

**A SITE RELIABILITY ENGINEERING FRAMEWORK FOR BENCHMARKING
SAAS PLATFORM PERFORMANCE**

By

Matthews Wong
12202010

BACHELOR'S DEGREE
in

Information Technology
Faculty of Engineering and Information Technology



SWISS GERMAN UNIVERSITY
The Prominence Tower
Jalan Jalur Sutera Barat No. 15, Alam Sutera
Tangerang, Banten 15143 - Indonesia

May 2026

STATEMENT BY THE AUTHOR

I hereby declare that this submission is my own work and to the best of my knowledge, it contains no material previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma at any educational institution, except where due acknowledgement is made in the thesis.



Matthews Wong

Student

12 Juni 2026

Date

Approved by:



Dr. Maulahikmah Galinium, S.Kom., M.Sc

Thesis Advisor

18.6.2026

Date

Aditya Fidiyanto, S.T



Thesis Co-Advisor

12-Juni-2026

Date



Dr. Maulahikmah Galinium, S.Kom., M.Sc

Dean

18.6.2026

Date

ABSTRACT

A SITE RELIABILITY ENGINEERING FRAMEWORK FOR BENCHMARKING SAAS PLATFORM PERFORMANCE

By

Matthews Wong

Dr. Maulahikmah Galinium, S.Kom., M.Sc , Advisor

Aditya Fidiyanto , S.T, Co-Advisor

SWISS GERMAN UNIVERSITY

Ontego Traces, a logistics platform used by German delivery operators, processed tours in production with no performance baseline and no way to detect service degradation under load. This thesis built an SRE-aligned benchmarking framework and ran ten controlled experiments against the Tour Data Provider, the platform's primary data ingestion endpoint. Horizontal scaling helped less than expected. Adding a second service replica cut the 95th percentile response time from 8.4 s to 4.5 s but raised throughput only 9.3%, because the shared PostgreSQL database was the binding constraint. Three configuration changes (turning off write-ahead log synchronisation, enlarging the shared buffer cache, and widening the HikariCP connection pool) raised throughput by 32% and cut tail latency by 57% on a single replica, outperforming two-replica scaling on every metric. A live `'pg_stat_activity'` investigation during the tuning experiments revealed the hard ceiling: every Create Tour request locks the same row in the `'tenant_config'` table, serialising all concurrent writes in a way no infrastructure change can remove. Resolving it requires an application-level fix that is outside the infrastructure scope of this thesis. Beyond the load testing results, the platform gained a CPU-based autoscaling daemon for Docker Swarm, a Prometheus and Grafana monitoring stack collecting all four SRE golden signals, and a CI/CD quality gate that blocks production deploys when 95th percentile latency exceeds 5 seconds or the error rate exceeds 5%.

Keywords: Ontego Traces , Load Testing , Microservices Architecture, Autoscaling , Observability



DEDICATION

This work is dedicated to commault Indonesia, whose trust, openness, and willingness to support this research provided the environment and real-world platform that made this thesis possible.

To my parents, whose endless encouragement and unwavering support have carried me through every step of this journey — this work is as much yours as it is mine.



ACKNOWLEDGEMENTS

First and foremost, I am deeply thankful and beyond grateful for the wisdom and strength that God has graciously given me throughout this journey. This work would not have been possible without His guidance every step of the way.

I wish to express my sincere gratitude to Dr. Maulahikmah Galinium, S.Kom., M.Sc., my thesis advisor, for the continued guidance and support that shaped this research, and to Mr. Aditya Fidiyanto, S.T., my co-advisor from commsult Indonesia, for his valuable input and practical perspective. I also extend my deepest appreciation to Mr. Leonardo Kurnia for the opportunity to conduct this research at commsult Indonesia and for his mentorship throughout the process.

I wish to thank the Ontego Traces team — Mr. Vernando Cahyadi and Mr. Mikail Rasyid — for their time and willingness to share the knowledge necessary to ensure this research aligned with real requirements, as well as all colleagues at commsult Indonesia for their support and warm working environment.

I also wish to thank the youth organization of GBI House of Grace for the endless support and prayers that kept me encouraged throughout this process. Finally, to my parents — thank you for your countless prayers, unconditional love, and steadfast support. Your encouragement has been a constant source of strength throughout this journey, and knowing you were always behind me made every challenge easier to face and every milestone more meaningful to achieve.

TABLE OF CONTENTS

	Page
ABSTRACT.....	3
DEDICATION.....	5
ACKNOWLEDGEMENTS.....	6
Chapter 1: Introduction.....	14
1.1 Background.....	14
1.2 Problem Statement.....	15
1.3 Research Objectives.....	17
1.4 Research Questions.....	18
1.5 Hypothesis.....	18
1.6 Significance of the Study.....	19
1.7 Scope and Limitations.....	25
Chapter 2: Literature Review.....	27
2.1 Theoretical Perspectives.....	27
2.1.1 Site Reliability Engineering and the Definition of Performance.....	27
2.1.2 Software-as-a-Service Platforms.....	28
2.1.3 Performance Benchmarking and the Definition of a "Framework".....	29
2.1.4 Load Testing.....	32
2.1.5 Container Orchestration and Auto-scaling.....	34
2.1.6 Monitoring and Observability.....	38
2.1.7 CI/CD Pipeline Integration.....	42
2.1.8 Database Layer Performance.....	45
2.1.9 Conceptual Framework.....	47
2.2 Previous Studies.....	48
Chapter 3: Research Methods.....	56
3.1 Methodology Overview.....	56
3.2 Requirements Gathering.....	58
3.3 Tech Selection.....	59
3.4 Environment Setup.....	59
3.4.1 Server Provisioning.....	60
3.4.2 Dual-Host Topology.....	61

3.4.3 Application Stack Services.....	62
3.5 Auto-Scaler and Monitoring Stack Design.....	63
3.6 Load Test Scenario Design.....	66
3.7 Data Collection Procedure.....	71
3.8 Bottleneck Diagnosis.....	73
3.9 Iterative Optimisation.....	74
3.10 Validity and Reliability.....	78
3.10.1 Internal Validity.....	78
3.10.2 External Validity.....	78
3.10.3 Reliability.....	79
Chapter 4: Results and Discussions.....	80
4.1 Test Environment and Toolchain.....	80
4.2 Load Test Script Design.....	81
4.2.1 Payload.....	81
4.2.2 Per-Iteration Variable Generation.....	82
4.2.3 OAuth2 Authentication.....	83
4.2.4 HTTP Headers.....	84
4.2.5 Load Profile.....	84
4.3 Smoke Test — Script and Endpoint Validation.....	85
4.3.1 Real Tour Data — Evidence of Realistic Simulation.....	86
4.4 Comparative Load Test: 1 Replica vs 2 Replicas (Stage 5 Baseline).....	87
4.4.1 Procedure.....	87
4.4.2 Results.....	88
4.4.3 Interpretation.....	89
4.5 Artemis Queue Throughput Measurement (Stage 5 Queue Diagnostic).....	90
4.5.1 Measurement Approach.....	90
4.5.2 Results.....	92
4.5.3 Interpretation.....	92
4.6 Bottleneck Analysis (Stage 5 — DB Identified).....	93
4.6.1 Why Tail Latency Improved but Queue Throughput Did Not.....	93
4.6.2 The Three Bottleneck Scenarios.....	96
4.6.3 Summary.....	97
4.7 Database Configuration Tuning (Stage 6 — Iteration 1).....	97
4.7.1 Baseline Configuration.....	97

4.7.2 Tuning Changes Applied.....	99
4.7.3 Post-Tuning Results.....	101
4.7.4 Interpretation.....	101
4.7.5 What Tuning Can and Cannot Fix.....	103
4.8 Hot-Row Investigation (Stage 6 — Iteration 2).....	104
4.8.1 Identifying the Hot Row via pg_stat_activity.....	104
4.8.2 Pool Size Sweep — Finding the Least-Bad Configuration.....	106
4.8.3 Why Two Replicas Cannot Win at the Infrastructure Level.....	108
4.8.4 Infrastructure Ceiling.....	110
4.9 PgBouncer Connection Pooler Experiment (Stage 6 — Iteration 3).....	110
4.9.1 PgBouncer Architecture and Configuration.....	111
4.9.2 Interpretation.....	114
4.9.3 Auto-Scaler Result.....	115
4.9.4 Final Result.....	117
Chapter 5: Conclusions and Recommendations.....	120
5.1 Conclusions.....	120
5.1.1 H1 — Load Characterisation: Confirmed.....	120
5.1.2 H2 — Autoscaling Effectiveness: Confirmed.....	120
5.1.3 H3 — Observability Sufficiency: Confirmed.....	121
5.1.4 H4 — Pipeline Automation: Confirmed.....	121
5.1.5 Hypothesis Summary.....	122
5.1.6 What the Framework Proves and Does Not Prove.....	123
5.2 Recommendations.....	123
5.2.1 Recommendations for commsult.....	123
5.2.2 Recommendations for Future Research.....	125
5.3 Implementation Roadmap.....	127
GLOSSARY.....	129
REFERENCES.....	130
CURRICULUM VITAE.....	137

LIST OF FIGURES

Figures	Page
Figure 1.1: Uptime Kuma — GLC Traces Tracking Data Provider (/tdp/actuator/health).....	16
Figure 1.2: Four-Pillar Research Framework.....	20
Figure 1.3: Bottleneck Diagnosis and Auto-scaling Decision Tree.....	21
Figure 1.4: Performance Quality Gate Inserted Between Existing Staging Deploy and Existing Manual Production Stage.....	23
Figure 1.5: Performance Quality Gate Sequence — Post-Deploy k6 Smoke Test with Live Metric Streaming.....	24
Figure 1.6: GDPR Test Data Preparation Flow — From Production Log Capture to Anonymised Mock Payload.....	25
Figure 2.1: k6 Load Test Flow — From GDPR-Safe Payload to Real-Time SLI Metrics.....	34
Figure 2.2: Auto-Scaling Decision Loop — CPU-Based Scaling Daemon for Docker Swarm	37
Figure 2.3: Observability Stack Architecture — Four Golden Signals® on a Unified Dashboard	42
Figure 2.4: Bitbucket Pipeline Flow — SLO Quality Gate Appended to the Existing Stage-Release Pipeline.....	45
Figure 2.5: Conceptual Framework — Three Theoretical Pillars and Four Operational Practices	48
Figure 3.1: Nine-Stage Methodology Workflow.....	56
Figure 3.2: Dual-Host Topology — GCP Staging and DigitalOcean Load Test Engine.....	62
Figure 3.3: Monitoring Stack Data Flow — InfluxDB Push and Prometheus Federation.....	64
Figure 3.4: Auto-Scaler Decision Loop — Asymmetric Streak Consensus with Cooldown...66	
Figure 3.5: k6 Load Test Script Flow.....	69
Figure 3.6: CI/CD Quality Gate — SLO Smoke Test Appended to the Stage-Release Pipeline	70
Figure 3.7: Standard Experiment Procedure.....	72
Figure 3.8: Iterative Optimisation Feedback Loop.....	75
Figure 4.1: Tour Data Provider Endpoint Under Test.....	81
Figure 4.2: Per-Iteration Variable Generation Helpers — randInt and tourDateNow.....	82
Figure 4.3: k6 setup() — One-Time OAuth2 Token Acquisition from Keycloak.....	83

Figure 4.4: k6 Request Parameters and HTTP Headers.....	84
Figure 4.5: k6 options — Staged Load Profile for the Auto-Scaler Demonstration.....	85
Figure 4.6: Smoke Test Command — Single VU, Single Iteration End-to-End Validation...	85
Figure 4.7: Ontego Traces Web App Tour List After the Full Experiment Suite.....	87
Figure 4.8: Docker Swarm Service Scale Commands — Toggling Between 1 and 2 TDP Replicas.....	88
Figure 4.9: Throughput and p95 Latency — 1 Replica vs 2 Replicas (Untuned).....	90
Figure 4.10: Artemis Queue Stat — CLI Sampling Command.....	91
Figure 4.11: Artemis Queue Throughput — 1 Replica vs 2 Replicas.....	93
Figure 4.12: Bottleneck Topology — Where Scaling Helps and Where It Does Not.....	95
Figure 4.13: pg_stat_activity Baseline — Pre-Tuning Connection Inventory and Result.....	99
Figure 4.14: PostgreSQL Tuning — ALTER SYSTEM Commands for synchronous_commit, wal_buffers, and shared_buffers.....	99
Figure 4.15: HikariCP Connection-Pool Configuration — Pool Sized 5 → 10 per Replica.	101
Figure 4.16: Bottleneck Shift Across the Three Tuning Scenarios.....	102
Figure 4.17: Throughput and p95 Across Four Tuning Configurations.....	103
Figure 4.18: Query to identify Blocked Sessions.....	105
Figure 4.19: Result Blocked Sessions on the Hot Row via pg_locks and pg_stat_activity...	105
Figure 4.20: Hot-Row UPDATE Statement on tenant_config Issued by Every Create Tour Request.....	105
Figure 4.21: HikariCP Pool Size Sweep.....	108
Figure 4.22: PgBouncer Topology — Two TDP Replicas Multiplexed Through Transaction-Mode Pool.....	111
Figure 4.23: PgBouncer pgbouncer.ini — Transaction-Mode Pool Configuration.....	111
Figure 4.24: PgBouncer vs Direct PostgreSQL Connection.....	113
Figure 4.25: PgBouncer Per-Query Latency Overhead (Idle Path).....	113
Figure 4.26: Complete Experiment Summary — Throughput and p95 Ranking.....	119
Figure 5.1: Implementation Roadmap — Three-Phase Adoption at commsult.....	127

LIST OF TABLES

Table	Page
Table 2.1: Comparison of Open-Source Load Testing Tools (adapted from Sotomayor et al., 2023).....	33
Table 2.2: Comparison of Container Orchestration Platforms (adapted from Pahl et al., 2019; Usman et al., 2022).....	36
Table 2.3: Comparison of Monitoring and Observability Solutions (adapted from Beyer et al., 2016).....	40
Table 2.4: Comparison of CI/CD Platforms (adapted from Humble and Farley, 2010).....	44
Table 2.5: Comparison of Related Research, Including the Present Thesis (Wong, 2026).....	50
Table 3.1: Research Methodology Classification.....	57
Table 3.2: GCP Staging Host Specifications.....	60
Table 3.3: DigitalOcean Load Test Engine Specifications.....	61
Table 3.4: Application Stack Services (otg-traces-stage).....	62
Table 3.5: Experiment Variables.....	66
Table 3.6: Load Profile Rationale — Constraints Observed in the Pilot Phase.....	67
Table 3.7: Load Profiles Used in the Thesis.....	68
Table 3.8: Primary Metrics and Their Sources.....	69
Table 3.9: Ten Experiment Scenarios and Their Iteration Origin.....	77
Table 4.1: Chapter 4 Roadmap Mapped to the Nine-Stage Methodology.....	80
Table 4.2: Per-Iteration Variables Generated by the k6 Script.....	82
Table 4.3: Smoke Test Results — Checks and Metrics.....	85
Table 4.4: HTTP Performance — 1 Replica vs 2 Replicas (10 VUs · 120 s).....	88
Table 4.5: Artemis Queue Metrics Used for Diagnostics.....	91
Table 4.6: Artemis glc_traces_tour Queue — 1 Replica vs 2 Replicas (10 VUs · 120 s).....	92
Table 4.7: Bottleneck Scenario Summary.....	97
Table 4.8: Pre-Tuning Configuration.....	98
Table 4.9: PostgreSQL Tuning Rationale.....	100
Table 4.10: Four-Scenario Comparison — Untuned and Tuned (10 VUs · 120 s).....	101
Table 4.11: Database Tuning Effectiveness by Bottleneck Type.....	103
Table 4.12: Two-Replica Pool Size Sweep (10 VUs · 120 s).....	106
Table 4.13: Complete Experiment Summary — All Scenarios.....	108

Table 4.14: PgBouncer vs Direct Connection (10 VUs · 120 s).....	112
Table 4.15: Auto-Scaler Event Summary — Staged 9-Minute Profile.....	115
Table 4.16: Auto-Scaler vs Manual Scaling — Throughput Ceiling Comparison.....	116
Table 5.1: Hypothesis Verdict Summary.....	122

