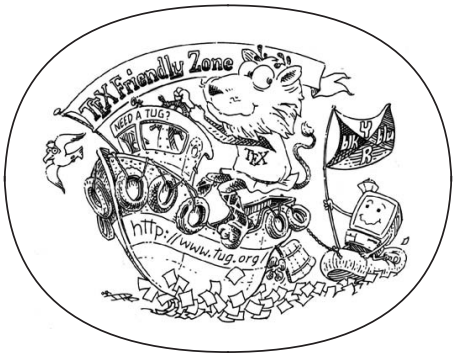
ERIC RWEMIGABO

AUTONOMIC APPLICATION TOPOLOGY REDISTRIBUTION

[August 16, 2018 at 11:55 – classicthesis version 1.0]

AUTONOMIC APPLICATION TOPOLOGY REDISTRIBUTION

ERIC RWEMIGABO



Autonomic Computing

MSc Computing Science (Software Engineering and Distributed Systems)
Computer Science
Faculty of Science and Engineering
University of Groningen

May 2018

[August 16, 2018 at 11:55 – classicthesis version 1.0]

Eric Rwemigabo: *Autonomic application topology redistribution,* Autonomic Computing, MSc Computing Science (Software Engineering and Distributed Systems), © May 2018

SUPERVISORS:
Vasilios Andrikopoulos
Mircea Lungu

LOCATION:
Netherlands

TIME FRAME:
December 2017 – May 2018

Ohana means family.
Family means nobody gets left behind, or forgotten.

— Lilo & Stitch

Dedicated to the loving memory of Abdul-Malik N E Umaru.
1992 – 2017

ABSTRACT

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— knuth:1974 [knuth:1974]

ACKNOWLEDGMENTS

Put your acknowledgments here.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

1 INTRODUCTION

After years of advancement in Cloud Computing, including a significant increase in the number of Cloud service providers within a short period of time, application developers and enterprises have been left with a wide range of choice to fill their need for cloud services. Therefore, with this wide range of choices, one may find themselves making a choice of using a service that is cheaper on paper but ends up costing them more in the long run.

Optimisation refers to the minimisation of allocated resources conditional to keeping the quality of a service at an acceptable level [11]. If the use of the resources one is paying cheaply for isn't optimised, it will end up costing them in the long run through various ways like affecting the performance of their services, over provisioning certain services to mention but a few.

1.1 PROBLEM STATEMENT

One argument for the use of Cloud Computing(CC) from an enterprise perspective is it makes it easier for enterprises to scale their services, which are increasingly reliant on accurate information according to client demand [5]. Given this aspect, we can then go further and conclude that the optimal scaling of their services would be important to them because they would want to get the best utilisation of these services in order to get the most out of their money especially for a start up with a limited budget that would go with cloud computing as a cheaper and more flexible option. In order to achieve the optimal utilisation of resources provided by CC services, the enterprise has to have a way to monitor and analyse how their services use the allocated resources and then make changes if necessary.

A number of projects have been taken on to work on the optimisation of systems with different architectural structures and using different approaches to implement their automation systems. However, many of these systems tackle one or two areas which the system I have developed for this project covers. In this project, I design and develop a component-based architecture, which uses MAPE-K loops introduced and explained in a following chapter to perform the optimisation of the cloud based application with a micro service style architecture that it is deployed on. This strategy ensures the coverage of the missing areas like the implementation of a plan and Knowledge base component that can make use of other data otherwise unavail-

able or unused by the automation system and hence try and make better decisions in the automation of the application being monitored.

1.2 PROJECT DETAILS

As seen in the paper [2], a CBA Lifecycle is proposed, which can be viewed as a set of MAPE-K loops [7] shifting between the defined architectural models (alpha-topologies). These shifts are caused by controllers that provide coordination across the different stages of the lifecycle and in order to validate this, an IDE in the manner discussed by the MODAClouds approach [4] is required therefore for field study validation of the lifecycle through collaboration with the industry.

The system I have developed is meant to realise the for the above mentioned proposal through the implementation of the back end functionalities by the use of the aforementioned MAPE-K loops. MAPE-K stands for Monitor, Analyse, Plan and Execute by using Knowledge about the system's configuration and/ or including other information like the historical data. For the implementation of my system, I develop each of these as separate components, which interact with oneanother in order to complete this loop. This system's loop is run on top of a cloud container application and uses the available APIs from the container provider to monitor and record statistics of each of the containers in the particular micro service. These statistics are the start point at which the cloud application can be monitored and then optimised by switching between the different topologies provided by the owner of the application (System user). A switch occurs if the application topology doesn't meet the service level objectives specified by the owner of the application and it is either under using or over using the resources provided to it hence costing the application owner either financially or in terms of loss of users due to poor response times.

1.3 DOCUMENT ANALYSIS.

In the chapters that follow; Chapter 2 starts off with a literature review, which covers some of the fundamental concepts that make up the project hence providing some insight into Autonomic Computing. I then go ahead and provide the details of the system including the full system's requirements and additionally looking at the requirements of the individual components of the system in detail.

In Chapter 3, I provide the important design documentation of the system. I start by presenting some of the design patterns used for the system to achieve communication between the various components and other useful aspects that help the system achieve its complete

functionality. Additionally, the full system's architecture is also provided hence concluding the chapter.

Chapter 4, provides the some of the other details of the system by looking at the Technologies that help the system be able to accomplish the different tasks involved in the MAPE-K process and additionally provides the details of each of the MAPE-K components and finally how they interact with each by providing the full system's Architecture I then close off the chapter by describing some of the important algorithms used in the project.

Chapter 5 presents the Evaluation of the system this will introduce the applications that were chosen for testing my system, and then go further to include the different battery of tests run in order to confirm the functionality of the components of the system. The results of these tests will be presented and then data presented about the complete system's functionality.

Finally, in Chapter 6, I present the final review of the complete system by providing details of the various functionalities that I was able to implement and additionally, each of the unimplemented functionalities. This chapter is then closed off with a proposal of some of the future work that can be done in terms of both the extension of this system and in the field of Autonomous Application Topology Redistribution.

2 BACKGROUND AND RELATED WORK

In this chapter, a review of the literature relevant to the development process of this application and other relevant terms to help with further understanding of this project are to be presented. Further more, we take a look at the MODA Clouds project, which is another could application automation system, which uses the MAPE strategy and a few other systems developed that implement and test some of the individual components developed in this system.

2.1 BACKGROUND

This section covers some background knowledge into the work done in the research areas related to this work, including terms constantly used throughout this report, which help to drive this project.

2.1.1 *Topologies*

The concept an application topology, which based on the article The optimal distribution of applications in the cloud [3] in summary is a labelled graph with a set of nodes, edges, labels and, source and target functions. It introduces and explains in full detail the concept an application's topology. A Topology can be referred to as a μ-Topology, which is split into α-Topologies, and γ-Topologies, concepts which are also explained in the article mentioned above. The article also introduces the concept of viable topologies. The focus of this section will be on making clear what the α-Topologies and the Viable topologies are as these are the concepts most relevant to the system's development. Therefore, these are discussed further below including the relation to the project.

2.1.1.1 *α-Topologies*

As can be seen from an example in the article [3], a type graph for a viable application topology can be referred to as a μ-Topology and therefore, the α-Topology is the application specific subgraph of the μ-Topology, which refers to the general application architectural setup. The relation and importance of the above to the target functionality of the system being developed, which is to automatically switch between the available viable topologies provided by the user, is that it provided knowledge on what the system should be able to expect as input in terms of the topologies that will be used to determine the

best option for the optimisation of the application the system is run on.

2.1.1.2 *Viable Topologies*

Knowing about the α-Topologies, it becomes clear what the term viable topologies refers to in the context of this project, are the different suitable topology options that will be available to the automation system being developed in order for it to be able to select the best topology option, which follows the set Service Level agreements and therefore is the best option for the particular application usage scenario. The concept of viable topologies is important for the development process of this automation project because it is a requirement for the system that the application it will be run on should have a number of viable topologies defined by the system architect, which will be the topologies the system switches between to optimise the resource usage of the application.

2.1.2 *Auto-scaling (Autonomous Computing)*

Different researchers have delved into a number of projects using different strategies to try and solve a variety of problems in the field of automation. These different projects range from: *The monitoring of different metrics of the system*, For example whether to monitor the lower level metrics like the memory, cpu usage or even the network statistics or the higher level metrics like the rate of response to requests of the system being optimised. *To the type of analysis strategy used to determine whether to scale up or down* for example the use of a predictive methods (see [8]), reactive or rule based approaches (see [1]) or hybrid methods using both reactive and predictive approaches. These different projects and automation strategies are discussed in a survey [10], which helped provide an insight into the different options available to help with the completion of the different components of the system. The strategies used in the project will be looked into in a later chapter.

2.1.3 *Control Loops (MAPE-K)*

Throughout the various papers discussed in the survey [10] in the previous chapter, it is noticeable that the MAPE control loop strategy is a commonly used automation strategy, which involves the use four main procedures that make up this strategy, which are Monitor, Analyse, Plan and Execute. The MAPE automation strategy [7] used for this project is complimented with a Knowledge base, which all the MAPE components interact with in order for the system to make an informed optimisation decision. These MAPE control loops including the knowledge base are the core driver of the system being developed

as they are what makes up the complete functionality of the system, which are further looked at in a following chapter including the implementation process of the autonomous system.

2.1.4 *Microservices Architecture*

Microservices as discussed by Martin Fowler [9] describes an architecture style of building systems into a suite of smaller services each running on their own. These may be written in different programming languages and use different data storages. The microservice architecture is the architecture style focus for this project in terms of applications that the developed autonomous system will be able to perform its optimisation services on. The isolated services in this architecture style make it possible for the developed system to be able to individually run its loops on each service and therefore in the end perform the full assessment of the whole system, therefore performing its optimisation more efficiently and it's because of this characteristic that the Microservice style architecture was selected for this project. Additionally, the microservice style architecture is also well supported by most of the containerisation engines. This is an advantage in the case of this project since the popularity of containers among cloud developers recently has increased and therefore I was able to easily find a number of test applications that have the micro service architecture style and were deployed in containers especially the one selected for this project (Docker), which is introduced and shown later in a following chapter.

2.1.5 *Fuzzy logic*

2.2 RELATED WORK

- cloud specific automation

2.2.1 *MODA Clouds*

Other work that has been done similar to this and a more specified look into the MODA Clouds project.

2.2.2 *Other work*

[10]

3 SPECIFICATION AND DESIGN

In this chapter, I present the specification and design of the project in two sections, which include the requirements specification of the system and its components, and finally a use case diagram including some of the use cases.

3.1 REQUIREMENTS SPECIFICATION

This section presents some of the most important functional requirements in the project. I start off by presenting some of the complete system's requirements and then go ahead to present some of the requirements for the particular components of the system.

3.1.1 *System requirements*

3.1.2 *Component Requirements*

3.2 DESIGN DOCUMENTATION

3.2.1 *Use Cases*

3.2.2 *Use Case Diagram*

3.3 COMPLETE SYSTEM ARCHITECTURE?

4 IMPLEMENTATION

For this chapter, the details of the implementation of the system are presented including a Component Diagram showing the particular components of the system and how they interact and share data, some of the design patterns used, some of the technologies that were helpful and necessary to the implementation of some of the system's components including their use to the system and finally, a deeper look into the particular components that make up the system and their functionalities.

4.1 COMPONENT DIAGRAM AND PATTERNS

This section presents the component diagram, which is composed of each of the system's components and their interaction with each other in order to complete the system's main functionality.

4.1.1 *Component Diagram*

4.1.2 *Design Patterns*

4.2 TECHNOLOGIES AND THEIR UTILISATION IN THE SYSTEM

4.2.1 *Java*

4.2.2 *Docker*

NB: Show ease of use / microservice compatibility

4.2.2.1 *Docker API*

4.2.2.2 *Docker Compose*

4.2.3 *MySQL*

4.2.4 *Fuzzy Logic*

4.3 APPLICATION COMPONENTS

[6]

5 SYSTEM TESTING AND EVALUATION

During the development of the system, there were a battery of tests run on each of the individual components to ensure their functionality and furthermore their interaction with each other when integrated in order to ensure that they work together. However, this chapter presents the more important tests run on the more complete versions of the system. After completing most of the implementation of the system's components, that is until the level of the Analysis of the data that is collected by the system, the more important phase of testing begun and in the following sections, I present the results of most of these tests and finally an evaluation of the results of these tests.

5.1 MONITOR AND ANALYSIS COMPONENT TESTING.

5.2 PLAN AND EXECUTION TESTING.

5.3 FULL SYSTEM TESTING AND EVALUATION.

6 CONCLUSIONS

Part I

APPENDIX

A

APPENDIX

BIBLIOGRAPHY

[1] *AWS Auto Scaling*. https://aws.amazon.com/autoscaling/. Accessed: 2018-08-15.

[2] Vasilios Andrikopoulos. "Engineering Cloud-based Applications: Towards an Application Lifecycle." In: *European Conference on Service-Oriented and Cloud Computing*. Springer. 2017, pp. 57–72.

[3] Vasilios Andrikopoulos, Santiago Gómez Sáez, Frank Leymann, and Johannes Wettinger. "Optimal distribution of applications in the cloud." In: *International Conference on Advanced Information Systems Engineering*. Springer. 2014, pp. 75–90.

[4] Danilo Ardagna, Elisabetta Di Nitto, Giuliano Casale, Dana Petcu, Parastoo Mohagheghi, Sébastien Mosser, Peter Matthews, Anke Gericke, Cyril Ballagny, Francesco D'Andria, et al. "Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds." In: *Proceedings of the 4th International Workshop on Modeling in Software Engineering*. IEEE Press. 2012, pp. 50–56.

[5] Maricela-Georgiana Avram. "Advantages and challenges of adopting cloud computing from an enterprise perspective." In: *Procedia Technology* 12 (2014), pp. 529–534.

[6] Didac Gil De La Iglesia and Danny Weyns. "MAPE-K formal templates to rigorously design behaviors for self-adaptive systems." In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 10.3 (2015), p. 15.

[7] Jeffrey O Kephart and David M Chess. "The vision of autonomic computing." In: *Computer* 36.1 (2003), pp. 41–50.

[8] Joao Loff and Joao Garcia. "Vadara: Predictive elasticity for cloud applications." In: *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*. IEEE. 2014, pp. 541–546.

[9] *Microservices*. https://martinfowler.com/articles/microservices.html. Accessed: 2018-08-04.

[10] Chenhao Qu, Rodrigo N Calheiros, and Rajkumar Buyya. "Auto-scaling web applications in clouds: A taxonomy and survey." In: *ACM Computing Surveys (CSUR)* 51.4 (2018), p. 73.

[11] Alfonso Quarati, Daniele D'Agostino, Antonella Galizia, Matteo Mangini, and Andrea Clematis. "Delivering cloud services with QoS requirements: an opportunity for ICT SMEs." In: *International Conference on Grid Economics and Business Models*. Springer. 2012, pp. 197–211.

DECLARATION

Put your declaration here.

Netherlands, May 2018

Eric Rwemigabo