

**A COST-EFFECTIVE AUTO-SCALING STRATEGY  
BETWEEN SERVER AND SERVERLESS  
ARCHITECTURES FOR CLOUD DEPLOYMENTS**

LDSB Weerasinghe

(189357U)

Degree of Master of Science in Computer Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

May 2020

**A COST-EFFECTIVE AUTO-SCALING STRATEGY  
BETWEEN SERVER AND SERVERLESS  
ARCHITECTURES FOR CLOUD DEPLOYMENTS**

LDSB Weerasinghe

(189357U)

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

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 acknowledgement 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: .....

Date:.....

Name: Sidath Weerasinghe

The supervisor should certify the dissertation with the following declaration.

The above candidate has carried out research for the Masters Dissertation under my supervision.

Signature of the supervisor: .....

Date: .....

Name: Dr. Indika Perera

## ABSTRACT

Nowadays, cloud computing is marking as an emerging technology in the universe. Most of people use cloud environments to deploy their systems. Many enterprise systems give their benchmarks according to the specific instances in the cloud. Cloud providers provide primary services as SaaS, PaaS, IaaS and FaaS, then Cloud consumers use those services as per their demands.

The systems which are developed on several years ago are still running upon on-premise environments. They are tightly coupled legacy systems. Hence people cannot adopt new technologies to those monolithic systems. Nevertheless, people who are in the software industry need to integrate new technologies into their systems to stay competitive in the IT industry. To get into the new technology that should be cost-effective, more reliable, and should need to cater the new and existing functionalities of the system. One of the problems is that migrating the whole system or partial system to the new technology is hazardous. Sometimes the system needs to handle substantial traffic loads, according to the business needs. In order to satisfy sudden traffic spikes, systems need to scale horizontally or vertically. If their application server is deployed upon the on-premise server, scaling is complicated, and if the application is deployed on the cloud, VMs scaling is possible, but it can be very costly.

In this research, the goal is to overcome these scaling problems. The research component presents the strategy to load balancing the traffic between the server and serverless function. The cost of the serverless function calculated only for the executes requests, so it is cheaper than running a new server. Therefore this hybrid strategy is a highly cost-effective way to scaling the servers and also proven that the proposed system is capable of load balancing the traffic according to the server loads with the analytical results. Other researches are proven that this research component is the right solution for ongoing industry problems.

**Keywords:** Cloud Computing, Serverless, Load Balancing

## **ACKNOWLEDGMENTS**

My sincere appreciation goes to my family for the continuous support and motivation given to manage my MSc research work. I also express my heartfelt gratitude to Dr. Indika Perera, my supervisor, for the supervision and advice given throughout to this research period. I also thank my batch mates, for their untiring help to find research materials. I am also thankful to WSO2 Telco (Pvt) Ltd, for providing support to complete this research.

# TABLE OF CONTENTS

DECLARATION .....	ii
ABSTRACT .....	iii
ACKNOWLEDGMENTS .....	iv
TABLE OF CONTENTS.....	v
LIST OF FIGURES .....	vii
LIST OF TABLES.....	ix
LIST OF ABBREVIATIONS.....	x
CHAPTER 1 INTRODUCTION.....	1
1.1 Preamble .....	1
1.2 Motivation.....	2
1.2.1 Infrastructure as a Service (IaaS).....	2
1.2.2 Platform as a Service (PaaS).....	2
1.2.3 Software as a Service (SaaS) .....	3
1.2.4 Cloud types .....	3
1.3 Monolithic Applications .....	5
1.4 Microservices.....	7
1.5 Serverless Computing.....	8
1.6 Problem Description .....	9
1.6.1 Goal.....	9
1.6.2 Objectives .....	10
CHAPTER 2 LITERATURE REVIEW .....	11
2.1 Introduction.....	11
2.2 Cloud Computing.....	11
2.2.1 Infrastructure as a service (IaaS).....	11
2.2.2 Platform as a service (PaaS) .....	13
2.2.3 Software as a service (SaaS).....	15
2.3 Serverless Computing and Microservices.....	15
2.4 Workload Load Balancing .....	21
2.4.1 Round Robin Method.....	21
2.4.2 Central Manager Load Balancing Algorithm.....	22

2.4.3 Self-Adaptive Load Balancing Algorithm .....	23
2.5 Workload Prediction and Resource Allocation.....	24
2.1.5 Auto Scaling .....	29
CHAPTER 3 METHODOLOGY .....	32
3.1 Introduction.....	32
3.2 Converting Monolithic application to Microservices .....	32
3.3 Converting Microservices into serverless .....	34
3.4 Serverless (AWS) .....	37
3.5 Server Status Monitoring – NRPE.....	40
3.6 Architecture Design .....	43
3.7 Implementation .....	44
3.8 Integration to Existing Systems .....	47
CHAPTER 4 RESULT AND EVALUATION.....	49
4.1 Introduction.....	49
4.2 Testing on System.....	49
4.3 Test Strategies and Procedure.....	50
4.4 Evaluation .....	52
4.4.1 Performance Evaluation.....	52
4.4.2 Cost Evaluation.....	61
CHAPTER 4 CONCLUSION .....	64
4.1 Introduction.....	64
4.2 Problems Faced and the Benefits of the Research Module.....	64
4.3 Revisited the Objectives .....	65
4.4 Limitations .....	67
4.5 Further works .....	67
4.6 Summary .....	67
REFERENCES .....	68

## LIST OF FIGURES

Figure 1 : Cloud computing stack.....	1
Figure 2: Monolithic application architecture.....	6
Figure 3: Microservices architecture.....	7
Figure 4: How developer control the serverless cloud.....	16
Figure 5: Architecture of serverless platform.....	17
Figure 6: IO overhead while concurrent execution.....	19
Figure 7: Serverless function throughput on concurrent invocations.....	19
Figure 8: CPU utilization during the concurrent execution.....	20
Figure 9: Price comparison.....	20
Figure 10: Round Robin method.....	21
Figure 11: Central manager algorithm.....	22
Figure 12: Work flow of self-adaptive load balancing algorithm.....	23
Figure 13: How work perdition happen.....	26
Figure 14: Components of the rule based resource allocation.....	27
Figure 15: Results graph after resource allocation.....	27
Figure 16: CPU load average's difference.....	28
Figure 17: Horizontal scaling.....	29
Figure 18: Vertical scaling.....	30
Figure 19: Hybrid approach.....	33
Figure 20: Microservices communication architecture.....	34
Figure 21: Java dependency for AWS serverless.....	35
Figure 22: Java wrapper for microservices.....	36
Figure 23: Creating Lambda function.....	37
Figure 24: Creating roles on Lambda.....	38
Figure 25: Uploading jars to Lambda.....	38
Figure 26: Adding API gateway.....	39
Figure 27: Integration on Lambda.....	39
Figure 28: NRPE architecture.....	40
Figure 29: NRPE configurations on security.....	41
Figure 30: NRPE implantation.....	41
Figure 31: NRPE plugins.....	42
Figure 32: Java dependency for NRPE.....	42
Figure 33: Creating JNRPE client.....	42
Figure 34: Results for JNRPE.....	43
Figure 35: High-level architecture.....	44
Figure 36: Configuration file.....	45
Figure 37: Allow all traffic.....	45
Figure 38: Logical flow diagram.....	46
Figure 39: Schedule tasks.....	46
Figure 40: Existing system architecture diagram.....	47
Figure 41: Architecture with new component.....	48



Figure 42: Load test with Jmeter .....	50
Figure 43: Testing with research component.....	51
Figure 44: Graph of statistics for test case 1.....	53
Figure 45: Graph of statistics for test case 1 with research component .....	54
Figure 46: Graph of statistics for test case 2.....	55
Figure 47: Graph of statistics for test case 2 with research component .....	56
Figure 48: Graph of statistics for test case 3.....	57
Figure 49: Graph of statistics for test case 3 with research component .....	58
Figure 50: Graph of statistics for test case 4.....	59
Figure 51: Graph of statistics for test case 4 with research component .....	60
Figure 52: Cost calculation for 20TPS .....	61
Figure 53: Cost calculation on 20TPS in serverless .....	62
Figure 54: Cost calculation on 100TPS in serverless.....	62
Figure 55: Graph on cost comparison .....	63

## LIST OF TABLES

Table 1: Comparison on Azure & Google.....	13
Table 2: Algorithm comparison .....	23
Table 3: Server specification .....	51
Table 4: Server specification (with research component).....	51
Table 5: Test case 1 details.....	52
Table 6: Test case 1 with research component details .....	53
Table 7: Request count on test case 1.....	54
Table 8: Test case 2 details.....	55
Table 9: Test case 2 with research component details .....	55
Table 10: Test case 2 request count .....	56
Table 11: Test case 3 details .....	57
Table 12: Test case 3 with research component details.....	57
Table 13: Test case 3 request count .....	58
Table 14: Test case 4 details .....	59
Table 15: Test case 4 with research component details.....	59
Table 16: Test case 4 request count .....	60
Table 17: AWS cloud cost estimation .....	61

## LIST OF ABBREVIATIONS

SaaS	Software as a Service
PaaS	Platform as a Service
FaaS	Function as a Service
IaaS	Infrastructure as a Service
NRPE	Nagios Remote Plugin Executor
AWS	Amazon Web Services
GCP	Google Cloud Platform
TPS	Transitions Per Second