

Container Scheduling Optimization in a Clustering Environment

H G P Madhushankha

(219360K)

Degree of Master of Science in Computer Science

Department of Computer Science

Faculty of Engineering

University of Moratuwa

Sri Lanka

June 2024

DECLARATION

I declare that this is my own work, and this thesis/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 the University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/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: 16/06/2024

The above candidate has carried out research for the Master's dissertation under my supervision. I confirm that the declaration made above by the student is true and correct.

Name of the Supervisor: Prof. Indika Perera

Signature of the Supervisor:

Date: 17/06/2024

ACKNOWLEDGEMENT

This research has been made with the intention of deriving a solution in the microservice world which is improving day by day with the efforts of different fields of expertise. For the enhancement of this research, there were several people who spent their time and helped with their capabilities. I would like to offer my heartfelt gratitude to all of them including my supervisor Prof. Indika Perera, Department of Computer Science & Engineering, University of Moratuwa. And special thanks to my parents and my colleagues who encouraged me all the time.

ABSTRACT

The progress in container orchestration technology has led to a saturation level of microservices related solutions in recent years. Web applications typically exhibit a fluctuating workload that is challenging to estimate. Based on system parameters, the microservice design has the capability to dynamically alter the number of containers, either manually or automatically. There is a persistent issue regarding the adjustment of the number of containers in real time to match the current workload of the microservice. Several strategies have been utilized to tackle this difficulty effectively and swiftly, particularly in situations where there is a sudden and significant increase or decrease in the demand for services.

Scheduling the container resources according to the demand is one of the most focused requirement of the industry. Service providers and researchers are attempting to decrease the expenses while upholding the Quality of Service. Among the autoscaling approaches rule-based approach, machine learning models, Fuzzy rules, control theory and hybrid approaches have been discussed in previous studies. To deal with the above research problem, a container scheduling strategy as a hybrid approach, based on machine learning techniques and rule based techniques will be discussed in this article.

There is still potential for improvement in optimizing container scheduling based on fluctuating workloads while utilizing cluster resources. Based on this study and analysis the article has suggested a solution for Container Scheduling Optimization in Clustering Environment using a machine learning model with a rule-based approach as a hybrid approach to determine the scaling factor for the demand of the microservices. The suggested hybrid technique offers greater granularity in manipulating the scaling decisions. This is because the system predicts low-level parameters (demanding CPU Units) based on high-level data, rather than directly influencing the scaling factor.

This research aims to assess the reliability of the proposed autoscaling strategy by analyzing the results obtained. This offers an implementation of autoscaling using microservices and a reference design for domains that are shared by the community.

Keywords: microservices, container, machine learning, Fuzzy rules, control theory

TABLE OF CONTENT

1	Introduction.....	1
1.1	Background of Project.....	1
1.2	Motivation	1
1.3	Research Problem.....	2
1.4	Aims	2
1.5	Objectives	3
1.6	The Methodology Chosen	3
1.7	Summary of Chapters	3
2	Literature Review	4
2.1	Introduction	4
2.2	What is Autoscaling	4
2.3	What is the Purpose of Autoscaling?.....	5
2.4	Autoscaling Methods.....	5
2.5	Resource Estimation.....	6
2.6	Taxonomy of Autoscaling	6
2.7	Scaling Indicators	8
2.7.1	Low-Level Metrics.....	9
2.7.2	High-Level Metrics	9
2.7.3	Hybrid Metrics	9
2.8	Autoscaling Approaches.....	10
2.8.1	Rule-Based Approaches	10
2.8.2	Machine Learning	11
2.8.2.1	Reinforcement Learning	11
2.8.2.2	Support Vector Machine (SVM).....	13
2.8.2.3	Q-learning	13
2.8.2.4	Regression.....	13
2.8.2.5	Time Series Analysis	14
2.8.3	Hybrid Approaches	14
2.8.4	Analytical Modeling.....	14
2.8.5	Fuzzy Rules	15
2.8.6	Control Theory	16
2.9	Oscillation Mitigation.....	16
2.9.1	Cooling Time	17

2.9.2	Dynamic Parameters	17
2.10	ML Algorithm Related Works	17
2.11	Conclusion	22
3	Methodology	23
3.1	Introduction	23
3.2	Requirement Specification	24
3.2.1	Finding from Requirement elicitation	25
3.3	Use Case Diagram	26
3.4	Functional requirements	27
3.5	Non -Functional Requirements	27
3.6	System Design	28
3.6.1	System Architecture	28
3.6.2	Component Diagram of the Proposed Solution	30
3.6.3	Sequence Diagrams	31
3.6.3.1	Initial Training Phase	31
3.6.3.2	Execution of the Scaling Decision	31
3.6.3.3	Continuous Model Training	32
3.6.4	Process Flow Chart	33
3.6.4.1	Model Training Flow Chart	33
3.6.4.2	Autoscaling Decision Execution Flow Chart	34
4	Development And Implementation	35
4.1	Introduction	35
4.1.1	Tiers of the system representation	35
4.2	System implementation	36
4.2.1	Development Environment	36
4.2.2	Development Tools and Frameworks	37
4.2.3	Sample Web Application (Kasper)	37
4.2.3.1	Deployment Flow of the Sample Web Application	38
4.2.4	Controller Application	40
4.2.4.1	Metrics Reader Service in Controller Application	41
4.2.4.2	Autoscaling Service in the Controller Application	42
4.2.5	CPU Prediction Model	43
4.2.6	Final ML Model Selection	45
4.3	System Implementation Challenges	45
4.3.1	Main challenges faced	45

5	Evaluation	45
5.1	System testing.....	46
5.1.1	Unit Testing.....	46
5.1.2	Integration Testing	46
5.1.3	Functional Testing.....	46
5.1.4	Non-Functional Testing	50
5.1.4.1	Performance Measures.....	50
5.1.4.2	Scalability Measures	50
5.1.4.3	Security Measures.....	50
5.1.4.4	Maintainability.....	50
5.1.5	User Experience	51
5.1.6	Limitations in Testing	51
5.2	System Design and Architecture of the Solution	52
5.3	Metric Statistics Fetching and Load Pressure Analysis	52
5.3.1	Linear Shape Cluster Metrics Distribution	54
5.3.2	Zig-Zag Shape Cluster Metrics Distribution.....	56
5.3.3	Ladder Shape Cluster Metrics Distribution.....	58
5.3.4	Training Data Set	60
5.4	Machine Learning Prediction Model for Load Pressure Evaluation	61
5.4.1	Standard evaluations for the ML algorithms.....	63
5.5	Scaling Decision Mechanism	64
5.6	Autoscaling Operation.....	64
6	Conclusion	72
6.1	Summary	72
6.2	Research Limitations	73
6.3	Future Work	73

LIST OF FIGURES

Figure	Description	Page
Figure 1	- How Autoscaling works	4
Figure 2	- The taxonomy of web application autoscaling.....	7
Figure 3	- Rule base autoscaling algorithm.....	10
Figure 4	- Experimental Setup	20
Figure 5	- Use case of the proposed system.....	26
Figure 6	- High Level System Architecture	29
Figure 7	- Component diagram of the proposed solution	30
Figure 8	- Sequential diagram for the initial model training.....	31
Figure 9	- Sequence diagram for execution of the autoscaling.....	32
Figure 10	- Sequence diagram for continuous model training.....	32
Figure 11	-ML Model training flow chart	33
Figure 12	- Autoscaling decision execution flow chart	34
Figure 13	- Tiers of the system	35
Figure 14	- System Implementation.....	36
Figure 15	- High level view of the sample web application	37
Figure 16	- Deployment flow of the web application	38
Figure 17	- Dockerfile.....	39
Figure 18	- docker build step	39
Figure 19	- docker tag step.....	39
Figure 20	- docker push step	40
Figure 21	- Metrics reader service	41
Figure 22	- Autoscaling service	42
Figure 23	- CPU Prediction model flow chart	43
Figure 24	- Linear shape load pattern	52
Figure 25	- Zig-Zag shape load pattern.....	53
Figure 26	- Ladder shape load pattern.....	53
Figure 27	- Test results for linear shape load.....	54
Figure 28	- Test results for zig-zag shape load	54
Figure 29	- Test results for ladder shape load	54
Figure 30	- Requests count.....	54
Figure 31	- Response Time	55
Figure 32	- Active connection count.....	55
Figure 33	- New connection count.....	55

Figure 34 - CPU utilization	56
Figure 35 - Request Count	56
Figure 36 - Response Time	57
Figure 37 - Active connection count	57
Figure 38 - New connection count	57
Figure 39 - CPU Utilization in the cluster	58
Figure 40 - Request Count	58
Figure 41 - Response Time	59
Figure 42 - Active connection count	59
Figure 43 - New connection count	59
Figure 44 - CPU Utilization in the cluster	60
Figure 45 – Training data-set.csv	60
Figure 46 – Support Vector Regression results	61
Figure 47 - Random Forest Regression results	62
Figure 48 - Decision Tree Regression results	62
Figure 49 - Polynomial Regression results	62
Figure 50 - Multiple Linear Regression results.....	63
Figure 51 - Random input load	65
Figure 52 - Request Count	65
Figure 53 - Response Time distribution.....	66
Figure 54 - Active Connection Count	67
Figure 55 - New connection count	67
Figure 56 - Container Scheduling	68
Figure 57 - Cluster CPU Utilization	69
Figure 58 - Cluster CPU utilization deviation against optimal CPU in step scaling .	69
Figure 59 - Cluster CPU utilization deviation against optimal CPU in predictive scaling	70
Figure 60 - CPU Units Utilization	71

LIST OF TABLES

Table	Description	Page
Table 1	- Feature vector and Regression target.....	21
Table 2	- Accuracy comparison	21
Table 3	- Selected development strategies	23
Table 4	- Regression results	63

LIST OF ABBREVIATIONS

Abbreviation	Description
ML	Machine Learning
SLA	Service Level Agreement
SLO	Service Level Objectives
VM	Virtual Machine
RAM	Random Access Memory
CPU	Central Processing Unit
QoS	Quality of Service
RIL	Reinforcement Learning
SVM	Support Vector Machine
SVR	Support Vector Regression
MAPE	Monitor Analyze Plan Execute
IDE	integrated development environment
SOA	Service Oriented Architecture
IoT	Internet of Things
PID	Proportional Integral Derivative
PI	Proportional Integral
I	Integral
ANN	Artificial Neural Network
RT	Regression Tree
RF	Random Forest
AWS	Amazon Web Services

LIST OF APPENDICES

Appendix	Description	Page
Appendix A	Sample Web Application docker images in AWS ECR	79
Appendix B	AWS Task Definition for Sample Web Application deployment in AWS ECS Cluster	80
Appendix C	AWS Cost consumption in last few Months	80