Software Architecture for the Cloud – A Roadmap Towards Control-Theoretic, Model-Based Cloud Architecture

Claus Pahl[1](✉) and Pooyan Jamshidi[2]

[1] IC4 & Lero, School of Computing, Dublin City University, Dubin, Ireland
Claus.Pahl@dcu.ie
[2] Department of Computing, Imperial College London, London, UK

Abstract. The cloud is a distributed architecture providing resources as tiered services. Through the principles of service-orientation and generally provided using virtualisation, the deployment and provisioning of applications can be managed dynamically, resulting in cloud platforms and applications as interdependent adaptive systems. Dynamically adaptive systems require a representation of requirements as dynamically manageable models, enacted through a controller implementing a feedback look based on a control-theoretic framework. We argue that a control theory and model-based architectural framework for the cloud is needed. While some critical aspects such as uncertainty have already been taken into account, what has not been accounted for are challenges resulting from the cloud architecture as a multi-tiered, distributed environment. We identify challenges and define a framework that aims at a better understanding and a roadmap towards control-theoretic, model-based cloud architecture – driven by software architecture concerns.

Keywords: Cloud computing · Control theory · Adaptive system · Software architecture · Microservice · Model-based controller · Uncertainty

1 Introduction

Adapting systems to changing requirements is often a necessity to guarantee on-going correct and satisfying performance. Self-adaptive systems are systems that are able to adjust their behaviour in response to their perception of the environment and the system itself [3]. The software engineering community has approached this from the requirements engineering perspective [11], but has recognised the need for software architecture to play a major role in a solution.

Requirements need to have a representation at runtime to allow self-adaptive systems to interact with the environment, i.e., reflect this through models that also link in the decision-making process necessary to change the underlying system itself [1,2,6]. Dynamically adaptive systems require a representation of requirements as dynamically manageable models, enacted through a controller implementing a feedback look based on a control-theoretic framework [5].

© Springer International Publishing Switzerland 2015
D. Weyns et al. (Eds.): ECSA 2015, LNCS 9278, pp. 212–220, 2015.
DOI: 10.1007/978-3-319-23727-5_17

The cloud is moving towards a distributed, often federated architecture of many individual cloud services [10], providing resources as services in a tiered fashion. The configuration, deployment and provisioning of application architectures can be managed dynamically as a response to changes in requirements and changes in the execution platform environment, resulting in cloud platforms and the applications in them as interdependent adaptive systems. Microservices are emerging as a new architectural style, aiming at realising software systems as a package of small services, each deployable on a different platform. These run in their own process while communicating through lightweight mechanisms without any centralized control[1]. We argue that a cloud-specific control-theoretic, model-based architectural framework is needed. While critical aspects such as uncertainty have been investigated [4,8,9] for the cloud, what has not been accounted for are the challenges resulting from the cloud architecture as a multi-tiered, distributed environment for increasingly fragmented application architectures.

We identify the challenges and define a conceptual framework. The target is a roadmap towards control-theoretic, model-based cloud architecture in which software architecture concerns play the central role.

2 Cloud Architecture – Definition and Scenario

Our view on cloud systems from an architectural perspective addresses the key shortcomings of the current discussion of control-theoretic approaches to adaptive systems, and cloud in particular. We will also argue for a model-based approach to controller definition later on as well. The cloud allows the distributed, tiered deployment of software. The underlying architecture links infrastructure and platform providers with the software applications running in them. Software is usually logically architected in a layered format, but in the cloud mapped onto (virtualised) physical tiers.

- Logical layers organise code. Typical layers include presentation, business logic and data management and storage. However, this does not imply that the layers run on different computers or in different processes.
- Physical tiers are about the location of the application execution. Tiers are places where layers are deployed and where layers run.

The cloud services provided as infrastructure-as-a-services (IaaS), platform-as-a-service (PaaS) or software-as-a-service (SaaS) realise these tiers, albeit in a virtualised form accessed through services.

A further complication arises through clouds as distributed, often federated systems, even if providing the same or similar services, will operate differently. Interaction between the layers, but also horizontally is possible and necessary, which we capture in the following architectural scenario in Figure 1.

Let us illustrate a common problem. An infrastructure server might have the capacity to deal with 100 user applications at the same time, but the workload

[1] http://martinfowler.com/articles/microservices.html.

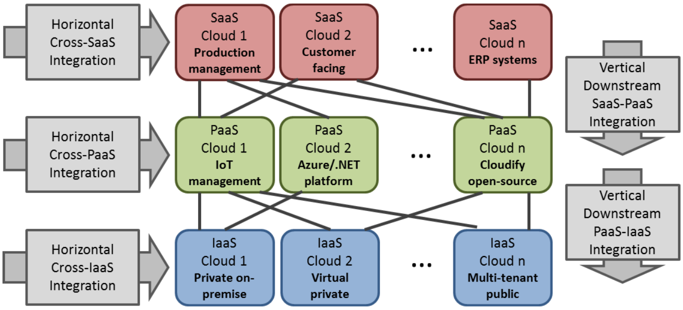
Horizontal Cross-SaaS Integration
SaaS Cloud 1 Production management
SaaS Cloud 2 Customer facing
...
SaaS Cloud n ERP systems
Vertical Downstream SaaS-PaaS Integration
Horizontal Cross-PaaS Integration
PaaS Cloud 1 IoT management
PaaS Cloud 2 Azure/.NET platform
...
PaaS Cloud n Cloudify open-source
Vertical Downstream PaaS-IaaS Integration
Horizontal Cross-IaaS Integration
IaaS Cloud 1 Private on-premise
IaaS Cloud 2 Virtual private
...
IaaS Cloud n Multi-tenant public

Fig. 1. Tiered and Distributed Cloud Architecture.

might temporarily reduce significantly. Load balancing would allow the system architecture to be adapted and applications relocated to one server, thus scaling down the deployment of servers. Here, the system reacts to external factors – the reduced load – and adapts the configuration to reduce the costs (a non-functional requirement) while still maintaining adequate performance (also a non-functional requirement). Two observations emerge.

- Workload and QoS dictate the adaptation. Cost and quality as drivers for decisions – i.e., decisions are made based on non-functional requirements.
- In the cloud as a tiered architecture, where user applications might run on third-party provided infrastructure servers. Factors that influence here down-scaling as the adaptation include (i) application performance at the user tier/layer and (ii) system workload at the infrastructure tier/layer.

Other scenarios here could involve changing non-functional requirements rather than changing environment factors. The performance requirement might need to be tightened, resulting in an up-scaling of the infrastructure.

Recently, microservice architectures have been discussed, which aim to break up application architectures into independently deployable services that can be rapidly deployed to any infrastructure resource as required. Microservices are independently deployable, usually supported by a fully automated deployment and orchestration framework. They require the ability to deploy often and independently at arbitrary schedules, instead of requiring synchronized deployments at fixed times. The microservice deployment and orchestration across the vertical and horizontal dimensions of the cloud are central architecture concerns. Clouds provide a management tool for their flexible deployment schedules and provisioning orchestration needs, particularly, if these are to be PaaS-provisioned.

3 Dynamic Requirements and Models

As the example above has indicated, both requirements (the user-facing tiers) and the platform (the infrastructure-facing tiers) can change dynamically. What is needed first is a review of modelling concerns for this context. Drivers of change are often requirements to maintain quality-of-service at the user end to maintain within the limits (non-functional requirements) possible stated in a service-level agreement. In [3], a number of model dimensions are identified that help to frame the adaptivity problem:

- Goals as system objectives: evolution, flexibility, multiplicity, dependency
- Change captures causes of adaptation: source, type, frequency, anticipation.
- Mechanism implements adaptation: type, autonomy, organisation, scope, duration, timeliness, triggering
- Effects define adaptation impact: criticality, predictability, overhead, resilience

A challenges here is the mapping of requirements to the underlying architecture. The solution is a control loop, based on control-theoretic foundations [11], but importantly, the layering of the application architecture onto the tiered cloud. The run-time representation of requirements in the form of application requirements and cloud infrastructure models needs to provide model manipulation and access features to allow introspection and reasoning about these models [1,6].

A specific challenge is the uncertainty that arises in the interaction between models and the system architecture – the latter possibly at different tiers/layer, all interacting with one another along their interfaces, cf. Figure 1. Respective models that capture uncertainty and can map this as actions within the control loop are needed. The models themselves need to reflect the adaptation approach, requiring to capture the non-functional properties, but more significantly allow prediction and reasoning to take place in an environment prone to uncertainty.

4 Measurement, Prediction and Uncertainty

The state of a system is characterised by a range of non-functional properties that need to be aligned with non-functional requirements. Due to the layering, mapping and managing these across layers, but also within one layer, is challenging. In general, we need to measure at different layers and map between the different tiers in the cloud.

- The upper level represents the application service-level qualities.
- The lower level are the loads of infrastructure resources that run the service.

Furthermore, there is a mapping of the infrastructure loads into a cost model – which can of course be a major driver of adaptation decisions.

Measurement and Uncertainty. Ideally, system state attributes can be reliably measured. However, the cloud adds a high degree of uncertainty here [11]:

- Uncertainty Level 1: general confidence about the shape of the future, but some key variables do not have precise values.

- Uncertainty Level 2: there are a variety of possible future scenarios, that can be listed and are mutually exclusive and exhaustive.
- Uncertainty Level 3: it is feasible to construct future scenarios, but these are mere possibilities and are unlikely to be exhaustive.
- Uncertainty Level 4: it is not even possible to frame possible future scenarios.

Uncertainty emerges from various sources in cloud systems – as uncertainty from different interpretations and decisions in the adaptation definition process or as uncertainty arising from possible different, distributed monitoring systems resulting in partially unreliable and incomplete data [9]:

- Uncertainty in Adaptation Definition. Adaptation policies need a careful determination of thresholds. This relies on a users knowledge of system behaviour and how resources are managed. Therefore, the accuracy of policies remains subjective, making the effect of adaptations prone to uncertainty. Unpredictable changes in environment or application demand may require adaptation models to be continuously re-evaluated and revised.
- Uncertainty in Dynamic Resource Provisioning. Acquiring and releasing virtual resources in the cloud is not instantaneous. A cloud controller uses the platform services to initiate the acquisition process and has to wait until resources are available. During this time, which may take minutes for VMs, the cloud application is vulnerable to workload increases, causing uncertainty.
- Uncertainty in Monitoring Data. The cloud controller needs to continuously monitor the state of the application as well as of the resources in which the application is deployed in order to timely react to load variations. Monitoring involves a distribution of data collected by measurement-specific probes or sensors, which are not immune to measurement deviations (so-called sensory noise). This sensory noise is another source of uncertainty, as it results in oscillations that may affect how the controller allocates resources.

Formal Models for Uncertainty. Models captures the state, its behaviour and the adaptation rules. Models of different types can reflect how we deal with uncertainty in dynamic systems. The dynamics of a system are often based on state models, describing sequences of possible actions as a protocol. In [1], a Markovian model is used (DTMC – Discrete Time Markov Chains; alternatives could include continuous time models), formalising specific properties in logics such as a probabilistic logics [2] to reason in uncertain spaces – in an uncertain space, the probability of the next state is included in the model.

Others propose fuzzy logic [9], where fuzziness is expressed as a varying, non-binary truth value. This allows the uncertainty of a system situation to be expressed through a membership functions on fuzzy sets. For instance, a fuzzification of adaptation rules [9] can be done. As an example, qualitative values for infrastructure workload and service performance (such as 'very low' or 'very high' for workload) are presented as membership functions in a fuzzy set model, resulting in smoother controller responses.

Analysis and Prediction – Cross-Tier Mapping and Uncertainty. Unreliable or incomplete data causes uncertainty, which can be alleviated to some extent by prediction. Furthermore, the delay in providing resources, as discussed above, also makes prediction a suitable approach. Two aspects emerge:

- Analysing measure system data allows us to predict behaviour, reducing uncertainty and increasing the robustness of the adaptation.
- Prediction also helps to link the layers and tiers in the architecture, as for instance infrastructure tier metrics can be used to predict service-level quality. Prediction captures dependencies and becomes a link between the tiers.

Through predication and analysis of monitored data, we can e.g. identify stable quality utilisation patterns. We can map infrastructure workload patterns for CPU, storage and network utilisation at the infrastructure tier to service-level performance patterns, thus linking models (here pattern-based) across tiers [12].

We can implement a prediction technique for the same workload and performance prediction context, based on simple and double exponential smoothing to smoothen outliers and to anticipate trends. Here the aim is the robustness of the prediction and overall adaptation process (by looking ahead in vulnerable moments when the system is about to change).

5 Control Theory and Controller Architecture

Control theory and control engineering can be applied to build self-adaptive systems. Control theory can help to build the models and the reasoning about them to inform the decision making [3]. Decision making is a multi-objective process [11]. Constructing a utility function that involves all stakeholders (such as end-users and the providers of the various tiers of the system in question) is a challenging task [7]. This utility function is implemented by the cloud controller. This construction of utility function (the model) and the controller is a process involving the following steps [5]: identify goals, identify knobs (measure), devise model and design controller, complemented by validation and verification steps.

A key property of this controller is robustness. Robustness tells how resilient the controller is against noise and uncertainty. Prediction, as discussed above, is in addition to a proper calibration of the model a contributor to robustness. Prediction across layers has already addressed the challenges arising from the tiered cloud. architecture. Techniques such as horizontal scaling can deal with the distribution dimension at each tier.

All concerns need to be managed by a control loop. Often, the MAPE-K model is utilised [1], cf. Fig. 2, as the structure of a controller: *M*onitor application and environment (in control-theoretic terms disturbances such as workload). *A*nalyse the input data and detect any possible violation. *P*lan corrective actions in terms of adding resources or removing existing unutilized ones. *E*xecute the plan according to a specific platform. Uitilise a shared *K*nowledge (model).

It is the task of the controller to synchronise models with run-time architecture [11]. The model part of the controller needs to be implemented and

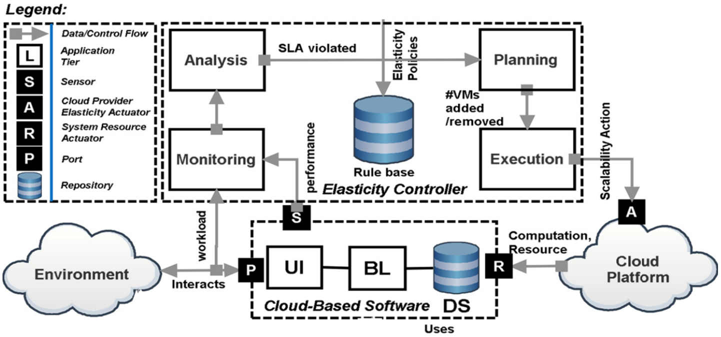

Fig. 2. MAPE-K Control Loop for the Cloud.

integrated with the cloud architecture in order to allow a model-driven cloud control of non-functional aspects [5].

Controller construction still faces a number of problems [4], including uncertainty, synthesize controllers, heterogeneity, unpredictable workloads, resource bottlenecks, multi-tier applications, multi-cloud resources and scalability. We have already discussed uncertainty and unpredictability. The last few points indicate the importance of the cloud as a problem from the software architecture perspective, i.e., an architecture onto which the concerns need to be projected:

- Measurement: the controller integrates different models representing the application and infrastructure models at the different tiers – vertical dimension.
- Actuating/executing: typically within a tier, but across services, e.g., based on scalability actions as adaptations – the horizontal dimension.

Uncertainty [5,8] could also be addressed by reducing the dependency on human stakeholders. Here, machine learning can serve to learn adaptation rules rather than relying on uncertain, possibly erroneous or inconsistent user input. Again, the software architecture perspective can clarify this. A suitable architecture would add a meta-model layer on top of the MAPE-K control loop, representing the learning loop on the models. Models can provide prediction and the feedback loop can correct it, e.g., a queuing model provide how much resources are needed to guarantee an SLA. Since the model is not precise, then it can be augmented with a feedback to correct the error, called feedforwarding.

6 Conclusion

The cloud is a distributed, multi-tiered platform onto which layered, modular software application architectures are mapped. The virtualisation of the cloud resources causes this to be an adaptive system, that is, however, subject to uncertainty and other challenges. Our contribution is the discussion from a software architecture perspective and to propose a roadmap towards a model-based control-theoretic solution that defines some core contributors to future solutions:

- models for uncertainty, allowing prediction and enforcing robustness in a control-theoretic framework,
- a model-driven multi-tier cloud controller to manage layered built from easily deployable microservices,
- adapting the architectural configuration in the cloud, but also re-architecting the application for the cloud.

There is a need for a controller framework that addresses the layered architecture of an application mapped onto tiered cloud resource services through a set of linked models for robust control-theoretic uncertainty management. Challenges for this framework include big data and real-time analytics for the a dynamic adaptation as well as stream processing.

Acknowledgments. This work was supported by Science Foundation Ireland grant 13/RC/2094 to Lero (www.lero.ie) and by the Irish Centre for Cloud Computing and Commerce (IC4), a Technology Centre funded by Enterprise Ireland and the IDA.

References

1. Baresi, L., Ghezzi, C.: A journey through smscom: self-managing situational computing. Computer Science - Research and Development **28**(4), 267–277 (2013)
2. Chan, K., Poernomo, I.H., Schmidt, H., Jayaputera, J.: A model-oriented framework for runtime monitoring of nonfunctional properties. In: Reussner, R., Mayer, J., Stafford, J.A., Overhage, S., Becker, S., Schroeder, P.J. (eds.) QoSA 2005 and SOQUA 2005. LNCS, vol. 3712, pp. 38–52. Springer, Heidelberg (2005)
3. de Lemos, R., Giese, H., Müller, H.A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., et al.: Software engineering for self-adaptive systems: a second research roadmap. In: de Lemos, R., Giese, H., Müller, H.A., Shaw, M. (eds.) Software Engineering for Self-Adaptive Systems. LNCS, vol. 7475, pp. 1–32. Springer, Heidelberg (2013)
4. Farokhi, S., Jamshidi, P., Brandic, I., Elmroth, E.: Self-adaptation challenges for cloud-based applications: a control theoretic perspective. In: 10th International Workshop on Feedback Computing 2015 (2015)
5. Filieri, A., Maggio, M., Angelopoulos, K., D'Ippolito, N., Gerostathopoulos, I., Hempel, A., Hoffmann, H., Jamshidi, P., Kalyvianaki, E., Klein, C., Krikava, F., Misailovic, S., Papadopoulos, A., Ray, S., Sharifloo, A., Shevtsov, S., Ujma, M., Vogel, T.: Software engineering meets control theory. In: Intl Symposium on Software Engineering for Adaptive and Self-Managing Systems SEAMS 2015 (2015)
6. Ghezzi, C., Pinto, L., Spoletini, P., Tamburrelli, G.: Managing non-functional uncertainty via model-driven adaptivity. In: Inl. Conf. on Soft. Eng. (2013)
7. van Hoorn, A., Rohr, M., Gul, A., Hasselbring, W.: An adaptation framework enabling resource-efficient operation of software systems. In: Proceedings of the Warm Up Workshop for ACM/IEEE ICSE 2010, WUP 2009. ACM (2009)
8. Iftikhar, M., Weyns, D.: Assuring system goals under uncertainty with active formal models of self-adaptation. In: Companion Proceedings of the 36th International Conference on Software Engineering. ACM (2014)
9. Jamshidi, P., Ahmad, A., Pahl, C.: Autonomic resource provisioning for cloud-based software. In: Intl. Symp. on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2014 (2014)

10. Pahl, C.: Containers and clusters for edge cloud architectures - a technology review. In: Intl. Conference on Future Internet of Things and Cloud, FiCloud 2015 (2015)
11. Sawyer, P., Bencomo, N., Whittle, J., Letier, E., Finkelstein, A.: Requirements-aware systems: a research agenda for re for self-adaptive systems. In: International Requirements Engineering Conference, RE 2010, pp. 95–103 (2010)
12. Zhang, L., Zhang, Y., Jamshidi, P., Xu, L., Pahl, C.: Workload patterns for quality-driven dynamic cloud service configuration and auto-scaling. In: International Conference on Utility and Cloud Computing, UCC 2014 (2014)