Energy-Efficient Function Invocation Scheduling for Sustainable Edge FaaS

Francesca Righetti[1], Carlo Vallati[1], Nicola Tonellotto[1], Gabriele Russo Russo[2], Valeria Cardellini[2], Giuseppe Anastasi[1]
[1]*Dep. of Information Engineering, University of Pisa, Pisa, Italy. E-mail: name.surname@unipi.it*
[2]*DICII, Tor Vergata University of Rome, Rome, Italy. E-mail: surname@ing.uniroma2.it*

***Abstract*—Function-as-a-Service (FaaS) is emerging as a lightweight computing model to deploy event-driven applications in the edge-cloud continuum. However, sustainable deployment of FaaS at the edge requires minimizing energy consumption while maintaining Quality of Service (QoS). This paper presents the *Energy-Efficient Function Invocation Scheduling*, E^2FIS, a scheduling algorithm for edge-based FaaS platforms formulated as a Mixed Integer Linear Program (MILP) problem. E^2FIS optimizes function-to-node assignments by consolidating workload on energy-efficient nodes and powering off idle ones. Real-world experiments show that E^2FIS achieves up to 92% energy savings compared to baseline strategies while preserving QoS.**

***Index Terms*—Function-as-a-Service, Edge-Cloud Continuum, Energy Efficiency, Scheduling.**

I. Introduction

Function-as-a-Service (FaaS) has emerged as an efficient paradigm for deploying applications composed of self-contained functions triggered by events, offering the advantage of simplified deployment without the burden of infrastructure management. With the shift from centralized cloud computing to the edge-cloud continuum, there is a growing need to bring FaaS capabilities at the edge, i.e., closer to data sources and end-users. This transition enables lower latency and greater responsiveness to end-users, but also introduces new challenges. Edge nodes are heterogeneous and functions with different execution deadlines and resource requirements must be scheduled on nodes that can accommodate their requirements. The growing demand for FaaS services and the large-scale deployment of FaaS platforms across the edge-cloud continuum place a strong emphasis on energy efficiency in resource management to reduce operational costs and minimize environmental impact [1]. This requires balancing the QoS compliance [2] with minimizing energy consumption to ensure efficient and sustainable operations.

To achieve this, we proposed E^2FIS: *Energy-Efficient Function Invocation Scheduling*, a framework for optimizing resource management in edge FaaS platforms, with a focus on energy efficiency and QoS compliance, guaranteeing that functions meet their deadlines. Unlike previous approaches, E^2FIS prioritizes workload consolidation, scheduling functions on energy-efficient nodes and deactivating idle ones, achieving system-wide energy savings. Additionally, it explicitly considers node and function heterogeneity, making it well-suited for the edge-cloud continuum. We formulate function scheduling as a Mixed Integer Linear Programming (MILP) problem to minimize energy consumption by consolidating FaaS workload, assigning functions to energy-efficient nodes. Idle nodes are switched off to reduce energy waste.

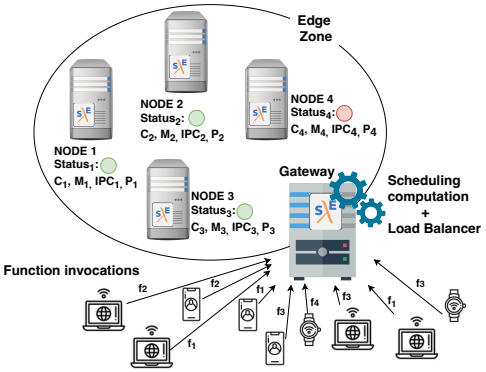

Fig. 1: System architecture

We evaluated E^2FIS through real experiments on the Serverledge FaaS platform [3], comparing its performance against the Earliest Deadline First (EDF) scheduling [4]. Results show that E^2FIS reduces energy consumption up to 92%, ensuring that functions execute within their deadlines.

II. System Architecture and Problem Formulation

The system architecture for E^2FIS is shown in Fig. 1. Function invocations are directed to the closest edge zone, where a central gateway executes E^2FIS. Time is divided into discrete epochs, with the optimization problem at the core of E^2FIS solved before each epoch begins. The solution determines function assignments to worker nodes and, hence, the minimum number of nodes that must be active to handle the FaaS workload. Any idle worker node is powered off to minimize energy consumption.

Each edge node i is characterized by: (i) $status_i$, indicating if the node is active or powered off, (ii) C_i, available computational capacity, i.e., CPU, (iii) M_i, available memory, (iv) IPC_i, instructions per cycle, which is the average number of instructions executed per clock cycle, and (v) P_i, power consumption. The functions registered in the FaaS platform are identified by the set $F = \{f_1, f_2, ..., f_M\}$. Each function f_j is characterized by: (i) m_j, the amount of memory required for its execution, (ii) w_j, the number of instructions it requires to execute, i.e., workload, and (iii) d_j, deadline, the maximum time that elapse between function invocation and completion.

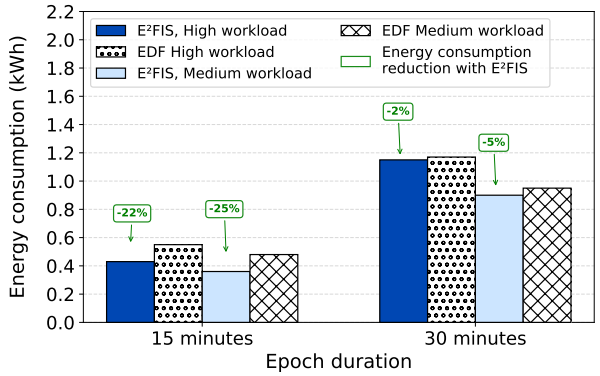

Fig. 2: Total energy consumption. Real experiments

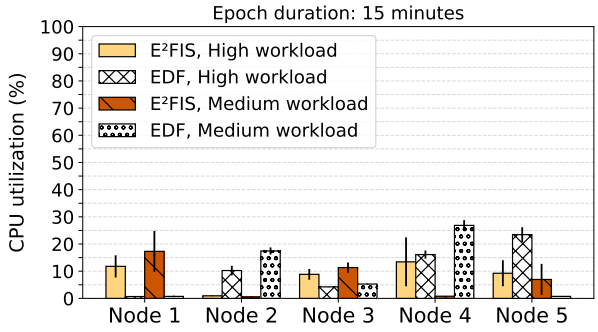

Fig. 3: CPU utilization. Real experiments

To ensure functions meet their deadlines, E^2FIS assumes the worst-case load to compute the function allocation to nodes. Specifically, for each epoch, an estimate of the maximum number of concurrent invocations of a function within a given time slot, denoted as n_j, is given as input. A forecasting model could estimate n_j, but load prediction is beyond the scope of this paper. We assume n_j is known in advance.

The core of E^2FIS is an optimization problem, formulated as a MILP, that determines the optimal functions placement across worker nodes. It prioritizes execution on the most energy-efficient nodes while ensuring functions meet their deadlines, allowing idle nodes to be powered off. To define this optimization problem, we consider an edge zone with N heterogeneous edge nodes, each with different computing capacity, memory, and power consumption. The problem is formulated as follows:

$$\min \quad \sum_{i=0}^{N} y_i E_i \tag{1.1}$$

$$\text{subject to} \quad \sum_{j=0}^{M} n_{ij} m_j \leq M_i y_i, \quad \forall i \tag{1.2}$$

$$\sum_{j=0}^{M} c_{ij} \leq C_i y_i, \quad \forall i \tag{1.3}$$

$$\frac{n_{ij} w_j}{c_{ij} IPC_i} \leq d_j, \quad \forall i, \forall j \tag{1.4}$$

$$\sum_{i=0}^{N} n_{ij} = n_j, \quad \forall j \tag{1.5}$$

The model produces three outputs: (i) the status of each node for the next epoch, represented by binary variable y_i; (ii) the maximum number of concurrent invocations of function f_j assigned to node i (n_{ij}); and (iii) the total CPU allocated to f_j on node i (c_{ij}). The objective function in Eq. 1.1 minimizes the total system energy consumption by determining the set of active nodes needed to execute all functions within their deadlines, where E_i is the energy required to keep node i active during the epoch. Constraint 1.2 ensures that memory required for f_j on node i does not exceed its capacity. Constraint 1.3 limits total CPU assigned to f_j on node i within its available capacity. Constraint 1.4 guarantees that function deadlines are respected. Constraint 1.5 ensures the total number of invocations of f_j matches the maximum n_j. Each node is characterized by its CPU capacity C_i (GHz), memory M_i (GB), and power consumption P_i (W). Energy consumption E_i is modeled as constant over time based on P_i, though other models can be integrated.

III. Experimental Analysis

To evaluate the effectiveness and practicality of E^2FIS, we carried out real-world experiments on the Serverledge FaaS platform, integrating E^2FIS. We considered an edge zone with 7 nodes equipped with amd64-architecture CPUs, epochs of 15 and 30 minutes and two workload types, n_j: *high* and *medium workload*. Fig. 2 shows the energy consumption of E^2FIS and EDF. E^2FIS achieves a significant reduction in energy consumption compared to EDF. This effect is particularly evident for 15 minute epochs, where energy consumption is reduced in a range from 22% to 25%.

From Fig. 3, we can see that E^2FIS predominantly relies on Node 1. This behavior aligns with the core principle of E^2FIS, that makes it the preferred choice for energy-efficient scheduling. Conversely, EDF prioritizes nodes with higher CPU frequencies, particularly Nodes 2, 4, and 5, which offer greater computational capacity but are less energy-efficient.

IV. Conclusions

E^2FIS (Energy-Efficient Function Invocation Scheduling) is a novel framework for optimizing centralized function scheduling in edge FaaS platforms, aimed at reducing energy consumption while meeting function deadlines, prioritizing execution on energy-efficient nodes and powering off idle ones. For future work, we will explore how to manage edge nodes with distributed scheduling to enhance scalability and reduce computation time in large-scale edge deployments.

References

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

[2] M. Shahrad, J. Balkind, and D. Wentzlaff, "Architectural implications of function-as-a-service computing," in *52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, p. 1063–1075.

[3] G. Russo Russo, T. Mannucci, V. Cardellini, and F. Lo Presti, "Serverledge: Decentralized function-as-a-service for the edge-cloud continuum," in *2023 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2023, pp. 131–140.

[4] F. Zhang and A. Burns, "Schedulability analysis for real-time systems with EDF scheduling," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1250–1258, 2009.