuristics that were briefly described in Sect. 11.2 relate to an important number of Cloud design and architecture patterns, of which some can be adopted without major changes in a multi-Cloud context. However, a subset was specifically extended and adapted in the MODAClouds context to better support the design of multi-Cloud applications. We briefly describe these patterns in the following subsections.

External Configuration Store

The *external configuration store* pattern propose to outsource configuration and deployment information for any component or services of the system into separate services thus improving reusability and flexibility in the deployment and/or configuration process of application components. This pattern, as depicted in Fig. 11.1, extends the *configuration store pattern* [5] and it partially involves other patterns and mechanisms, like the *resource management system* mechanism [3].

In the case of MODAClouds, the configuration of a multi-Cloud application does not only include properties associated to the functional behavior of the application,

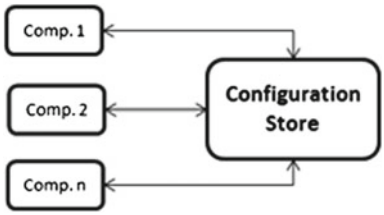

Fig. 11.1 The external configuration store pattern

but also provisioning and deployment information for the underlying infrastructure. Accordingly, the configuration store pattern was extended to include the overall information required for the deployment and configuration process of the multi-Cloud application. Therefore, one can achieve an externalization of the configuration and deployment information for any particular components or services into a separate service.

The use of this pattern could be relevant in various situations, like: (i) when the application contains several instances of the same component (or group of components), whose configuration must be synchronized; (ii) the configuration of the various components will have to be dynamically adjusted to accommodate various load and/or usage patterns; (iii) when similar reconfigurations need to be triggered on several parts of the application.

Leader-Followers

The aim of the *leaders-followers* pattern (or leader election pattern, see also [5]) is to dynamically delegate the management of subparts of the architecture to a separate component that has been elected. Such a feature is particularly relevant when Cloud applications aggregate several subsystems, with an appropriate level of complexity, such that the total complexity exceeds the capacity of a single management entity.

In a multi-Cloud context, the leader-followers pattern enables the election for each Cloud of a single component responsible for configuring and managing subparts of the execution environment. Thus, the leader (a master node) will have the necessary knowledge of its peers, managing their configurations accordingly.

This pattern is relevant especially (i) when the application contains numerous instances of the same component (or group of components), whose configuration and deployment must be synchronized, as in Fig. 11.2; (ii) when massive and simultaneous updates are necessary for instances of the same group of components.

Runtime Reconfiguration

The intent behind the *runtime reconfiguration* pattern is to dynamically reconfigure application components and frameworks as well as their execution environments to minimize the downtime in a production setting. This pattern is extended from the pattern with the same name from [5] to the dynamic adaptation of the application deployment using the `models@runtime` architecture. The use of this pattern together with the `models@runtime` architecture enables third-parties to adapt

only selected parts of the deployment whilst minimizing the downtime for the rest of the application.

Specific interest exists around this pattern especially when an application or the deployment of an application needs to be reconfigured dynamically at runtime, such as adapting logging policies, updating database connections, deploying new services, and others.

Particularly, in the case of MODAClouds, the `models@runtime` engine maintains a MODACloudML deployment model causally connected to the running system, and: (i) any modification to the CPIM will be reflected in the CPSM and propagated on-demand onto the running system; (ii) any change in the running system will be reflected in the CPSM, which, in turn, can be assessed with respect to the CPIM. Furthermore, by using the aforementioned deployment model, the `models@runtime` environment enables reducing the gap between the runtime and design-time activities.

Provider Adapter

In the case of multi-Cloud application it is highly important that the implementation of various components remain unmodified to the specificities of different Cloud environments. The *provider adapter* pattern offers the means for a smooth transition of applications and components from one Cloud provider to another.

The *provider adapter* pattern is highly relevant in the context of multi-Cloud applications, and it has been applied to the MODACloudML supporting tools and extended to the language itself through the concept of Cloud provider-independent

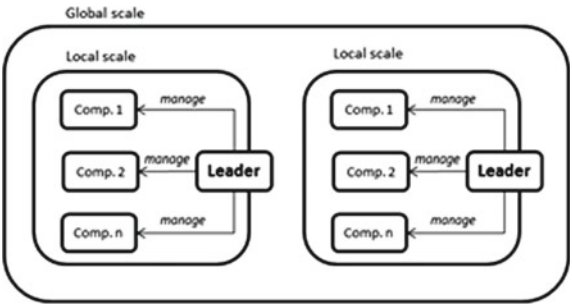

Fig. 11.2 The leader-followers pattern

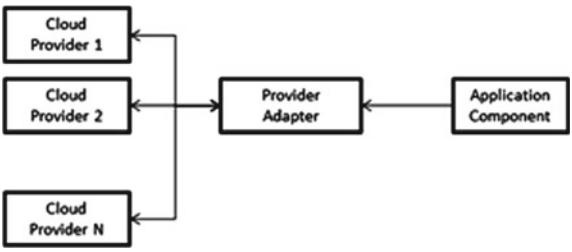

Fig. 11.3 The provider-adapter pattern

models that can be automatically or semi-automatically refined into Cloud provider-specific models.

This pattern is especially relevant when application components are not written for a specific single Cloud provider, and may move to or across other providers for maintenance reasons for instance (see also Fig. 11.3).

11.4 Conclusions

In this chapter we provided an overview of the set of guidances and patterns that have been defined or extended during the MODAClouds project on the basis of the experience gained in designing and managing multi-Cloud applications. All of them have been successfully applied during the project to support the design of both the MODAClouds tools and case studies. These patterns complement well the large set of existing pattern already available in the literature.

References

1. Fehling C, Leymann F, Retter R, Schupeck W, Arbitter P (2014) Cloud computing patterns—fundamentals to design, build, and manage cloud applications. Springer
2. AWS cloud design patterns. http://en.clouddesignpattern.org/index.php
3. Erl T, Cope R, Naserpour A (2015) Cloud computing design patterns. Prentice Hall/Pearson PTR. http://cloudpatterns.org/
4. Brandle C, Grose V, Hong MY, Imholz J, Kaggali P, Mantegazza M (2014) Cloud computing patterns of expertise. IBM RedPaper. http://www.redbooks.ibm.com/redpapers/pdfs/redp5040.pdf
5. Homer A, Sharp J, Brader L, Narumoto M, Swanson T (2014) Cloud design patterns: prescriptive architecture guidance for cloud applications (Microsoft patterns & practices). MSDN Library. https://msdn.microsoft.com/en-us/library/dn568099.aspx