Improving Cold start in Serverless computing using Workload prediction

Submitted in partial fulfilment of the requirements of the degree of

Bachelor of Technology (B.Tech)
by

Anish Agarwal (197110)
Vedant Gandhi(197187)
D. Sai Teja(197123)

Supervisor:

Dr. Rashmi Ranjan Rout

Department of Computer Science and Engineering

NIT Warangal

India

Acknowledgement

I would like to express my heartily gratitude towards Dr. Rashmi Ranjan Rout Sir for their valuable guidance, supervision, suggestions, encouragement and the help throughout the semester, for the completion of my project work. They kept me going when I was down and gave me the courage to keep moving forward.

I also want to thank evaluation committee for their valuable suggestions and conducting smooth presentation of the project.

Anish Agarwal
197110

Vedant Gandhi
197187

D. Sai Teja
197123

Declaration

I declare that this written submission represents my ideas, my supervisor's ideas and co-supervisor's ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Anish Agarwal	Vedant Gandhi	D. Sai Teja
197110	197187	197123

APPROVAL SHEET

This Dissertation Work entitled **"Improving Cold start in Serverless computing using Workload prediction"** by **Anish Agarwal (197110), Vedant Gandhi (197187), D. Sai Teja (197123),** is approved for

the degree of Bachelor of Technology (B.Tech).

Examiners

Supervisors

XXX (Supervisor)

XXX(Co-Supervisor)

Chairman

Name of Head CSE (HOD)

NIT Warangal

Certificate

This is to certify that the Dissertation work entitled **"Improving Cold start in Serverless computing using Workload prediction"** is a bonafide record of work carried out by **Anish Agarwal (197110), Vedant Gandhi (197187), D. Sai Teja (197123)**, submitted to the Dr. Rashmi Ranjan Rout of "Department of Computer Science and Engineering", in partial fulfilment of the requirements for the award of the degree of B.Tech at "National Institute of Technology, Warangal" during the 2022-2023.

(Signature)
Dr. Rashmi Ranjan Rout
Designation Supervisor
"Department of Computer Science and Engineering"

Abstract

Serverless computing has transformed the world of cloud-based and event-driven applications by allowing easy deployment of applications to the cloud and quick go-live service. The Function-as-a-Service (FaaS) model used in serverless computing enables developers to concentrate on their core business logic while the Cloud Service Provider (CSP) takes care of managing infrastructure, scaling the application, and handling software updates and other dependencies. Although the scale-to-zero feature of this model saves costs by removing idle functions from memory, it can lead to a delay in the cold start problem. The challenge is to reduce the cold start latency without using additional resources. One of the methods to do predictive analysis using time series data is using the Prophet model.

Keywords —- Serverless Computing, Cold Start Delay, Kubernetes, Time series forecasting, Prophet model

Contents

List of Figures

Chapter 1

Introduction

1.1 Serverless Computing

Serverless Computing is a cutting-edge Cloud Computing approach that has garnered considerable attention from both industry and academia due to its architecture. Despite the name, Serverless Computing does not imply that servers are absent; instead, it means that the user does not need to manage them.[4] In Serverless Computing, applications or micro-services are deployed as functions that execute when triggered by an event. Throughout the function's life-cycle, all operational overheads are managed by the cloud service provider (CSP). In Serverless Computing, the deployment of multi-tier applications is simplified for CSPs by using the concept of inter-dependent functions.[15]

1.2 Advantages and Drawbacks

1.2.1 Advantages

Serverless Computing offers numerous benefits, including the absence of back-end operational overheads. This means that the Cloud Service Provider (CSP) manages all the back-end-related tasks, such as server management, auto-scaling, and load-balancing. Consequently, users can concentrate solely on developing the business logic. In addition, the pricing model is granular and charged in increments of milliseconds. This flexibil-

ity in service-deployment has resulted in a significant shift of users toward serverless computing in recent times. It is especially prevalent in applications such as deep learning, block-chain, the Internet of Things, and big data analytic. [3]

1.2.2 Drawbacks

Serverless Computing has several performance drawbacks too, among which the Cold-start problem is a major concern. The Cold-start problem is related to the function life-cycle, which begins when an event triggers it. In serverless, containers are created according to the function's requirements, followed by the setup of the environment and network according to the function's image file. The source code files of the function are then loaded, and the function begins execution. The period between the triggering of the function and the start of its execution is known as containerization, and the time required for it is called the cold-start time. The challenge of minimizing this cold-start time is known as the cold-start problem. [13]

1.3 Objectives

- To reduce the number of cold start occurrence in serverless computing. This can be done by leveraging machine learning algorithms to predict future loads. Based on these predictions, pre-warmed containers can be maintained to reduce the likelihood of cold starts.
- Comparing the result with the existing autoscaling policy and proposing a new prediction based autoscaler.

Chapter 2

Background

2.1 Cold Start Problem

Serverless computing is dependent on containers, where each function is triggered, and a new container is created to deploy the function with its dependencies. In situations where multiple requests are made to execute a function simultaneously, multiple instances of the function are created, each of which is independently deployed and executed on a separate container. [13]

When an invocation occurs from an event source, the following steps must be performed to execute the function:

1. Initialize a new container that includes preparing the container environment such as setting environment variables, connecting to the database, user authentication, etc.
2. Allocate memory and computing resources.
3. Load the function and required libraries.
4. Execute the function.
5. Return the results to the user and save the logs in the database.

While the scale-to-zero feature of serverless computing is beneficial for cost reduction, it poses a challenge known as the cold start problem. This is because each new invocation requires the repetition of all the above steps,

including the initialization of a new container, memory and computing resource allocation, library loading, and function execution. Moreover, if the number of simultaneous invocations exceeds the available containers, multiple instances of a function must be created, and each instance must go through these steps separately, resulting in a cold start delay at the first invocation and when the scale increases.

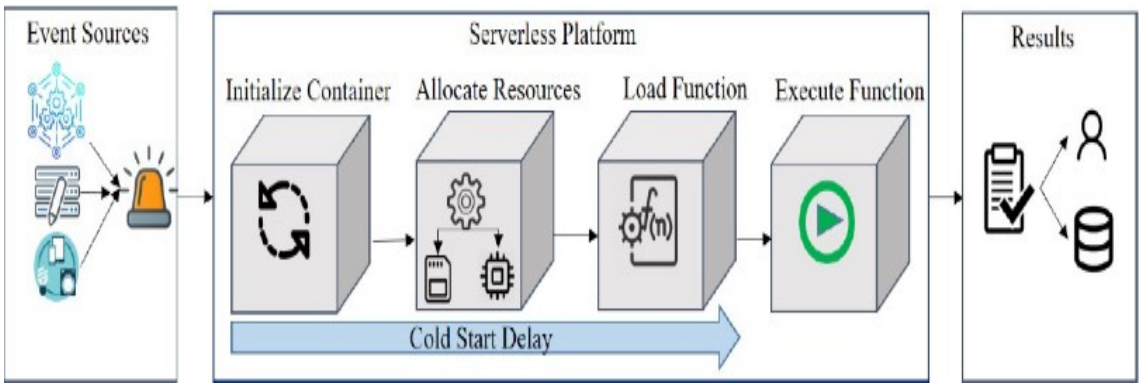

Figure 2.1: Faas execution flow

2.2 Characteristics of Serverless Computing

Function Programming, Function Serving, Pay-per-use billing system, Host-less, Lightweight, Statelessness, Burstiness.[4][2]

2.3 Challenges faced with the Serverless computing

Maintaining low Startup Latency, Scheduling Policy, Storage and Network facility, Improving Fault Tolerance.[15][2]

2.4 Open-source serverless frameworks

Currently, there are many open-source serverless frameworks (Apache Open-Whisk, Iron Functions, Oracle's Fn, Open-FaaS, Kubeless, Knative, Project Riff, etc.) that have been proposed to meet the increasing demands of serverless computing. Among these frameworks, some are built on top

of existing cloud service infrastructures (e.g., Kubernetes) to serve applications, while others are implemented from scratch by deploying the functions directly on servers.[4][2]

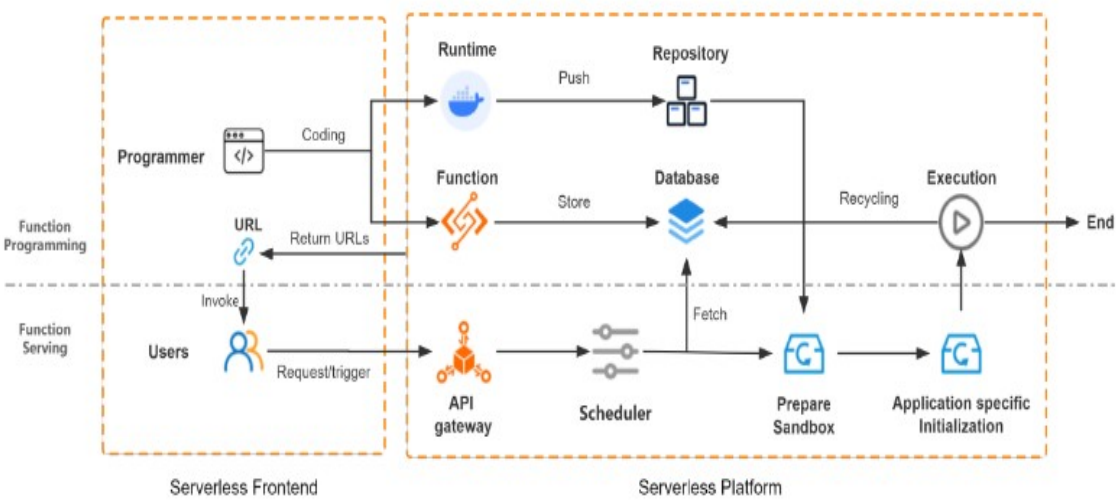

Figure 2.2: The whole process of Serverless Computing.

2.5 Edge Computing

The rise of Mobile Computing and Internet of Things (IoT) has led to the development of real-time and data-intensive applications. This has created challenges for centralized data centers and has led to the growing popularity of edge computing. Unlike traditional cloud data centers, edge nodes are distributed geographically, allowing them to be in close proximity to data prosumers and take advantage of wireless communication technologies and mobile networks. However, due to the limited resources of these densely distributed edge nodes, it is important to use them efficiently for hosted applications and services. Edge computing is a well-established network deployment method where services are executed as close to user devices as possible. This results in faster response times, alleviates user devices from computationally-intensive tasks, reduces energy consumption, and lowers traffic requirements compared to solutions where applications run fully on

user devices or in a remote data center.[6][12]

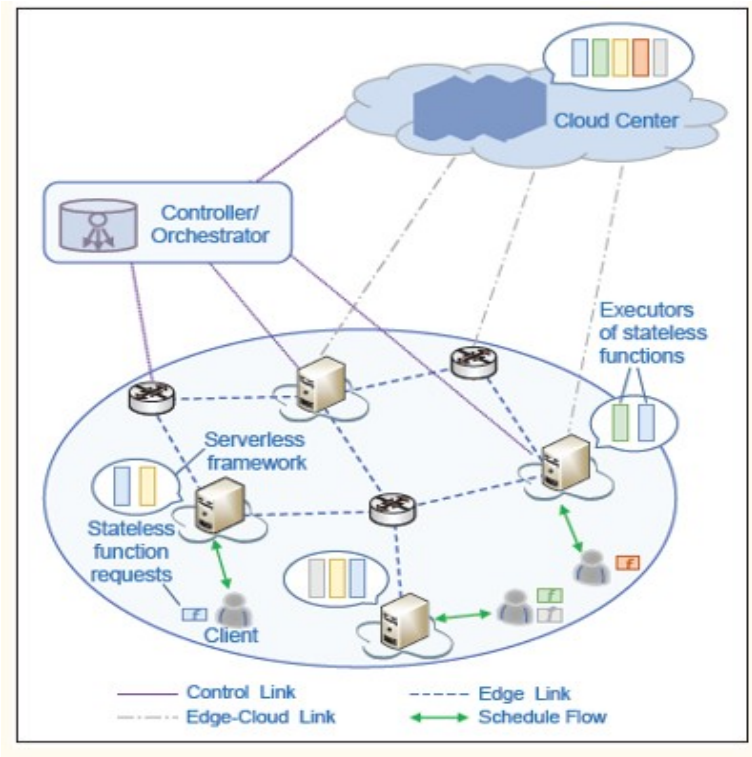

Figure 2.3: Architecture of Serverless Edge Computing Networks

Chapter 3

Related Work

Here you can see the way how you can cite the references.

3.1 Work done in reducing cold start delays

Several tools, including CloudWatch, Thundra.io, Dashbird.io, and Lambda Warmer, have been utilized to implement rules that periodically execute serverless functions and pre-warm them, for instance, every 5 minutes.

Agarwal et al. in [1] employed the Q-learning algorithm to determine the optimal number of function instances to maintain dynamically. They considered available function instances and discretized CPU consumption of each function as environmental states. The scaling up or down of function instances was then adjusted accordingly.

In [13] researchers explore the issue of cold starts in the Knative serverless platform and suggest a technique for migrating pods to reduce the occurrence of cold starts for function containers.

3.2 Work done in predicting future workloads using timeseries data

Albert W. Veenstra et. al. [14] have used Autoregressive time series model to predict the flow of trade of commodity markets on major trade routes. Results show that the model is able to estimate the long term seaborne trade

flow with small forecast errors.

Lim et al. [5] have proposed a mechanism based on control theory that utilizes an external feedback controller to extend cloud platforms and enable automatic allocation of computing resources for running applications. This approach also introduces the concept of proportional thresholding, a new control policy that considers the granular actuators provided by resource providers. However, the main challenge with control-based approaches is selecting the correct gain parameters for the model, as inadequate parameters may lead to system instability.

Mikolov et al [8] proposed a language-based model that leverages recurrent neural networks to allocate computing resources based on workload. The workload prediction is based on an ARIMA model and managed by a local infrastructure administrator.

Chapter 4

Proposed Solution

In this chapter, we are going to discuss the approach.

4.1 System overview

- Implement an ML model that predicts the number of future function invocations based on the past data.
- A learning-based time series forecasting model in Kubernetes autoscaler that, based on predicted number of future function invocations, allocates computing resources accordingly, making it possible to increase the containers before high function invocation traffic.
- Comparative analysis of our proposed model against HPA, that comes inbuilt with Kubernetes for the synthetic function workload pattern.[7]

4.2 Tools used

For this project we have used the following tools: Colab Notebook, python, Google cloud functions, Kubernetes, K6 load generator

4.3 Algorithms

Algorithm 1 Scaling algorithm

Initialize: $first\_inv = True$, $reqPerPod =$ int(sys.argv[1]), $initialDelay =$ int(sys.argv[2]), $interval =$ int(sys.argv[3]), $holidays[] =$ int(sys.argv[4])
for prediction in pred **do**
    if current_date is a holiday **then**
        $prediction+ = holidays[current\_date]$
    end if
    if $first\_inv == True$ **then**
        $prev\_pods = \lceil \frac{prediction}{reqPerPod} \rceil$
        $current\_pods = \lceil \frac{prediction}{reqPerPod} \rceil$
    else
        $current\_pods = \lceil \frac{prediction}{reqPerPod} \rceil$
    end if
    try
    Scale up the number of pods with $\max(current\_pods, prev\_pods)$
    $prev\_pods = \lceil \frac{prediction}{reqPerPod} \rceil$
    end try
    if exception caught **then**
        print("error")
    end if
    time.sleep$(interval - (initialDelay$ if $first\_inv$ else $0))$
    $first\_inv = False$
end for

4.3.1 PROPHET model

Prophet is a popular open-source time series forecasting tool developed by Facebook. It is designed to handle a wide range of time series data with multiple seasonality patterns and holiday effects. One of the key features of Prophet is its ability to incorporate the effects of holidays on the time series data.

To use the holiday parameter in Prophet, you first need to create a dataframe containing the dates of all the holidays that you want to model. The holidays can be of any frequency, such as daily, weekly, monthly, or yearly. Each row of the holiday dataframe should contain a column for the date and a column indicating the name of the holiday.

Once you have created the holiday dataframe, you can pass it as an argu-

ment to the Prophet model. The model will then include the holiday effects in the forecast by adding an additional component to the trend. Specifically, the model will create a binary variable for each holiday, indicating whether or not it falls on a given date. These binary variables are then included in the regression model as additional predictors.

By incorporating holiday effects into the model, Prophet is able to capture the impact of holidays on the time series data, such as changes in consumer behavior or changes in business operations during holiday periods. This can help improve the accuracy of the forecasts, especially for time series data with strong holiday effects.

Overall, the Prophet model with the holiday parameter is a powerful tool for time series forecasting, especially for data with multiple seasonalities and holiday effects.

4.3.2 SARIMA model

Seasonal Autoregressive Integrated Moving Average, SARIMA or Seasonal ARIMA, is an extension of ARIMA that explicitly supports univariate time series data with a seasonal component. It adds three new hyperparameters to specify the autoregression (AR), differencing (I) and moving average (MA) for the seasonal component of the series, as well as an additional parameter for the period of the seasonality. A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA. The seasonal part of the model consists of terms that are very similar to the non-seasonal components of the model, but they involve backshifts of the seasonal period.

4.3.3 LSTM model

LSTM stands for Long Short-Term Memory networks, used in the field of Deep Learning. It is a variety of recurrent neural networks (RNNs) that are capable of learning long-term dependencies, especially in Time series prediction problems. LSTM has feedback connections, i.e., it is capable of processing the entire sequence of data. LSTM can by default retain the information for a long period of time.For this reasons,it is used for processing,

predicting, and classifying on the basis of time-series data.

4.4 Flow diagrams

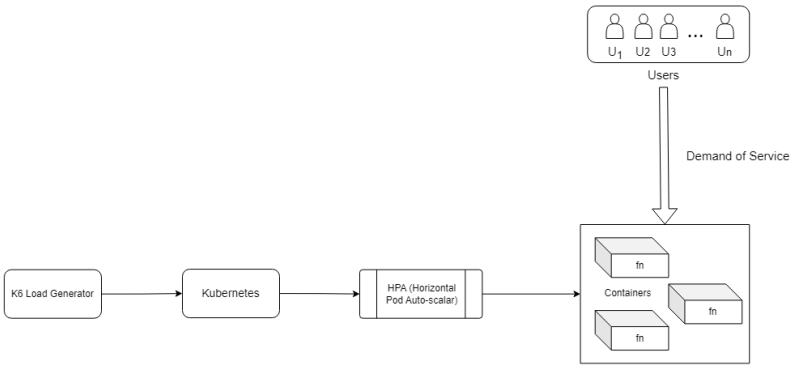

Figure 4.1: Traditional Kubernetes autoscaler approach

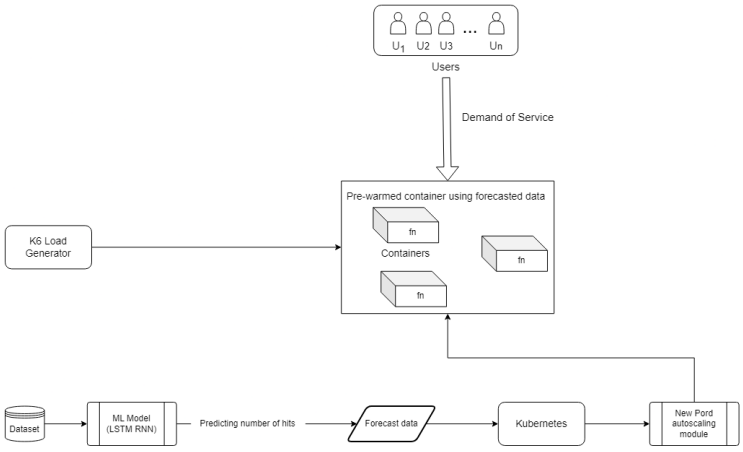

Figure 4.2: Custom autoscaler approach

Chapter 5

Experiments

5.1 Platform

For the simulation, we have used the python programming language. The python programming language consists of a set of libraries. It is used to simulate the network traffic pattern. For the experimental purpose, we have used the Google Colab notebook IDE.

5.2 Dataset[9]

In this work, a prediction model is used which predicts the average load on hourly basis of a distributed server for an interval of 24 hours. The input data chosen to train the forecasting model and evaluate our proposal were obtained from real web service logs from the Complutense University of Madrid. The dataset contains two attributes: holidays, timestamp and number of hits respectively.

For the experimental purpose, we have divided the data set in two parts. The first part of divided data has been used to train the system and the second part of data is used for testing the prediction. The testing part is used to test the accuracy of the predicted data.

We implemented various ML models i.e. Prophet model, SARIMA model and LSTM model.

5.3 Results of Training-testing ML models

We trained and tested the above mentioned ML models using the same dataset. A comparison between the predicted values of the models is provided below.

MODEL	MAE	MAPE
PROPHET(HOLIDAY)	525.61	26.45
PROPHET	650.82	35.67
LSTM	585.63	31.46
SARIMA	590.74	32.26

Figure 5.1: Comparison of results for ML models.

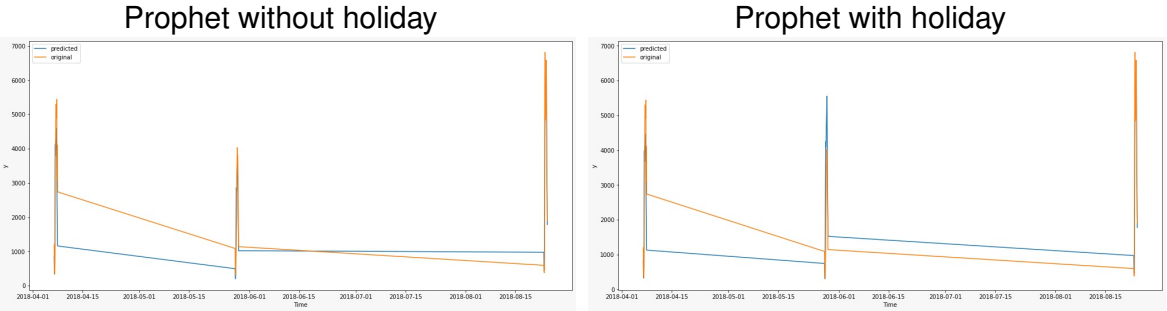

Figure 5.2: Comparison after using holiday feature.

As indicated by the above comparison, holiday plays a vital role in determining the traffic therefore a model such as Prophet that takes holidays into consideration is better for us.

Chapter 6

Conclusion and Future Work

- In this project, Prophet model [11] has been used to predict the future workload through the given data set. Results show the prediction accuracy of the model (i.e. minimum prediction error) is better than any other ML model. MAPE (Mean Absolute Percentage Error) and MAE (Mean Absolute Error) have been used to obtain the accuracy of the ML models.
- Prophet model takes into consideration an extra feature called Holidays for predicting the future function invocation traffic.
- Alongside, we have been learning about the working of Kubernetes and its components. We have also developed a static custom autoscalar for Kubernetes which scales the resources based on a simple mathematical expression.
- Using the knowledge of the static custom autoscalar we will be creating a dynamic prediction based autoscalar that will read metrics from the kubernetes metrics endpoint and use it to predict the future pods using our proposed algorithm.
- Lastly, we will compare the results for number of cold starts, resources used, latency with Horizontal Pods Autoscaler (HPA) of Kubernetes[10].

Bibliography

[1] Siddharth Agarwal, Maria A. Rodriguez, and Rajkumar Buyya. A reinforcement learning approach to reduce serverless function cold start frequency. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 797–803, 2021. 3.1

[2] Jaeeun Cho and Younghan Kim. A design of serverless computing service for edge clouds. In *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1889–1891, 2021. 2.2, 2.3, 2.4

[3] Akash Puliyadi Jegannathan, Rounak Saha, and Sourav Kanti Addya. A time series forecasting approach to minimize cold start time in cloud-serverless platform. In *2022 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pages 325–330, 2022. 1.2.1

[4] Yongkang Li, Yanying Lin, Yang Wang, Kejiang Ye, and Cheng-Zhong Xu. Serverless computing: State-of-the-art, challenges and opportunities. *IEEE Transactions on Services Computing*, pages 1–1, 2022. 1.1, 2.2, 2.4

[5] Harold Lim, Shivnath Babu, Jeffrey Chase, and Sujay Parekh. Automated control in cloud computing: Challenges and opportunities. *Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds, ACDC '09*, 06 2009. 3.2

[6] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716, 2019. 2.5

[7] Nima Mahmoudi and Hamzeh Khazaei. Performance modeling of metric-based serverless computing platforms. *IEEE Transactions on Cloud Computing*, pages 1–1, 2022. 4.1

[8] Tomas Mikolov, Martin Karafiát, Luká Burget, Jan Honza ernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, 2010. 3.2

[9] Available online. Dataset. (document), 5.2

[10] Available online. Kubernetes. 6

[11] Available online. Prophet ml model. 6

[12] Tri Thong Tran, Yu-Chen Zhang, Wei-Tung Liao, Yu-Jen Lin, Ming-Chia Li, and Huai-Sheng Huang. An autonomous mobile robot system based on serverless computing and edge computing. In *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 334–337, 2020. 2.5

[13] Parichehr Vahidinia, Bahar Farahani, and Fereidoon Shams Aliee. Mitigating cold start problem in serverless computing: A reinforcement learning approach. *IEEE Internet of Things Journal*, pages 1–1, 2022. 1.2.2, 2.1, 3.1

[14] Albert W Veenstra and Hercules E Haralambides. Multivariate autoregressive models for forecasting seaborne trade flows. *Transportation Research Part E: Logistics and Transportation Review*, 37(4):311–319, 2001. 3.2

[15] Renchao Xie, Qinqin Tang, Shi Qiao, Han Zhu, F. Richard Yu, and Tao Huang. When serverless computing meets edge computing: Archi-

tecture, challenges, and open issues. *IEEE Wireless Communications*, 28(5):126–133, 2021. 1.1, 2.3