Model-based Adaptation of Cloud Computing Applications

Christian Inzinger, Benjamin Satzger, Philipp Leitner, Waldemar Hummer and Schahram Dustdar
Distributed Systems Group, Vienna University of Technology,
Argentinierstrasse 8/184, A-1040 Vienna, Austria

Keywords: Cloud Computing, Model-based Adaptation.

Abstract: In this paper we propose a provider-managed, model-based adaptation approach for cloud computing applications, allowing customers to easily specify application behavior goals or adaptation rules. Delegating control over corrective actions to the cloud provider will pose advantages for both, customers and providers. Customers are relieved of effort and expertise requirements necessary to build sophisticated adaptation solutions, while providers can incorporate and analyze data from a multitude of customers to improve adaptation decisions. The envisioned approach will enable increased application performance, as well as cost savings for customers, whereas providers can manage their infrastructure more efficiently.

1 INTRODUCTION

The cloud computing paradigm has found widespread adoption throughout the last couple of years. The pay-per-use model of cloud computing proved to be convenient in many respects. It allows to cope with services of time-varying resource demand and peak loads. Cloud computing helps to avoid high initial investments in IT infrastructures, as resources can be dynamically provisioned even if the demand for a service is unknown in advance (Armbrust et al., 2010).

However, in order to benefit from the resource elasticity provided by cloud computing, applications need to be properly built. In this paper we argue that developers should be able to create models describing their application's adaptation capabilities together with adaptation goals defining their preferences. Cloud computing providers use this information to control and adapt the application according to the customers' objectives. Customers have only a limited view on the execution infrastructure, while cloud computing providers, when given access to necessary information from the application, have a complete view and can incorporate low-level infrastructure details to make adaptation more efficient and effective.

Models are already used for improving deployment of cloud computing applications. An example is Amazon's service called CloudFormation[1], providing a domain specific language (DSL) for defining the infrastructure required by an application, which can be provisioned in a single step. Research has brought forth sophisticated approaches for adapting application topologies and resource allocations, e.g., for latency requirements (Chang et al., 2012), power and cost considerations (Jung et al., 2010), or deadline-driven workflow scheduling (Kim et al., 2010), among other purposes. In practice, however, cloud providers currently do not provide mechanisms for further managing an application at runtime. We propose to use models allowing application developers to create "management hooks" for the cloud provider. This has the advantages that all applications can benefit from a sophisticated control mechanisms offered by the provider. Application developers use models to state the objectives for application control, relieving them from implementing complex adaptation schemes. Cloud providers have a deeper understanding of the application infrastructure, which they can use to improve application adaptation, but also to optimize resource usage. Also, providers can leverage the experiences and data from similar applications to improve control and adaptation over time.

In the next section we point out how models are used in cloud computing today. Then, in Section 3 we present our model-based adaptation approach. We wrap up in Section 4 and provide an outlook for future research and remaining challenges.

[1] http://aws.amazon.com/cloudformation

Inzinger C., Satzger B., Leitner P., Hummer W. and Dustdar S..
Model-based Adaptation of Cloud Computing Applications.
DOI: 10.5220/0004381803510355
In *Proceedings of the 1st International Conference on Model-Driven Engineering and Software Development* (MODA-2013), pages 351-355
ISBN: 978-989-8565-42-6

2 MODELS IN CLOUD COMPUTING

The main responsibilities of cloud application lifecycle management are infrastructure provisioning, application deployment, and finally control and adaptation at runtime.

Infrastructure provisioning involves setting up virtual machines, installing and configuring required software packages, and initializing cloud services, such as load balancers, database instances, persistent storage, or caches. Manual infrastructure provisioning is cumbersome and error-prone, and limits the level of reusability, e.g., to set up identical environments for development and testing. Furthermore, proper change management of infrastructure is not supported. Model-based approaches for infrastructure provisioning that have recently emerged help to overcome these limitations. Amazon's CloudFormation provides a service to create infrastructure templates used to create a collection of related infrastructure resources and provision them in an automated way. These models can be easily reused and shared using infrastructure template repositories, helping to establish and distribute best practices for different kinds of applications. The provisioned resources, e.g., virtual machines, still need to be properly configured to include all necessary application dependencies, as well as respective links between system components. To enable a predictable and repeatable process for this, deployment models are used to enable automated server configuration according to predefined specifications. Popular approaches for modeling software deployment include configuration management (CM) tools like Chef[2], Puppet[3], or cfengine[4]. Resource configuration is modeled using vendor-specific DSLs, specifying required software packages and libraries, as well as necessary configuration files and parameters to support the application to be deployed and all its components. The CM will attempt bring all managed resources into the state defined for its particular role by installing packages, deploying configuration files, and (re-)starting affected services.

Application deployment and redeployment deal with packaging and copying artifacts to according infrastructure resources, configuring them, establishing links between them, and finally making sure they are properly launched. Deployment models are used to ensure efficient and effective application rollouts and updates based on declarative specifications,

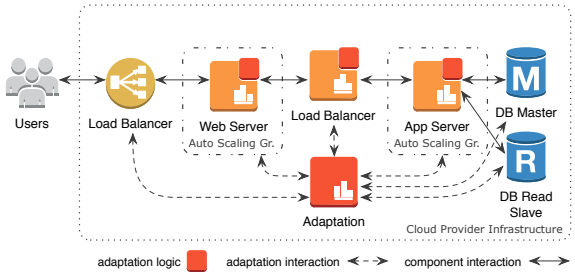

Figure 1: Traditional Cloud Application Architecture.

e.g. (Inzinger et al., 2012). Again, CM tools, as mentioned above, can be used to model application components, their dependencies, and required parameter settings.

While deployment models enable the predictable, testable and repeatable provisioning of runtime infrastructure, their responsibility stops at deployment time. The models are not designed to consider application specific runtime aspects or application behavior goals that should be reached. Cloud providers offer means to monitor basic infrastructure or application metrics, but adaptation is mostly limited to starting or stopping instances to cope with varying load patterns. Complex applications, however, have a variety of possibilities to cope with changes to the environment without necessarily resorting to scaling out available resources. Certain non-critical process steps could, for instance, be deferred to a later time if current load is too high, or service quality could be adapted according to environmental circumstances. Today's cloud providers, however, employ a black box model for applications deployed on their infrastructure, leaving customers responsible for realizing their own adaptation infrastructure to implement application-level reactions to changes in the environment. Future cloud service providers will offer means for integrating advanced application adaptation into their offerings using a model-based adaptation approach, as discussed in the next section.

3 A CASE FOR MODEL-BASED ADAPTATION

In this section, we present an approach for provider-assisted model-based application adaptation that will yield benefits for both providers and customers.

Figure 1 shows a traditional cloud application deployment for an exemplary web application. Incoming requests are distributed across several front end web servers using the provider's load balancer service. The web servers access the back end infrastruc-

[2]http://opscode.com/chef/
[3]http://puppetlabs.com
[4]http://cfengine.com

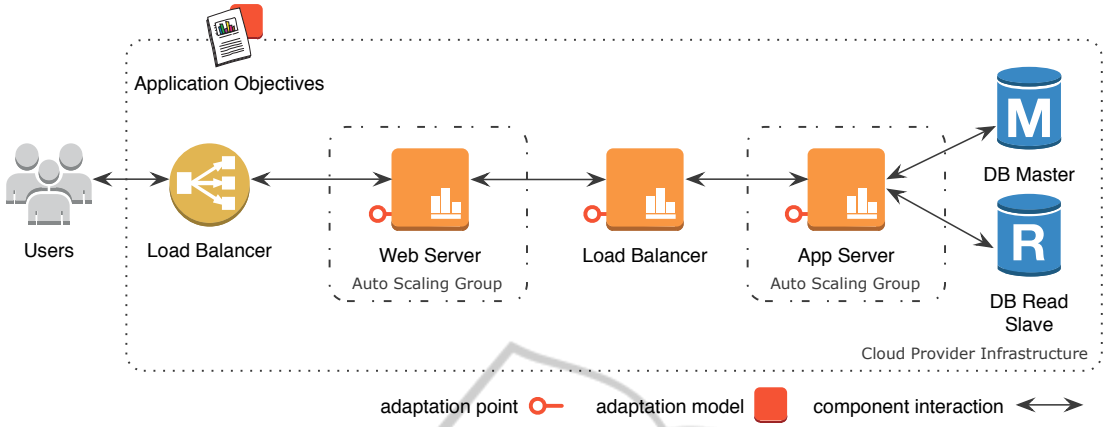

Figure 2: Cloud Application Architecture using Provider-Managed Adaptation.

ture using a custom middle tier load balancer, evenly distributing load on the available back end application servers. The application servers execute business logic and access the database cluster. Web and application server instances can be added or removed according to their current and predicted future load. The custom adaptation component controls service quality by performing corrective actions on all application services according to defined behavior goals, realized as autonomic control loop (Huebscher and McCann, 2008). It will, for instance, incrementally raise the fragment cache expiry time out up to a certain threshold on the web servers when an increase in request traffic is detected. Similarly, a recommendation module running on the app server instances will set to only return cached user group recommendations, or a list of the most purchased products, instead of personalized recommendations, in response to high order volumes. This will lower stress on the database, as well as the application servers and starting new instances is not immediately required. Analogously, when application load is low, the web servers will always serve fresh content and the recommendation module will always deliver personalized recommendations. Since these adaptation actions influence service quality and thus user experience, the customer will employ a utility function within the adaptation component to determine, when to change service quality and when to scale. Currently, cloud providers only offer support for automatic scaling of application components based on infrastructure and application metrics, but, as deployed applications are considered black boxes, do not provide means to alter component runtime behavior to react to changes in the environment. As indicated in the figure, customers need to deploy their own adaptation infrastructure, often scattered across deployed components.

We propose an approach for model-based cloud application adaptation. Customers specify emitted metrics, available adaptation points and desired objectives using provided models, which could be based on previous research such as (Zhang and Cheng, 2006), in order to make applications ready for provider-driven adaptation and control. This gives customers the ability to declaratively specify desired application behavior, while the provider has necessary information to take over control. Figure 2 shows the cloud application discussed above, deployed using the envisioned model-based provider-managed adaptation approach. A dedicated custom adaptation component is not needed anymore, as customers describe desired application behavior objectives in the adaptation model and delegate its execution to the cloud provider. This results in significant cost savings for the customer, as creating sophisticated and comprehensive adaptation solutions requires considerable effort and expertise. The business logic components do not contain any adaptation logic, but only provide designated adaptation points to allow for external management. Available adaptation points are referenced in the adaptation model for use in corrective actions. As a result, component design is simplified through the clear separation of adaptation capabilities and adaptation logic. As desired application behavior is declaratively specified in the application adaptation model, it can easily be reused or modified to accommodate changes without altering components. In the presented exemplary application, the customer specifies the following adaptation points: `SetFragmentCacheTimeout` to modify the fragment cache timeout in the web server component, and `SetRecommendationStrategy` to switch between personalized recommendations, cached customer group recommendations, or a list of the most sold products overall in the application server. Additionally, the `ScalePool` adaptation point supplied by

the cloud provider can be used to start and stop application instances. Monitoring metrics data required to take adaptation decisions are modeled with the application objectives, or a push-based approach similar to current provider-assisted scaling solutions such as the previously Amazon CloudWatch is used. The model allows for specification of conventional event condition action (ECA) rules to govern application behavior, so existing applications can easily migrate their current adaptation strategy. Alternatively, the customer can define a simple utility function to reflect application objectives according to desired characteristics, such as 'response time should not exceed 500ms', 'service quality should be as high as possible', and 'infrastructure cost should be as low as possible'. This is possible since adaptation points contain indications on how they will affect application behavior. Adaptation point `SetFragmentCacheTimeout`, for instance, will indicate that higher parameter values should decrease response time, but will also decrease service quality. These indications will initially be supplied by customers, but the cloud provider will monitor the effects of performed corrective actions and adjust adaptation point information during the runtime of the application. The provider-managed adaptation service uses the customer-supplied objectives and infers optimized actionable adaptation strategies. A cloud provider can not only use monitoring data from the application to be controlled, but also consider historical data from different but similar customers, leveraging time series analysis to better anticipate load patterns or even adaptation action effects for common components such as databases.

Allowing for provider-assisted application adaptation at runtime has a number of distinct advantages. Outsourcing some control over adaptation decisions will benefit customers. Providers can offer optimized adaptation decisions based on data from similar customers. Furthermore, providers are able to improve resource provisioning strategies and offer better service to customers.

By handing off control over the execution of adaptation actions, customers can benefit from the provider's expertise in reliability and resilience engineering. They are relieved of implementing their own complex adaptation mechanisms and can use an advanced, thoroughly tested solution offered by the cloud provider without investing large amounts of capital or personnel. Leveraging economies of scale, the cloud provider in turn can offer their adaptation solution to customers with competitive pricing strategies while retaining appropriate return on investment.

Using data from a multitude of customers, the cloud provider can make better adaptation decisions for customers, enabling higher levels of service, as well as cost savings for customers and the provider. By analyzing time series data of a large number of customers, the provider can anticipate, for instance, future load patterns much more accurately than a single customer could with their data alone. This allows for better adaptation decisions, as the provider can establish categories of customers, enabling more precise load predictions based on signals like traffic patterns and geographical distribution. Using this data, the provider can offer targeted adaptation decisions or recommendations for geographic placement of instances, or even migration of instances to more suitable locations.

Furthermore, the additional information available to the provider will enable highly optimized resource provisioning, keeping over-provisioning at even lower levels than with traditional elastic applications that can only analyze traffic patterns from their own past. Customers naturally benefit from lower resource usage by paying less, while the provider now has access to previously unused but, due to inefficient provisioning, inaccessible resources. Furthermore, the supplied adaptation goals can be used to react in different ways to varying environmental conditions. Depending on the utility, it might be more reasonable for an application to defer certain processing steps to a later time, thus reducing load on application instances, before scaling out by adding new machines. Similarly, when request traffic declines it might be beneficial to process queued tasks on the now underused instances – at least until their current billing interval is over – before terminating them to reduce infrastructure costs. The customer-provided utility functions guide the adaptation decisions considering application performance, service quality, as well as infrastructure costs. While the presented approach is ideally deployed by the cloud provider, a transitional implementation could also be realized as part of a cloud abstraction layer like the meta cloud (Satzger et al., 2013), allowing the use of a managed adaptation solution without explicit cloud provider support.

4 CONCLUSIONS

In this paper we presented a novel approach for provider-managed, model-based adaptation of cloud computing applications. Customers are not required to implement their own adaptation solution, but can use a simple model to specify desired application behavior. The presented approach has the potential for significant cost savings and increased application quality for customers by utilizing a sophisti-

cated adaptation infrastructure managed by the cloud provider that can offer better adaptation decisions by considering data from multiple different, but similar, customers. Furthermore, cloud providers will be able to manage their infrastructure more efficiently due to reduced resource over-provisioning resulting from improved adaptation decisions.

Deferring control over adaptation decisions to a cloud provider poses several challenges that need to be addressed. Our approach requires that customers trust providers with potentially confidential information about their applications' inner structure, such as adaptation strategies and request traffic patterns. The widespread adoption of cloud computing has shown that customers already trust reputable providers with hosting their applications, so we argue that the addition of adaptation strategies will not be problematic, provided that service contracts clearly state how provided data will be used. Customers that do not want their data used to improve adaptation decisions for others could still benefit of the provider's adaptation infrastructure, saving them implementation effort, but will in turn not be able to receive adaptation decisions optimized using data from others. Furthermore, service level agreements need to explicitly state the new DSLs and tools needed, as well as responsibilities of both customers and providers when using managed adaptation infrastructure.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Commission's Seventh Framework Program [FP7/2007-2013] under grant agreement 257483 (Indenica), as well as from the Austrian Science Fund (FWF) under grant P23313-N23 (Audit 4 SOAs).

REFERENCES

Armbrust, M., Stoica, I., Zaharia, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., and Rabkin, A. (2010). A View of Cloud Computing. *Communications of the ACM*, 53(4):50–58.

Chang, F., Viswanathan, R., and Wood, T. L. (2012). Placement in Clouds for Application-Level Latency Requirements. In *5th IEEE International Conference on Cloud Computing (CLOUD)*, pages 327–335.

Huebscher, M. C. and McCann, J. A. (2008). A survey of autonomic computing—degrees, models, and applications. *ACM Computing Surveys*, 40(3):7:1–7:28.

Inzinger, C., Satzger, B., Hummer, W., and Dustdar, S. (2012). Specification and deployment of distributed monitoring and adaptation infrastructures. In *Workshop on Performance Assessment and Auditing in Service Computing (PAASC) held in conjunction with ICSOC*.

Jung, G., Hiltunen, M., Joshi, K., Schlichting, R., and Pu, C. (2010). Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures. In *30th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 62–73.

Kim, H., el Khamra, Y., Jha, S., and Parashar, M. (2010). Exploring application and infrastructure adaptation on hybrid grid-cloud infrastructure. In *19th ACM International Symposium on High Performance Distributed Computing (HPDC)*, pages 402–412.

Satzger, B., Hummer, W., Inzinger, C., and Dustdar, S. (2013). Winds of Change: From Vendor Lock-In to the Meta Cloud. *Internet Computing*, 17(1).

Zhang, J. and Cheng, B. H. C. (2006). Model-based development of dynamically adaptive software. In *28th International Conference on Software engineering (ICSE)*, pages 371–380.