

**A CONTAINER-BASED PLATFORM FOR MULTI-  
CLOUD APPLICATION ORCHESTRATION**

A.M.A.S. Adikari

(179301K)

Degree of MSc in Computer Science specialising in Cloud Computing

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2020

# **A CONTAINER-BASED PLATFORM FOR MULTI- CLOUD APPLICATION ORCHESTRATION**

Adikari Mudiyanseelage Akila Srinath Adikari

(179301K)

Thesis submitted in partial fulfillment of the requirements for the  
degree Master of Science in Computer Science specialising in Cloud Computing

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka


May 2020

## DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgment any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature:

  
\_\_\_\_\_  
(A. M. A. S. Adikari)

Date: 21/04/2020

The above candidate has carried out research for the Master's dissertation under my supervision.

Name of the supervisor: Dr. H. M. N. Dilum Bandara

Signature of the Supervisor:

  
\_\_\_\_\_

Date: 17/03/2020

## ABSTRACT

Multi-cloud applications are becoming popular, as they can run across multiple public and private cloud platforms while overcoming vendor lock-in, reducing cost, and enhancing flexibility and reliability. Applications hosted on multiple cloud platforms use either libraries or service-based abstraction layers. Application orchestration platforms further simplify the deployment and management of multi-cloud applications by providing auto-scaling, service metering, health monitoring, and a rich set of operational tools. Containerization is particularly useful in multi-cloud applications, as it provides a consistent environment for an application regardless of where it is deployed. However, container orchestration platforms such as Docker Swarm lack support and operational tools to enable seamless application orchestration across multi-cloud resources.

In this research, we developed a container-based platform for application orchestration in a multi-cloud setup as a set of microservices and required operational tools addressing the above limitations. Docker was chosen to demonstrate the proof of concept solution, as it already provides features to orchestrate microservices. Containerized multi-cloud applications can use the proposed application orchestration platform to achieve resource elasticity across multiple cloud platforms. To trigger scale in and out decisions, we used a rule-based approach where we compared the container runtime metrics provided by Docker with preconfigured threshold values. We evaluated the utility of the proposed platform using three web applications that were compute-intensive, memory-intensive, and utilized a RESTful application programming interface integrated with an external cloud service. The proposed container-based application orchestration platform improved the throughput of the three web applications by 180%, 73%, and 46%, respectively, compared to the same web applications deployed in a private cloud. Whereas the response time was reduced by 36%, -232%, and 7%, respectively. Even for cases where latency is increased error rate was reduced.

## **ACKNOWLEDGMENTS**

I would like to express my sincere gratitude to my research supervisor Dr. Dilum Bandara for providing supervision and resources to enhance my research idea. His expertise in the related field was an immense support for me to initiate this research work in this passion and identify possible technologies to complete this research.

I would also like to thank Dr. Malaka Walpola, Dr. Indika Perera and Dr. Charith Chitraranjan for helping and encouraging us to initiate and continuing this research until the end. Further, I would like to thank all my colleagues for joining together to share knowledge, research material, technology guidance, and experience to make this research success. Specially appreciate their encouragement till the end.

This would not have been a reality without the support and love from my parents throughout my life. I am deeply grateful for them for being ever strong support in every step of my life and heartfelt of blessings. Special thank goes to my beloved wife, including my family for being with me and sharing time to encourage to complete this research.

I would like to greatly thank Dr. Sankalpa Gamwarige, General Manager, Zone24x7 supporting me by providing necessary guidance to select this degree program and all my colleagues at Zone24x7 for their continuous support to manage my work and MSc research work.

# TABLE OF CONTENTS

DECLARATION .....	i
ABSTRACT .....	ii
ACKNOWLEDGMENTS .....	iii
TABLE OF CONTENTS .....	iv
LIST OF FIGURES .....	vi
LIST OF TABLES .....	viii
LIST OF ABBREVIATIONS .....	ix
1 INTRODUCTION .....	1
1.1 Background .....	1
1.2 Motivation .....	2
1.3 Problem Statement .....	3
1.4 Objectives .....	4
1.5 Outline .....	4
2 LITERATURE REVIEW .....	6
2.1 Cloud Computing .....	6
2.1.1 Cloud Deployment Models .....	6
2.1.2 Virtualization Based on VMs .....	7
2.1.3 Container-based Virtualization .....	9
2.2 Multi-Cloud .....	13
2.2.1 Multi-Cloud Software Solutions .....	14
2.2.2 Multi-Cloud Microservice Architecture .....	15
2.2.3 PaaS Solutions for Private Clouds .....	15
3 METHODOLOGY .....	19
3.1 Solution Approach .....	19
3.2 High-Level Architecture .....	21
3.3 Detailed Design .....	26
3.3.1 AppDock Cluster Admin .....	27
3.3.2 Command Line Interface .....	31
3.3.3 AppDock Scaling Service .....	33
3.3.4 AppDock Monitoring Agent .....	36

3.3.5	AppDock LogDB .....	38
3.3.6	AppDock HTTP Proxy Interface .....	41
3.3.7	Docker HTTP Proxy Interface .....	41
3.4	Cluster Deployment .....	41
3.5	Summary .....	42
4	PERFORMANCE EVALUATION .....	44
4.1	Workload.....	44
4.2	Experimental Setup .....	47
4.3	Performance Evaluation of CPU Intensive Workload .....	49
4.3.1	Throughput Analysis .....	49
4.3.2	Response Time Analysis .....	50
4.3.3	Resource Utilization Analysis.....	51
4.4	Performance Evaluation of Memory Intensive Workload .....	52
4.4.1	Throughput Analysis.....	53
4.4.2	Response Time Analysis .....	54
4.4.3	Resource Utilization Analysis.....	54
4.5	Performance Evaluation of RESTful API Workload.....	56
4.5.1	Throughput Analysis.....	57
4.5.2	Response Time Analysis .....	58
4.5.3	Resource Utilization Comparison .....	58
4.6	Summary .....	60
5	CONCLUSIONS.....	61
5.1	Summary .....	61
5.2	Research Limitations.....	63
5.3	Future Work .....	65
	References .....	67
	APPENDIX A – Available Methods in Proxy Interfaces .....	71
	APPENDIX B – Commands in AppDock CLI.....	73

## LIST OF FIGURES

Figure 2-1	Virtualization via containers and VMs.....	9
Figure 2-2	Docker architecture. Source:.....	10
Figure 3-1	Conceptual view of a service that integrates containers across multiple cloud providers.....	20
Figure 3-2	High-level deployment diagram of the platform deployed in private-public IaaS infrastructure. ....	22
Figure 3-3	Detailed view of the orchestration layer.....	23
Figure 3-4	Service view of the AppDock platform.....	25
Figure 3-5	Component view of the application.....	27
Figure 3-6	Configurations required by cloud providers.....	29
Figure 3-7	File storage built using Docker volumes and NFS.....	30
Figure 3-8	Properties maintained by local storage.....	33
Figure 3-9	Properties of INodeStatAnalysisStatus object.....	34
Figure 3-10	Properties of IScalingServiceConfig object .....	34
Figure 3-11	Algorithm for adding and removing nodes. ....	35
Figure 3-12	Properties of IRuntimeStat object .....	37
Figure 3-13	Calculation for CPU utilization.....	38
Figure 3-14	Calculation for memory utilization. ....	38
Figure 3-15	Class diagram for the repository. ....	39
Figure 3-16	Configurations required when deploying an AppDock cluster .....	42
Figure 4-1	Deployment diagram for the experimental setup. ....	48
Figure 4-2	Throughput comparison – CPU intensive application. ....	50
Figure 4-3	Response time comparison - CPU intensive application.....	50
Figure 4-4	CPU utilization under CPU intensive workload – private cloud mode..	51
Figure 4-5	CPU utilization under CPU intensive workload – multi-cloud mode....	51
Figure 4-6	Memory utilization under CPU intensive workload – private cloud mode.....	52
Figure 4-7	Memory utilization under CPU intensive workload – multi-cloud mode. ....	52
Figure 4-8	Throughput comparison – memory-intensive workload. ....	53
Figure 4-9	Response time comparison - memory intensive workload.....	54
Figure 4-10	Memory utilization under memory-intensive workload – private cloud mode.....	55
Figure 4-11	Memory utilization under memory-intensive workload – multi-cloud mode.....	55
Figure 4-12	CPU utilization under memory-intensive workload – private cloud mode.....	56
Figure 4-13	CPU utilization under memory-intensive workload – multi-cloud mode. ....	56
Figure 4-14	Overall throughput comparison – REST API workload. ....	57



Figure 4-15 Overall response time comparison - REST API workload.....	58
Figure 4-16 CPU utilization under REST API workload - private cloud mode .....	58
Figure 4-17 CPU utilization under REST API workload- multi-cloud mode.....	59
Figure 4-18 Memory utilization under REST API workload – private cloud mode. .....	59
Figure 4-19 Memory utilization under REST API workload – multi-cloud mode..	60

## LIST OF TABLES

Table 2-1	Container technologies used by different PaaS vendors.....	13
Table 4-1	Testing parameters for the CPU-intensive workload.....	45
Table 4-2	Testing parameters for the memory-intensive workload.....	45
Table 4-3	Testing parameters for the RESTful API application.....	46
Table A-1	Available methods in the AppDock HTTP proxy interface .....	71
Table A-2	Available methods in the Docker HTTP proxy interface .....	72

## **LIST OF ABBREVIATIONS**

API	Application Programming Interface
ARaaS	Application Runtime as a Service
AWS	Amazon Web Services
CD	Continuous Deployment
CLI	Command Line Interface
CN	Container
CPU	Central Processing Unit
CRUD	Create, Read, Update and Delete
DB	Database
DoS	Denial of Service
EC2	Elastic Compute Cloud
HW	Hardware
IaaS	Infrastructure as a Service
IT	Information Technology
LXC	Linux Containers
NFS	Network File System
ODM	Object Document Mapper
OS	Operating System
PaaS	Platform as a Service
QoS	Quality of Service
REST	Representational State Transfer
SaaS	Software as a Service
SDK	Software Development Kit
SLA	Service Level Agreements
SOA	Service-Oriented Architecture
SW	Software
vCPU	Virtual Central Processing Unit
VM	Virtual Machine