Master Thesis

Title: Predictive Auto-Scaling for Cloud Applications,

Machine Learning Driven Framework using E-Commerce Data Workloads

Master of Science
Applied Computer Science

Chaithra Jagannatha Rao Telkar
11037491

05 November 2025

Supervisors

Prof. Dr. Gerd Moeckel
Prof. Paul Tanzer

Agenda

Introduction to Cloud Predictive Auto-Scaling

01

Introduction to Cloud Predictive Auto-Scaling

What is Auto-Scaling?

Automatic adjustment of computing resources based on system load.

Traditional scaling reacts *after* overload.

Why Predictive Auto-Scaling?

Predictive scaling *foresees* load and adjusts *before* performance drops.

Motivation and Problem Definition

02

Motivation

Motivation

- Online retail workloads fluctuate rapidly during events (sales, campaigns).
- Delayed scaling causes latency and revenue loss.
- Predictive auto-scaling ensures proactive resource readiness.
- Machine learning enables workload forecasting and uncertainty estimation.

Research Objectives

- **Delayed Scaling:** Reactive methods adjust resources only *after* overload, increasing latency and harming user experience.
- **Resource Inefficiency:** Over-provisioning wastes cost; under-provisioning slows performance.
- **No Adaptive Forecasting:** Traditional approaches fail to learn complex, nonlinear workload patterns.
- **No Uncertainty Handling:** Existing models ignore prediction uncertainty, causing unstable scaling decisions.

Tools and Technology

03

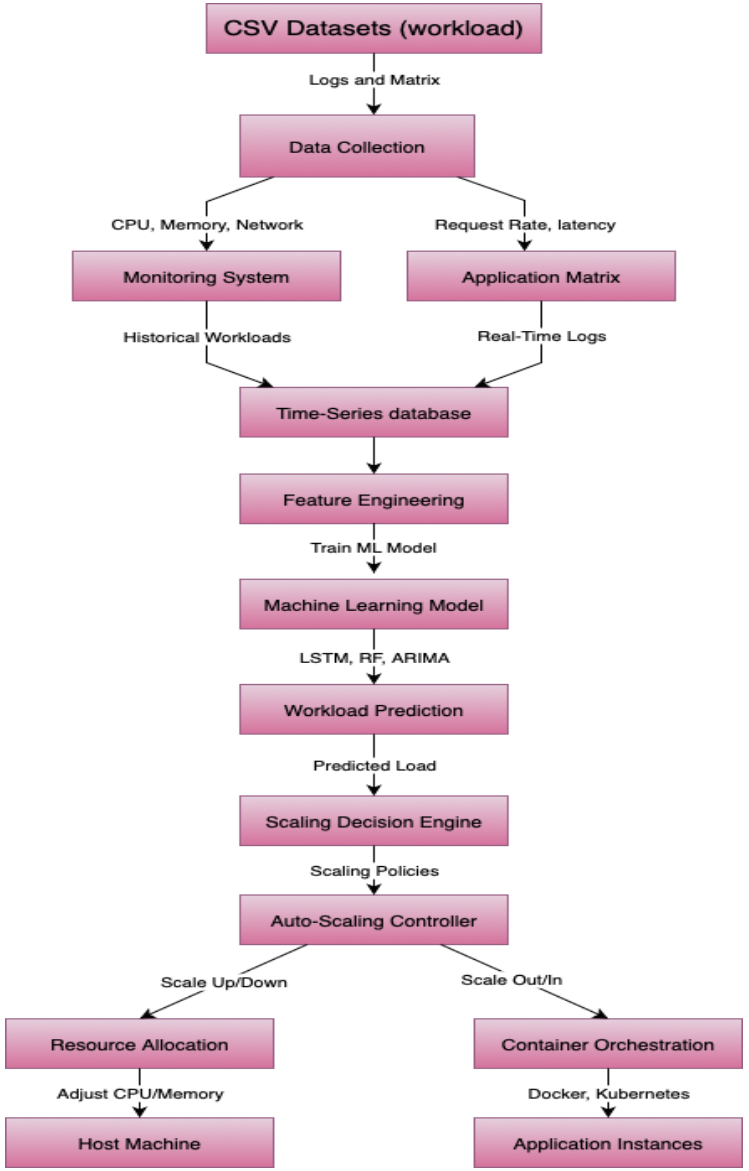
Tools and Technology Used

- **Dataset:** Alibaba Cluster Trace 2018 (real cloud workload data)
- **Languages:** Python (for ML pipeline), Node.js (for controller)
- **Libraries:** pandas, numpy, scikit-learn, TensorFlow/Keras
- **Monitoring Stack:**
  - Prometheus (metrics collection)
  - Alertmanager (notifications)
- **Containerization:** Docker & Docker Compose

Proposed Architecture diagram

04

Architecture diagram



Data Input: Alibaba 2018 workload CSVs provide real historical logs such as CPU, memory, network, and execution time, used to understand real cloud behavior.

Monitoring & Preprocessing: These raw logs are collected, cleaned, and turned into continuous time-series data for model training.

Machine Learning Core: Models like LSTM, Random Forest, and ARIMA learn workload patterns and predict future latency (p95) with uncertainty.

Decision & Scaling: The controller applies the predictive rule mean + σ > SLO → Scale Up, mean + σ < SLO → Scale Down to manage resources proactively.

Execution Layer: The system automatically adjusts CPU/memory or container replicas through Docker/Kubernetes, keeping latency low and cost efficient.

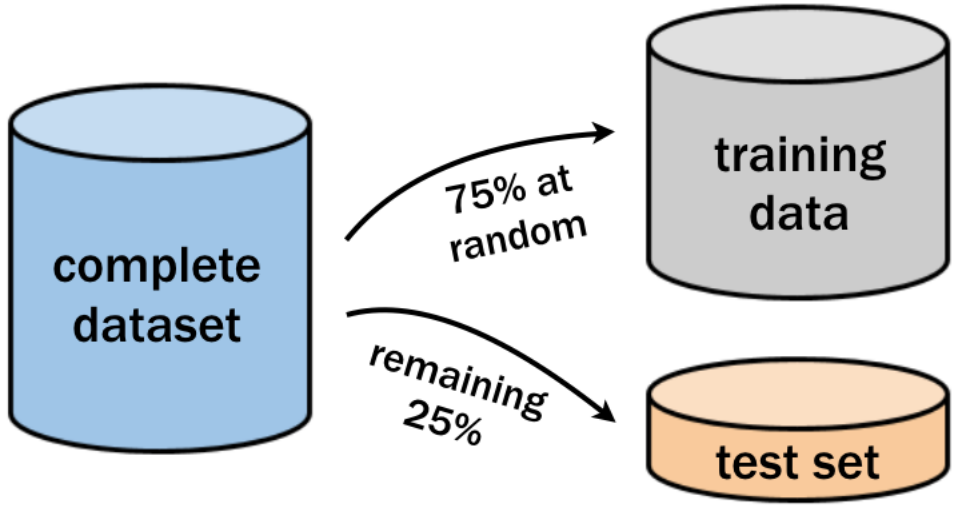
Dataset and Preprocessing

05

Dataset and Preprocessing

Dataset: Alibaba Cluster Trace (2018)

- Real traces from Alibaba's e-commerce cloud infrastructure.
- Contains CPU, memory, job, and container usage logs.
- Sampled 2000 rows for modeling and evaluation.
- Reflects realistic e-commerce resource dynamics.



Preprocessing:

- Loaded and time-sorted Alibaba CSV data by timestamp.
- Output → clean csv data used by pipeline.
- Aligned timestamps and handled missing values.
- Generated latency target (response_time_p95) using rolling 95th percentile of execution time.
- Added lag and rolling features for CPU, memory, network, and execution time.

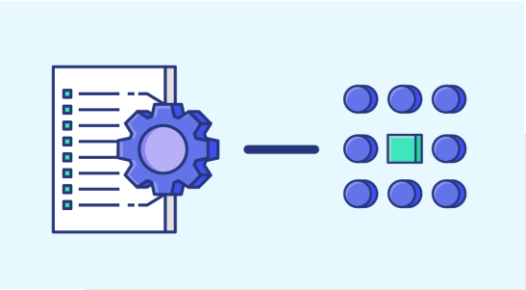
Feature Engineering and Models

06

Feature Engineering and Models

Feature engineering:

- Converted all metrics to numeric time-series.
- Added lag features (t-1, t-2, t-3) for workload history.
- Added rolling averages (3 & 5 points) for CPU, memory, and network.
- Created response_time_p95 as latency target.
- Removed missing data and scaled features using MinMaxScaler.



Models:

- **LSTM:** Learns time-based workload patterns to predict future latency (response_time_p95), Monte Carlo dropout gives uncertainty (σ).
- **Random Forest:** Predicts future latency and provides quantile predictions (p50, p90, p95) for scaling decisions.
- **ARIMA:** Baseline time-series model that forecasts latency using past trends.

Predictive Decision Rule

07

Predictive Decision Rule

Formula:

mean + σ > SLO → Scale Up

mean + σ < SLO → Scale Down

Interpretation:

- **mean:** Predicted future latency
- **σ:** Prediction uncertainty (safety margin)
- **SLO:** Latency limit (e.g., 0.09 s)

Why It's Unique:

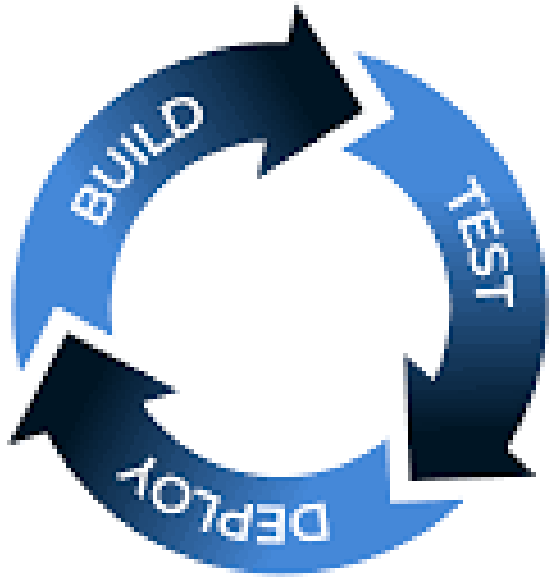
- Adds **risk awareness** using prediction uncertainty.
- **Proactively scales** before performance drops

Implementation of the Pipeline

08

Implementation Pipeline

- **Training Pipeline:** Runs ML workflow → outputs predictive_signal.json.
- **Controller:** Reads predictive signal and applies scaling rule (mean + σ > SLO).
- **Prometheus:** Continuously scrapes system metrics from /metrics endpoint.
- **Docker Compose:** Orchestrates all components (ML, controller, monitoring stack).



Results and Evaluation

09

Model Evaluation

Models trained: LSTM, Random Forest, ARIMA.

Metrics used:

- MAE (Mean Absolute Error)
- RMSE (Root Mean Square Error)

Observation:

LSTM achieved the most accurate time-series prediction of latency (response_time_p95).

[LSTM] MAE=0.004 RMSE=0.006

```
Evaluation started ...
[LSTM] MAE=0.0040  RMSE=0.0051  N=1976
[RF] MAE=0.0002  RMSE=0.0004  N=1996
[ARIMA] MAE=0.0046  RMSE=0.0058  N=1995
Evaluation completed ✅
```

Auto-Scaling Controller Logs

The controller read the predictive signal and **initiated scaling** when:

mean + σ > SLO

```
chaithratelkar@Mac cloud1 % docker logs --since=30m --tail=500 -f cloud1-scaling-controll
er-1 \
  | egrep -E "predictive read|Initiating scaling|Scaling completed"

info: predictive read: metric=response_time_p95 mean=0.0928 sigma=0.0019 alpha=1 decision
=0.0946 threshold(SLO p95 (seconds))=0.09 {"service":"auto-scaling-controller","timestamp
":"2025-10-22T22:06:00.478Z"}
info: Initiating scaling to 3 replicas. Reason: predictive_response_time_p95: value=0.094
6 threshold=0.09 (mean=0.0928, σ=0.0019, α=1) {"service":"auto-scaling-controller","times
tamp":"2025-10-22T22:06:00.479Z"}
info: Scaling completed: 2 -> 3 replicas in 2.001s {"service":"auto-scaling-controller","
timestamp":"2025-10-22T22:06:02.482Z"}
```

mean=0.0928, sigma=0.0019, decision=0.0946 > threshold=0.09

Scaling completed: 2 → 3 replicas

Interpretation: The system predicted that latency will exceed SLO and scaled up *proactively* before degradation occurred.

Prometheus Visualization

Prometheus metrics confirmed multiple scaling decisions:

- Query: scaling_decisions_total
- Shows 8 scaling events over time.

Graph indicates system responsiveness and predictive scaling trend.

Conclusion & Future Outlook:

10

Conclusion

Developed a **predictive auto-scaling framework** combining machine learning, observability, and intelligent control to maintain optimal cloud performance.

Integrated a **complete data pipeline** using **LSTM, Random Forest, and ARIMA** models to forecast **p95 response time** and guide scaling actions.

Implemented a **safety-aware decision rule** (mean + σ > SLO) for proactive scaling, with tunable **α** and **SLO_P95_SEC** for balancing reliability and efficiency.

Ensured **transparency and traceability** through **Prometheus metrics and controller logs**, proving a scalable and reproducible approach for modern cloud systems.

Future Outlook

Integrate **multi-metric prediction** (CPU, memory, latency) jointly.

Implement **reinforcement learning-based scaling** for adaptive threshold tuning.

Extend to **Kubernetes cluster-level scaling** for production use.

Demo

Q & A

Thanks for your attention
Danke für Ihre Aufmerksamkeit..