Energy Aware Scheduling in Serverless Edge Computing

CS4099 Project Final Report

Submitted by

Falguni Biswas	**B200784CS**
Elsa Jibiachen	**B200049CS**
Divyang Jain	**B200826CS**

Under the Guidance of
Sudarshan Annadanam

Department of Computer Science and Engineering
National Institute of Technology Calicut
Calicut, Kerala, India - 673 601

May 3, 2024

NATIONAL INSTITUTE OF TECHNOLOGY
CALICUT, KERALA, INDIA - 673 601

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

2024

CERTIFICATE

Certified that this is a bonafide record of the project work titled

ENERGY-AWARE SCHEDULING IN SERVERLESS EDGE COMPUTING

done by

Falguni Biswas
Elsa Mary
Divyang Jain

of eighth semester B. Tech in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering of the National Institute of Technology Calicut

Project Guide
Sudarshan Annadanam
Assistant Professor

DECLARATION

I hereby declare that the project titled, **Energy Aware Scheduling in Serverless Edge Computing**, is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or any other institute of higher learning, except where due acknowledgement and reference has been made in the text.

Place : NIT Calicut
Date : 11-May-2024

Signature :
Name : Falguni Biswas
Reg. No. : B200784CS

Signature :
Name : Elsa Mary
Reg. No. : B200049CS

Signature :
Name : Divyang Jain
Reg. No. : B200826CS

Abstract

In this paper, we provide an energy-aware scheduling method for serverless edge computing environments.Energy awareness is critical since edge nodes,in many Internet of Things (IoT) domains, are meant to be powered by renewable energy sources that are variable, making low-powered and/or overloaded (bottleneck) nodes unavailable and not operating their services. We have a group of nodes connected through a controller, and all of them are powered by renewable energy sources. In this environment, we are following the Serverless Edge Computing paradigm, where applications are broken down into functions that interact with each other. To address fluctuations in user demand and enable dynamic scaling, we propose replicating functions and optimizing scheduling to minimize costs while maximizing operational availability.

ACKNOWLEDGEMENT

We would like to express our sincere and heartfelt thanks to Dr. Subhasree M, Head, Computer Science Engineering Department, NIT Calicut, for giving us the opportunity to work on this project. Our Project Coordinator Sudarshan Annadanam Sir deserves our heartfelt thanks for his diligent work in organizing the project milestones. We are also extremely grateful to our guide and mentor Venkat Ramireddy Sir for his active guidance, help, cooperation and encouragement throughout the thesis. Thanks to Raju Hazari Sir who also guided us through the project for the last year. Without their support, we would not have progressed with the project. we would not have progressed in the project. We thank our parents and teachers who encouraged and supported us in all our work. We also appreciate the help of the staff of CSE Department, NIT Calicut. In addition, we would like to thank our friends and seniors who cooperated with us during the project..

Contents

List of Tables

Chapter 1

Introduction

The world of computing has evolved from centralized cloud architectures to diverse models catering to different data processing and application deployment needs. One of the most significant of these models is Edge Computing, which relocates computational resources closer to the data source, thereby reducing latency and enabling real-time decision-making. Edge Computing is a response to the limitations of centralized cloud architectures. It empowers devices like IoT sensors with computational capabilities, thereby challenges regarding limited resources and decentralized management across various edge infrastructures. To address some of these challenges, Serverless Computing emerged as a revolutionary cloud computing model that abstracts infrastructure management and focuses on granular function execution. By automatically scaling resources and executing functions in response to events, Serverless Computing offers agility and cost-effectiveness. Nonetheless, challenges persist regarding cold start latency and managing applications. Serverless Edge Computing merges Edge and Serverless paradigms to create a powerful solution. By combining agility, scalability, and event-driven features of Serverless Computing with localized processing advantages of Edge Computing, it has the potential to revolutionize computing. This integration aims to optimize resource utilization, minimize latency, and enhance real-time decision-making by deploying functions closer to the data source at the network edge.reducing bandwidth usage and improving efficiency. An application is divided into microservices.A microservice is a small, self-contained software component that performs a specific task or function within an application. In remote setup,edge nodes are powered from renewable resources.Because renewable energy sources are inherently unstable, customers who depend on steady and dependable power generation face difficulties. We have made an effort to increase the accessibility and dependability of renewable energy sources in light of this constraint. We also recognize the need to reduce the expenses related to the use of renewable energy.

Chapter 2

Literature Survey

The current research in the field of serverless edge computing is focused on optimizing energy consumption and improving system performance. One of the key areas of exploration is the use of stateless functions and the placement of microservices within these functions. This approach offers several benefits such as scalability, flexibility, and ease of deployment, making the architecture more efficient and manageable. Moreover, the use of Docker technology for loading microservices in containers has been widely discussed as it provides a lightweight and efficient way to encapsulate microservices along with their dependencies. This method simplifies the deployment process in containerized environments, contributing to improved system efficiency.

Energy-aware resource scheduling algorithms are designed to optimize the performance of renewable energy- and battery-powered edge nodes while considering energy constraints. They incorporate features like "warm scheduling" and "sticky offloading" to enhance operational availability and performance. By pre-scheduling functions based on energy availability, these algorithms optimize resource utilization and improve overall system performance. Additionally, the concept of queue stabilization is crucial for maintaining a balanced state in the task offloading process, preventing backlog, and ensuring system stability.

In an effort to reduce latency issues in serverless edge computing, various strategies have been introduced. One such strategy is the use of ”warm containers” which are pre-loaded with necessary software to minimize delays caused by ”cold starts”. This enhances system responsiveness, and performance. Another approach is implementing threshold-based queueing solutions. This helps to reduce the waiting time of requests in the queue and hence the overall system latency. Moreover, warm containers are considered an energy-efficient alternative to cold containers. Since warm containers have minimal start-up latency, they require lower CPU speed allocation, contributing to decreased power consumption in the data center.
An optimization strategy has been proposed to minimize server energy consumption while meeting Quality of Service (QoS) requirements for mobile users. This approach integrates cold, warm, and running containers and manages CPU loads and container states efficiently. Its aim is to reduce power consumption while ensuring timely responses to user requests. The optimization is further enhanced by a threshold-based queueing solution that ensures only necessary containers are running, thus reducing energy consumption even more.

The insights gained from the study emphasize the significance of optimizing energy consumption and enhancing system performance in serverless edge computing. The possibility of achieving these goals is promising through the integration of stateless functions, efficient container management strategies, and energy-aware resource scheduling algorithms. These insights can be leveraged in future research to further explore these avenues.

Chapter 3

Problem Definition

In a serverless edge computing architecture, microservices are first generated on edge nodes, which are then subdivided into functions. These edge nodes are powered by renewable energy sources. To ensure Quality of Service (QoS), functions are replicated on a node and can be offloaded to other nodes if necessary. The main objective is to maximize operational availability of the nodes and minimize costs associated with offloading replicas.

Chapter 4

Methodology

4.1 System Model

The network topology consists of a set of edge nodes that establish a wireless connection with the controller. These edge nodes are powered by renewable energy resources. The controller node is responsible for scheduling functions in Edge Nodes. User requests are generated in the Edge nodes themselves. We have divided time into equal time slots to manage the operations efficiently.

Edge Nodes: In an edge computing network, there is a collection of edge nodes represented by D. Each edge node has a state of charge, S, which shows the current battery charge at each time slot. This state is limited by a maximum battery charge, ν. Moreover, each edge node has a resource capacity (c) that is measured in MIPS (Million Instructions Per Second), and W represents the maximum capacity. In this system, energy input (r) may come from various renewable sources. Conversely, energy consumption (e) mainly comes from computational tasks and operational activities that the edge nodes perform.Edge nodes are connected,have a Bandwidth B.

Symbol	**Description**	**Range**
T	Time interval	$\{t \mid t \in [0, T]\}$
i	Index for node	$i \in \{1, 2, \ldots, n\}$
d_i	i^{th} node	-
s_t^i	State of Charge of i^{th} node at time t	$[0, \nu]$
r_t^i	Renewable energy input on i^{th} node at time t	-
e_t^i	Energy consumed by i^{th} node at time t	-
c_t^i	Resource capacity of i^{th} node at time t (MIPS)	-0,W]
B_{i_1,i_2}	Bandwidth between i_1 and i_2, $B_{i_1,i_2} = 0$ if $i_1 \equiv i_2$	-
x_t^i	i^{th} Node availability at time t	$x_t^i \in \{0, 1\}$

Table 4.1: Description of symbols used in the model.

Application Workload: In edge computing, applications are built using microservices, which are individual components that perform specific functions.

Serverless functions: Microservices are divided into stateless functions that are executed in nodes. F represents the set of functions, while h represents the required resource capacity of functions. The resource capacity of a function is capped at the node's maximum capacity W. C represents the required battery charge of functions. The battery charge of a function is capped at the node's maximum battery charge V.

Function placement: When considering where to run functions replicas in edge computing, there are two main approaches to choose from: **local placement (L)** and **foreign placement (Q)**. Local placement means deploying functions directly on the node where it is generated, while foreign placement involves deploying functions on node where it was not generated. The decision between these strategies depends on the delay cost between nodes.

Symbol	Description	Range/Condition
j	Index for microservices	$j \in \{1, 2, \ldots, m\}$
a^j	j^{th} microservice	-
$\lambda_t^{i,j}$	Rate of generating workload for the j^{th} microservice at time t	$\lambda_t^{i,j} \geq 0$
m_t^j	Amount of processing j^{th} microservice requires at time t (MI)	-
l	Index for functions	-
$f_t^{l,j}$	l^{th} function of j^{th} microservice at time t	-
$h^{j,l}$	Resource capacity needed for l^{th} function of j^{th} microservice	-
$\gamma_t^{i,l,j}$	Required replica of l^{th} function of j^{th} microservice at time t	-
$f_t^{i,j,l,k}$	k^{th} replica l^{th} function of j^{th} microservice on i^{th} node at time t	-
$p_t^{i,j,l,k}$	Local placement of k^{th} replica of l^{th} function of j^{th} microservice on i^{th} node at time t	-
$\phi_t^{i,j,l,k}$	Foreign placement of k^{th} replica of l^{th} function of j^{th} microservice on i^{th} node at time t	-
x_t^i	i^{th} Node availability at time t, where $x_t^i \in \{0, 1\}$	-
$o^{k,l,j,h}$	Cost of offloading of k^{th} replica of l^{th} function of j^{th} microservice	$h \in \{send, recv\}$
-		

Table 4.2: Description of symbols used in the system model.

Node availability: Node availability refers to the ability of a specific node in an edge computing network to perform tasks during a particular time slot. It means that the node is operational and can complete the assigned tasks within the given time frame. The nodes can only function if they meet an energy threshold . The availability of a node during a specific time slot can be represented by X, where 1 indicates that the node is available, and 0 indicates that it is not available.

X=Node available(1) or not(0) during a time slot.

Our objective is to maximize operational availability while minimizing costs during offloading.

Terms

Symbol	Description
P_c	Power consumption rate
Z	Total cost calculated after all placements
N	Maximum number of replicas possible per function
β	Minimum operational availability threshold
θ	Low energy threshold

Table 4.3: Description of terms.

4.2 Problem Formulation

There are equation given below which are formulated around our objective function under some constraints:

Availability of Nodes:

$$\forall d^i \in D,\ x_t^i : \begin{cases} 1 & \text{if } s_t^i \geq \theta \\ 0 & \text{else} \end{cases} \tag{4.1}$$

State of Charge:

$$\forall d^i \in D, SoCofd^i : \quad s_t^i = \min(\nu, \max(0, s_{t-1}^i + r_t^i - e_t^i)) \tag{4.2}$$

Energy Calculation:

$$e_t^i = \left(\frac{c_t^i}{\omega} \cdot P_c + \left(\sum_{j=1}^{m} (o^{klsend} \cdot (r_t^i lj - p_t^{ijlk}) + \sum_{j=1}^{m} (o^{kljrecv} \cdot \phi_t^{ijlk}) \right) \right) \cdot \Delta t \tag{4.3}$$

Objective Function:

$$Z = \text{Cost} = \min \left(\sum_{i=1}^{n} \phi_t^{i,l,j,k} \cdot Delay_{i1,i2} \right) \tag{4.4}$$

Subject to:

1. $\min_{i \in [1,n]} \sum_{t=0}^{T} x_t^i > \beta$
2. $\sum_{i=1}^{n} (p_t^{i,j,l,k} + \phi_t^{i,j,l,k}) \leq \sum_{i=1}^{n} (\gamma_t^{ilj}) \forall t \in T, \forall a^j \in A$
3. $c_t^i = \sum_{j=1}^{m} ((p_t^{i,j,l,k} + \phi_t^{i,j,k,l}) \cdot h^{i,l}) \leq (x_t^i \cdot \omega) \quad \forall t \in T, \forall d^i \in D$

Required Number of Replicas per Function:

$$\gamma_t^{(i,j)} = \min \left(N, \frac{\lambda_t^{i,j} \cdot m_t^j}{h^{j,l}} \right) \tag{4.5}$$

NOTE: Delay is given by Bandwidth*Workload of Function Replica being transferred.

4.3 Algorithm

Our system comprises of n edge nodes and m functions, which are generated on the edge nodes themselves. Our algorithm involves calculating the number of replicas each function will require based on the workload. We then map these functions and their replicas in a data structure T. To optimize the placement of these replicas, we sort T in descending order based on the workload of functions. We then place the replicas of each function one by one, searching for an available node to place them in. To minimize delay cost, we identify the node with the minimum delay cost and place the function in that node. We calculate the Quality of Service by determining the number of function replicas that we were unable to place and multiplying it by the penalty set by the user.

Algorithm 1 Replica Placement Algorithm

```
1: Input: $N$ nodes, $M$ functions, $G$ Graph of $N$ nodes connected with each
   other, $V$ (2D vector of function with replicas), $L$ (maximum size of $V[i]$)
2: for $j = 1$ to $M$ do
3:     $r \leftarrow$ GetReplica($f_j$)                ▷ Get replicas for function $f_j$
4:     Add $r$ replicas to $V$ mapped with $f_j$
5:     L=max(L,r)
6: Sort $V$ in decreasing order of Workload
7: Initialize $i = 0$
8: $Count \leftarrow 0$                     ▷ Initialize count for QoS calculation
9: while $i < L$ do
10:    for $j = 0$ to $|V| - 1$ do
11:        if $i < |V[j]|$ then
12:            place =Call Place($V[j][i]$)            ▷ Place replica $V[j][i]$
13:            if place $= 0$ then
14:                $Count$ ++          ▷ Increment count for failed placements
15:            else
16:                Change $V[j][i]$'s current node     ▷ Change current node of
   successfully placed replica
17:        $i$ ++
18: $qos = Count \times penalty$            ▷ Calculate Quality of Service (QoS)
```

Algorithm 2 Place Function Algorithm

```
1: Input: $G$ Graph of $N$ nodes connected with each other, $V$ (2D vector of
   function with replicas), $A$ ( vector of available nodes)
2: function PlaceFunction($f_j$)
3:     A:Call Available(G)                              ▷ to find available nodes
4:     Initialize $Min = INT\_MAX$, $Node = 0$
5:     for each $a$ in $A$ do
6:         $C = getCost(f_j, G, a)$
7:         if $C < Min$ then
8:             $Min = C$
9:             $Node = a$
10:        end if
11:    end for
12:    return $Node$
13: end function
```

Chapter 5

Conclusion and Future work

We have successfully come up with a scheduling algorithm, that considers increasing reliability while minimizing cost during offloading. Additionally, it restricts offloading to promote a warm start.

The Algorithm can be improved by :
- Incorporating the functions to be scheduled as a directed acyclic communication graph instead of a set of functions
- Integration of machine learning techniques to predict workload patterns and optimize scheduling decisions proactively
- The algorithm was optimized based on bandwidth delay but it can be optimized based on more than one parameter using multi-objective optimization techniques

References

[1] M. S. Aslanpour, A. N. Toosi, M. A. Cheema, and R. Gaire, "Energy-aware resource scheduling for serverless edge computing," in *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pp. 190–199, 2022.

[2] M. Lin, J. Xi, W. Bai, and J. Wu, "Ant colony algorithm for multi-objective optimization of container-based microservice scheduling in cloud," *IEEE Access*, vol. 7, pp. 83088–83100, 2019.

[3] M. S. Aslanpour, A. N. Toosi, M. A. Cheema, and M. B. Chhetri, "faashouse: Sustainable serverless edge computing through energy-aware resource scheduling," *IEEE Transactions on Services Computing*, pp. 1–14, 2024.

[4] X. Duan, F. Xu, and Y. Sun, "Research on offloading strategy in edge computing of internet of things," in *2020 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, pp. 206–210, 2020.

[5] M. Adeppady, A. Conte, H. Karl, P. Giaccone, and C. F. Chiasserini, "Energy-aware provisioning of microservices for serverless edge computing," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, pp. 3070–3075, 2023.