Infrastructure for Policy and Model-based Reconfiguration of Cloud Systems

Thesis Proposal

Samar G. Sajnani

Computer Science Thesis (CS4490Z)

Department of Computer Sceince

Western University

11/16/2018

Supervisor: Dr. Kostas Kontogiannis

Course Instructor: Dr. Nazim Madhavji

1. Synopsis

In 1998, a paper demonstrated that a distributed system could use a computational method to generate dynamic reconfiguration based on dynamic change. Ever since there have been a slew of papers that demonstrate the study of dynamic reconfiguration using various models. Of these models, the metamodel framework maintains the structure of the system as a non-physical state. Changes can be made to this state through transformations. However, most of the metamodel based dynamic reconfiguration systems do not consider an automated trigger. An automated trigger is important because it does not rely on individuals but rather the system itself. These triggers can be seen in networking configurations and in Amazon Web Services elastic cloud computing.

2. Background

The concept of dynamic reconfiguration for cloud systems is not a novel idea. The implementations of dynamic reconfiguration have been studied in software architectures, graph transformation, software adaptation, metamodeling, and reconfiguration patterns. This thesis project will focus on metamodel-based reconfiguration because metamodel provide an accurate software representation of a cloud system that can be used to directly automate code generation (Zuniga-Prieto et. al, 2018).

2.1. Analysing Dynamic Change in Distributed Software Architectures

One of the first examples of dynamic configuration based on dynamic change was shown in a 1998 paper by Jeff Kramer and Jeff Magee. The paper described the use of a four step model for creating a dynamic change in distributed software system. The steps were composed of identifying elements of an arbitrary state, converting them to a passive state, performing the changes needed, and activating the elements from their passive state. The changes could include operations such as disconnecting, deleting, creating, and connecting. This paper showed a rudimentary theoretical method of distributed software reconfiguration, from which new research has spawned (Kramer & Magee, 1998).

2.2. Dynamic Reconfiguration and Metamodels

The focus of dynamic reconfiguration has been based on formal proofs and a general lack of empirical studies. Out of the empirical studies only a select few are related to cloud systems. In 2016 Prieto et al. demonstrated empirical evidence for the generation of a cloud service metamodel which was shown to be used by developers to specify a new or updated service and generate the reconfiguration scripts that carry out the service change (Zuniga-Prieto et. al, 2018). The important thing to note is that automation of dynamic reconfiguration is defined by scripts, however, the trigger of the dynamic change is the user. The user must provide a service or a service file to specify a change. This system does not cover cases where the dynamic change is triggered by the violation of a policy.

2.3. Importance of Policy-Action Systems

The use of actions and policies for dynamic reconfiguration can be seen in empirical studies and patents. Specifically, the use of policies for autoscaling in Amazon Web Services is rigorously tested for performance and accuracy (Evangelidis et. al, 2018). Also it is shown in a patent from 2015 that the use of actions and policies is important for dynamic reconfiguration of multi-user systems is crucial for autoscaling cloud networks depending on the size of the user base (Cheng-Ta & Williams, 2018).

2.4. Policy and Model-based Dynamic Reconfiguration of Cloud Systems

The combination of metamodel-based dynamic reconfiguration and policy-action systems is the focus of this thesis project. Such a system will allow for automation of the trigger and the dynamic reconfiguration process, creating a system with increased resilience to state changes.

3. Detailed Proposal

3.1. Research Objective

The goal of this thesis project is to empirically study the infrastructure for achieving automated adaptation and automated reconfiguration of cloud systems using a policy-action system and a metamodel framework, respectively. This goal is composed of four research objectives. The first being the identification of an appropriate metamodel and generation of a model-based representation of a cloud system. Finding a model to describe common Kubernetes-based cloud systems is important for research because non-physical models can then be used to predict actions in a cloud-based system. Furthermore, models are important in practice because of their abstraction of cloud elements. Second, gathering real-time data from the cloud-system, such as metrics and logs must be automated. This is significant from a research standpoint because it may be important to generate a given standard for logs and metrics that will trigger an appropriate action from a violated policy. In practice, it is important to show that logs and metrics can provide important information about the system state.

The next objective is to create policies and actions that will relate the logs and metrics data to actions that can be performed on a cloud-system model. This is important for research because it shows a connection between policy-action and metamodel dynamic reconfiguration. In practice, users can define their own policies and actions depending on the elements of their cloud system. Lastly, the actions on the cloud model need to be scripted as reconfiguration steps which can be executed with a cloud automation tool. In practice, this will be useful for generating an automation framework that can automate all the different actions of a cloud system.

The use of a novel policy-action and metamodel cloud reconfiguration system is hypothesized to reduce the development time needed for dynamic reconfiguration through automation.

3.2. Research Plan

The main technical issue is that the system that will be developed is a distributed system. As a result, if there are multiple nodes (computing units), each of them will have their own storage and their own services. To synchronize the information and produce a policy-action system it may have to be localized on a single node of the system. Another technical issue will be use of a server called the observer that will observe an action and enact a reconfiguration. Once the reconfiguration is complete it will update the system model. The role of the observer is non-trivial.

The system architecture will be composed of various technologies and languages. Docker will be used to containerize cloud services. Kubernetes will be used to orchestrate the Docker services. A log driver such as ElasticSearch can be used to gather Kubernetes system logs and a metrics monitoring application like Prometheus can be used to gather system metrics. The language of Eclipse Modelling Framework (EMF) will be needed because of its use in modelling software systems. The Java programming language will be used to create classes for the cloud elements such as nodes, containers, pods, and services. A cloud configuration automation tool such as Terraform, or Ansible, will be used to script reconfiguration actions.

The main threat of the project relates to the complexity of creating a policy-action based dynamic reconfiguration system. A lot of planning will be involved in generating such a system. As such breaking the system into its separate components as was shown in the objectives will be useful in mitigating this threat. Creating a timeline and organizing tasks will be important. Task organizers, such as JIRA or Github, will be useful in documenting the progress made over time. Finally, storing the source code and working on the task incrementally by using Git and Github will the source code management needed.

Another threat is that a similar paper may be published during the preparation of this thesis defense. This would in effect render the project as a non-novel project and may affect my thesis defense, significantly.

3.3. Research Methodology

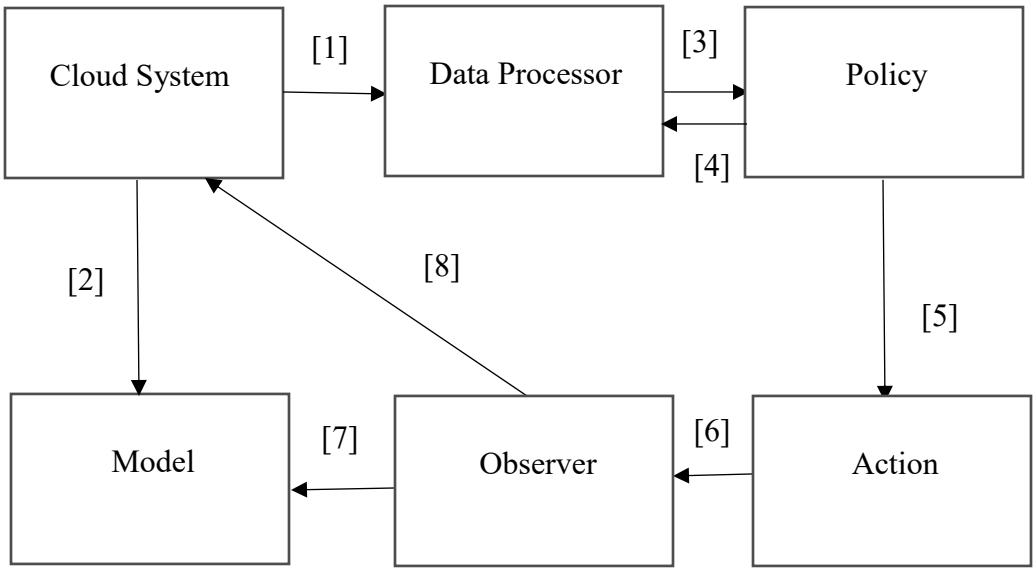

Figure 1 – Infrastructure of the Policy-Action Metamodel Reconfiguration System

The experimental setup can be described as follows. The cloud system is a Kubernetes and Docker based architecture. A unique model is generated for the cloud system using a metamodel and a cloud system configuration file as demonstrated in link [2]. The data processor will most likely be a cloud service. It will collect log and metric data from the cloud system as shown by link [1] in Figure 1.

The policy evaluates the conditions of the data that are processed by the data processor. Link [3] shows the interaction of the data with the policy. If the data violates the policy then link [5] is followed which results in an action. Otherwise the policy holds and more data is processed as per link [4].

An action is generated from the policy if the policy is violated. An action triggers the observer to generate the appropriate scripts to execute reconfiguration steps on the cloud system given by link [8]. The observer will also update the model once the reconfiguration steps are complete as shown by link [7].

4. Value of the Results and Industrial Relevance

The combination of policy-action triggers with the automation of metamodel dynamic reconfiguration in a cloud system is novel. In the research field, there is a general lack of empirical studies that demonstrate the principle of dynamic reconfiguration. As a result, it is crucial to generate and study such systems to produce more empirical evidence of their usage. Furthermore, providing a framework that can automate both the trigger and reconfiguration

phases is useful is specific cases. Lastly, it is important to study dynamic reconfigurations using metamodels because metamodels are an abstraction of a physical system. As such, to better understand, the physical system it is important to evaluate the different conditions of the system.

As for industrial practices, such a system could be used to reduce overall costs and increase efficiency (Verma et. al, 2010). This is because without dynamic reconfigurations, use of manual reconfigurations involve a significant amount of overhead and they involve having to follow a very specific procedure. With dynamic reconfiguration, a more abstracted procedure is discovered that can be modified and used based on the programmer's requirements.

References

Cheng-Ta, L., & Williams R.B. (2018). *United States Patent No. US 9942273B2.* Armonk, New York: International Business Machines Corp.

Evangelidis, A., Parker, D., & Bahsoon, R. (2018). Performance modelling and verification of cloud-base auto-scaling policies. *Future Generation Computer Systems, 87,* 629-638. doi: 10.1016/j.future.2017.12.047

Kramer, J., & Magee, J. (1998). Analysing dynamic change in distributed software architectures. *IEE Proceedings – Software, 145(5),* 146-154. doi: 10.1049/ip-sen:19982297

Verma, A., Kumar, G., & Koller, R. (2010). The cost of reconfiguration in a cloud. *Proceedings of the 11th International Middleware Conference,* 11-16. doi: 10.1145/1891719.1891721

Zúñiga-Prieto, M., González-Huerta, J., Insfran, E., & Abrahão, S. (2018). Dynamic reconfiguration of cloud application architectures. *Software: Practice and Experience, 48,* 327-344. doi: 10.1002/spe.2457