

LB/TH/43/2025
TH6015

**A Comparative Analysis of OpenStack Autoscaling
Engines: Evaluating Performance and Scalability, and
Usability of Heat and Senlin under Real-World Workload
Patterns**

Liyanage Dinith Ishanka Appuhami

239305R

Master of Science in Computer Science

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

January 2025

**A Comparative Analysis of OpenStack Autoscaling
Engines: Evaluating Performance, Scalability, and Usability
of Heat and Senlin under Real-World Workload Patterns**

Liyanage Dinith Ishanka Appuhami

239305R

Thesis submitted in partial fulfillment of the requirements for the degree.

Master of Science in Computer Science

Department of Computer Science and Engineering

Faculty of Engineering

University of Moratuwa

Sri Lanka

January 2025

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. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature:

Date: 21.04.2025

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

Name of Supervisor: Dr. M.P.A.P. Wijayasiri

Signature of the Supervisor:

Date: 21.04.2025

DEDICATION

I dedicate this thesis to the individuals who have helped me along the way in my academic career. I appreciate seeing this journey through to the very finish.

ACKNOWLEDGEMENT

First, thanks go to my supervisor, Dr. Adeesha Wijayasiri, for his guidance during this thesis project. He always provided thorough feedback when I needed it. I consider myself very lucky to have had the chance to work under his guidance.

Also, I'm grateful to all the MSc in CS academic staff who taught me different subjects in computer science.

Furthermore, I would like to thank the network and data center teams of Srilankan Airlines IT for allowing me to use their infrastructure for practicing and testing my work.

Finally, I would like to thank my family for their encouragement and understanding throughout this work.

ABSTRACT

Cloud computing is a demanded field that opens the door for various possibilities due to its characteristics such as flexibility and on-demand availability. Basically, it presents an infrastructure for different kinds of services and tools via the public internet [1]. Auto-scaling, that is allocating and deleting resources without the involvement of the user, is one of the principal features in the domain of cloud computing. [2]. In domain of autoscaling on cloud systems, the demand for autoscaling systems, applications and services continues to surge, there is an increasing need for flexible and cost-efficient solutions to handle dynamic traffic patterns. Most of the time private cloud based autoscaling solutions are not used, because of the absence of proper information of the private cloud based autoscaling solutions and their performance. Specially in OpenStack who is the leading private cloud provider in the domain of cloud computing. As a result of this lack of knowledge and performance information of the private cloud based autoscaling solutions, the community are still using manual scaling methods when necessary or they are using public cloud based autoscaling solutions. However, there are several identified issues such as inefficiency and management challenges in manual scaling methods. When it comes to the public cloud based autoscaling solutions, there are set of challenges and cons for example, lack of cost-effectiveness, lack of transparency and administrative control, and vendor lock-in issues are few of them. As a solution for the above-mentioned issues, in this project we are giving the community an opportunity to get an idea about the OpenStack based autoscaling solutions (namely Heat and Senlin) and their behavior and performance against various kinds of practical workloads. As per our knowledge, there is not any previous literature that compares the different autoscaling solutions in OpenStack cloud. This work involves doing comprehensive research on the existing cloud-based auto-scaling solutions and the Zed version of fully functional OpenStack environment is implemented including the Heat and the Senlin projects. These autoscaling engines are tested with 10 different kinds of practical workloads which are generated using Apache JMeter and their performance metrics are recorded. These metrics are analyzed using MCDA method and the optimum autoscaling solution for each workload pattern is determined. We hope this work will contribute to the community who are interested in OpenStack based autoscaling to decide the proper autoscaling solution based on the nature of their workloads.

Keywords: Cloud computing, Auto-scaling, OpenStack, Heat, Senlin, MCDA

TABLE OF CONTENTS

Declaration	i
Dedication	ii
Acknowledgement.....	iii
Abstract	iv
Table of Contents	v
List of Figures	ix
List of Tables.....	xi
List of Abbreviations.....	xii
List of Appendices	xiii
Chapter 1	1
Introduction	1
1.1 What is Cloud Computing?.....	1
1.2 Cloud computing service models	1
1.2.1 Infrastructure as a Service (IaaS)	1
1.2.2 Platform as a Service (PaaS)	2
1.2.3 Software as a Service (SaaS).....	2
1.3 Cloud computing deployment models	3
1.3.1 Public Cloud.....	3
1.3.2 Private Cloud.....	3
1.3.3 Hybrid Cloud.....	4
1.3.4 Community Cloud.....	4
1.3.5 Multi-Cloud.....	4
1.4 Virtualization in Cloud Computing.....	4
1.5 Auto-scaling in cloud computing.....	5
1.5.1 Classification of auto-scaling techniques.....	5
1.5.1.1 Threshold-based Rules	6
1.5.1.2 Reinforcement learning.....	6
1.5.1.3 Fuzzy learning.....	6
1.5.1.4 Queuing theory.....	7

1.5.1.5 Time-series analysis	7
1.5.1.6 Machine learning techniques	7
1.5.2 Available cloud-based autoscaling in the industry	7
1.5.2.1 Rule-based autoscaling.....	8
1.5.2.2 Step scaling	8
1.5.2.3 Target tracking scaling.....	9
1.5.2.4 Scheduled scaling.....	9
1.6 Cloud workloads and workload patterns.....	9
1.6.1 Cloud workloads	9
1.6.2 Cloud workload patterns	10
1.6.2.1 Static workload.....	11
1.6.2.2 High growth workload	11
1.6.2.3 On-Off workload pattern.....	12
1.6.2.4 Aperiodic bursting workload pattern	12
1.6.2.5 Periodic bursting workload pattern	13
1.6.2.6 High Growth - Static - High Shrink workload pattern.....	13
1.6.2.7 High Growth - Static - High Shrink On-Off with No Interval Workload Pattern	14
1.6.2.8 High Growth - Static - High Shrink On-Off with Interval Workload Pattern	14
1.6.2.9 High Growth On-Off Workload Pattern	15
1.6.2.10 High Growth High Shrink Workload Pattern	15
1.7 OpenStack Services.....	16
1.7.1 Dashboard (Horizon).....	16
1.7.2 Nova (Compute).....	17
1.7.3 Neutron (Network)	17
1.7.4 Swift (Object Storage)	17
1.7.5 Cinder (Block Storage)	17
1.7.6 Keystone (Identity).....	17
1.7.7 Glance (Image service)	17
1.7.8 Ceilometer (Telemetry).....	18
1.7.8.1 Key Features of Ceilometer:	18

1.7.8.2 Use Cases of Ceilometer:	20
1.7.9 Aodh (Alarming)	21
1.7.10 Gnocchi (Time series database)	21
1.7.11 Octavia (LBaaS)	22
1.7.12 Heat (Orchestration)	24
1.7.12.1 How it works:	24
1.7.12.2 Terminology Encountered while Using Heat Service:	25
1.7.13 Senlin (Clustering)	26
1.7.13.1 Components of Senlin:	26
1.8 Theory of Autoscaling in OpenStack	28
1.8.1 The main components of autoscaling:	28
1.9 Need for auto-scaling in private clouds	29
1.9.1 Dynamic Workloads	29
1.9.2 Cost Efficiency	29
1.9.3 Improved performance	29
1.9.4 Resource Optimization	29
1.9.5 Enhanced Availability and Reliability	30
1.9.6 Efficient Resource Planning	30
1.9.7 Quick Response to Unforeseen Events	30
1.10 Monitoring and visualizing OpenStack resources	30
1.10.1 Prometheus	30
1.10.2 Grafana	31
1.10.3 OpenStack-exporter	32
Chapter 2	34
RESEARCH PROBLEM	34
Chapter 3	35
RESEARCH OBJECTIVES	35
Chapter 4	36
LITERATURE REVIEW	36
4.1 Studies related to analyzing and comparing the autoscaling solutions in the domain of cloud	36
4.2 OpenStack-based autoscaling	40

4.2.1	Auto-Scaling Using Heat	40
4.2.2	Auto-Scaling Using Heat and Ceilometer.....	41
4.2.3	Auto-Scaling Using Heat and Monasca	43
4.2.4	Auto-Scaling Using Senlin.....	43
4.3	Non-OpenStack-based autoscaling	44
Chapter 5	46
METHODOLOGY	46
5.1	Install OpenStack private cloud	46
5.2	Install and configure monitoring setup	49
5.2.1	Installing Prometheus.....	49
5.2.2	Installing Grafana.....	50
5.2.3	Installing OpenStack-exporter	51
5.2.4	Integration of Prometheus, Grafana, and OpenStack-exporter	54
5.3	Deploy and prepare the VM.....	55
5.4	Implementation of autoscaling solutions	57
5.4.1	Heat, ceilometer and Gnocchi	57
5.4.2	Senlin, ceilometer and Gnocchi	57
5.5	Workload pattern generation.....	58
Chapter 6	60
RESULTS AND DISCUSSIONS	60
Chapter 7	67
CONCLUSION AND FUTURE WORK	74
REFERENCES	76
APPENDIX A	86
Heat implementation	86
APPENDIX B	89
Senlin implementation	89

LIST OF FIGURES

Figure	Description	Page
Figure 1.1	Server stack comparison between on-premises Infrastructure IaaS, PaaS, and SaaS	3
Figure 1.2	Static workload pattern	11
Figure 1.3	High growth workload pattern	11
Figure 1.4	On-Off workload pattern	12
Figure 1.5	Aperiodic bursting workload pattern	12
Figure 1.6	Periodic bursting workload pattern	13
Figure 1.7	High growth-static-high shrink workload pattern	14
Figure 1.8	High growth-static-high shrink on-off with no interval workload pattern	14
Figure 1.9	High growth-static-high shrink on-off with interval workload pattern	15
Figure 1.10	High growth on-off workload pattern	15
Figure 1.11	High growth High shrink workload pattern	16
Figure 1.12	Ceilometer design	18
Figure 1.13	Ceilometer event notification	19
Figure 1.14	Ceilometer data processing and storage	19
Figure 1.15	Ceilometer high-level architecture	20
Figure 1.16	Octavia architecture	24
Figure 1.17	Heat architecture	26
Figure 1.18	Senlin architecture	27
Figure 1.19	OpenStack system architecture	27
Figure 1.20	OpenStack autoscaling conceptual diagram	28
Figure 1.21	Grafana dashboard	32
Figure 1.22	Prometheus Architecture	32
Figure 1.23	Monitoring setup used in this project	33
Figure 4.1	Auto-scaling in Heat	41
Figure 4.2	Alarm generation and assessment in Ceilometer	42
Figure 5.1	Host OS information	46

Figure 5.2	local.conf file content	47
Figure 5.3	OpenStack installation completed	48
Figure 5.4	OpenStack services verification	48
Figure 5.5	OpenStack Horizon dashboard	48
Figure 5.6	Installing Prometheus	49
Figure 5.7	Verification of Prometheus Installation	49
Figure 5.8	Prometheus targets and ports	49
Figure 5.9	Scraped metrics by Prometheus	50
Figure 5.10	Verification of Grafana Server Installation	51
Figure 5.11	Grafana Dashboard	51
Figure 5.12	Pulling the latest OpenStack-exporter from GitHub	52
Figure 5.13	Verification of Docker image download	52
Figure 5.14	The content of cloud.yaml file	52
Figure 5.15	OpenStack-related metrics	53
Figure 5.16	Status of the custom build system service	54
Figure 5.17	Exporters are integrated with Prometheus	54
Figure 5.18	Prometheus data source is added in Grafana	54
Figure 5.19	Dashboards are configured to visualize data related to relevant exporter data	55
Figure 5.20	Node-exporter data visualization	55
Figure 5.21	OpenStack-exporter data visualization	55
Figure 5.22	Web server integration to Prometheus	56
Figure 5.23	Web server metric visualization using Grafana	56
Figure 5.24	Web server instance on Horizon dashboard	57
Figure 5.25	Heat-based Autoscaling Implementation	57
Figure 5.26	Senlin-based Autoscaling Implementation	58
Figure 5.27	Apache JMeter workload generation	59
Figure 5.28	The overall architecture of the implementation	59
Figure 6.1	CPU usage during the test1-Heat	60
Figure 6.2	Network Bandwidth usage during the test1-Heat	61
Figure 6.3	Behavior of VM creation/deletion during the test1-Heat	61
Figure 6.4	Statics of the web requests during the test1-Heat	61

LIST OF TABLES

Table	Description	Page
Table 1.1	Examples of step scaling rules	8
Table 1.2	Overview of key auto-scaling solutions offered by major public cloud providers	9
Table 4.1	Workloads used to evaluate autoscaling solutions in simulated experiments	38
Table 4.2	Autoscaling approaches for a simulated system across various workloads to manage APPDET's response time	38
Table 4.3	Performance comparison based on the amount of QOS violations	39
Table 4.4	Performance for AWS, AZURE, and GCP autoscaling solutions for scale-out events	39
Table 6.1	Summary of the results for all workloads when using Heat	62
Table 6.2	Summary of the results for all workloads when using Senlin	63
Table 6.3	MCDA analysis of workload 1	64
Table 6.4	MCDA analysis of workload 2	64
Table 6.5	MCDA analysis of workload 3	65
Table 6.6	MCDA analysis of workload 4	65
Table 6.7	MCDA analysis of workload 5	65
Table 6.8	MCDA analysis of workload 6	66
Table 6.9	MCDA analysis of workload 7	66
Table 6.10	MCDA analysis of workload 8	66
Table 6.11	MCDA analysis of workload 9	67
Table 6.12	MCDA analysis of workload 10	67
Table 6.13	The better solution for each workload based on MCDA analysis	67

LIST OF ABBREVIATIONS

Abbreviation	Description
VM	Virtual Machine
OS	Operating System
AWS	Amazon Web Services
GCE	Google Compute Engine
GCP	Google Compute Provider
VPS	Virtual Private Server
CMS	Content Management System
URL	Uniform Resource Locator
CPU	Central Processing Unit
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory network
AR	Auto Regressive
ARMA	Auto Regressive Moving Average
ES	Exponential Smoothing
API	Application Programming Interface
MQ	Message Queue
SLA	Service Level Agreement
MIMO	Multi-Input, Multi-Output
ML	Machine Learning
RT	Response Time
QPS	Queries Per Second
RDBMS	Relational Database Management System
MCDA	Multi-Criteria Decision Analysis

LIST OF APPENDICES

Appendix	Description	Page
Appendix A	Heat Implementation	80
Appendix B	Senlin Implementation	83