Fog computing task scheduling

a multi-objective AI-based approach

Théo FIGINI
M2 Computer Science
Academic year 2023-2024

Hosting organization: *LAAS-CNRS*

February, 1st 2024 - June, 25th 2024

Referring teacher:
Emmanuel BIABIANY

Tutors:
Tom GUÉROUT
Andrei DONCESCU

June, 18th 2024

Contents

Abstract

1 Introduction

As the number of devices grows exponentially, the demand for computing resources increases. The traditional cloud computing model is not sufficient to handle the massive amount of data generated by these devices. Fog computing is an alternative model that extends the computing resources to the edge of the network, closer to the devices. This model reduces the latency and bandwidth usage, and improves the overall performance of the system. However, the increasing demand for computing resources also raises concerns about the energy consumption and the environmental impact of fog computing systems.

In this context, task scheduling plays a crucial role in optimizing the performance of fog computing systems. Task scheduling algorithms aim to allocate the available resources efficiently to the tasks, while still meeting the Quality of Service (QoS) requirements of the users. Several task scheduling algorithms have been proposed in the literature, ranging from simple heuristics to complex optimization techniques. However, most of these algorithms focus on minimizing the energy consumption or maximizing the performance of the system, without considering the environmental impact of the energy sources used to power the system.

In this paper, we will propose an AI-based task scheduling algorithm that integrate the use of green energy sources while satisfying the QoS requirements of the users. The proposed algorithm will consider the availability of green energy sources, mainly solar panels, and will schedule the tasks to maximize the use of green energy while minimizing the energy consumption from the grid.

The rest of this paper is organized as follows. In Chapter 2, we present an overview of the existing research on task scheduling, energy efficiency and AI in the context of fog computing. In Chapter 3, we describe the proposed task scheduling algorithm and the integration of green energy sources. In Chapter 4, we present the experimental results and the performance evaluation of the algorithm. Finally, in Chapter 5, we conclude the paper and discuss the future work.

2 Related Work

There are many existing studies and researches on the topics task scheduling, energy efficiency and AI in the context of fog computing. The concept of fog computing was introduced by Cisco in 2012 [1], and since then, many researches have been conducted to improve the performance of fog computing systems and reduce their environmental impact.

2.1 Fog computing

Fog computing is a paradigm that extends cloud computing and services to the edge of the network. [2] and [3] respectively investigate the concepts surrounding fog and its applications in the context of IoT. [4] offers an extensive review of the existing work on fog computing.

2.1.1 Task Scheduling in Fog Computing

Genetic Algorithms

The task scheduling problem in fog computing has been widely studied in the literature. Many researchers have proposed different algorithms and techniques to optimize the task scheduling process in fog computing systems. [5] offers an analysis of the existing algorithms while identifying the challenges and research gaps. In [6], the authors propose a task scheduling algorithm based on Swarm Intelligence and Machine Learning. This hybrid approach minimizes the execution time and the makespan and improves the performance of the load balancing scheduling. In [7], an optimization model is proposed for the problem of mapping data stream over fog nodes while considering the load of those nodes and the latency between the sensors and the nodes. A heuristic based on genetic algorithms is then presented to address the complexity of the problem. In [8], the authors propose a task clustering and scheduling mechanism based on Differential Evolution to find the optimal execution time for the tasks. The mechanism is compared to the Firefly Algorithm and Particle Swarm Optimization, and the results show that the proposed mechanism outperforms the other two algorithms in terms of execution time, system efficiency and stability.

Fuzzy Logic

[9] proposes a ranking-based task scheduling method that combines fuzzy logic and user preferences. In [10], the authors propose an approach based on Fuzzy Logic for real-time task scheduling in IoT applications.

2.1.2 Energy Efficiency in Fog Computing

The energy efficiency of fog computing systems has been a major concern for researchers. [11] introduces a mathematical framework to evaluate the trade-off of fog computing systems, especially in terms of power consumption and energy efficiency. In [12], the authors propose two Integer Linear Programming models, where the second one aims at minimizing the energy consumption while maximizing successfully provisioned tasks. The authors of [13] provides an overview of the energy efficiency challenges in fog computing and presents a comprehensive survey of the existing energy-efficient techniques and algorithms. An energy-aware Metaheuristic algorithm based on the Harris Hawks Optimization algorithm, itself based on a local search strategy for task scheduling in fog computing, is proposed in [14]. [15] proposes an energy efficient algorithm through an integrated computation model.

2.1.3 AI in Fog Computing

The use of AI in fog computing has been a growing trend in the literature. An efficient binary CNN with numerous skip connections is proposed by the authors of [16]. In [17], the authors designed an intelligent energy-saving model based on CNN and a task scheduling model is designed based on the policy gradient algorithm. [18] present an improved convolutional neural network so solve the value function of a Continuous Markov Decision Process model, so it can be applied to a multi-user system.

3 Offloading using Fuzzy logic

3.1 Introduction

Introduced in 1965 by Lotfi Zadeh[19], fuzzy logic is based on a "degree of truth" instead of a finite value, usually 0 or 1, it aims to represent the vagueness of human language and thought. Fuzzy systems are the means to implement fuzzy logic, they are two types of fuzzy systems: Mamdani and Sugeno, both are similar but differ in the way the output is determined. The most common one Mamdani and it's the one we will be using in this project. The Mamdani fuzzy system follows three steps:

- The inputs are fuzzified into fuzzy membership functions,
- a set of rules are applied to the fuzzy inputs to determine the fuzzy output,
- the fuzzy output is defuzzified to get a crisp value.

3.1.1 Fuzzification

Fuzzification is the process of converting a crisp value into a fuzzy value, this is done by assigning a membership function to the input value. The membership function is a curve that defines how much the input value belongs to a certain fuzzy set. The most common membership functions are the triangular and trapezoidal functions, they are defined by three or four parameters respectively. The triangular function is defined by the parameters a, b and c and is given by:

$$\mu(x) = \begin{cases} 0 & \text{if } x \leq a, \\ \frac{x-a}{b-a} & \text{if } a \leq x \leq b, \\ \frac{c-x}{c-b} & \text{if } b \leq x \leq c, \\ 0 & \text{if } x \geq c. \end{cases} \tag{1}$$

The trapezoidal function is defined by the parameters a, b, c and d and is given by:

$$\mu(x) = \begin{cases} 0 & \text{if } x \leq a, \\ \frac{x-a}{b-a} & \text{if } a \leq x \leq b, \\ 1 & \text{if } b \leq x \leq c, \\ \frac{d-x}{d-c} & \text{if } c \leq x \leq d, \\ 0 & \text{if } x \geq d. \end{cases} \tag{2}$$

3.1.2 Rule evaluation

The rule evaluation is the process of determining the fuzzy output based on the fuzzy inputs and a set of rules. The rules are defined by two parts: the "if" or antecedent part and the "then" or consequent part. The antecedent part deals with inputs, it can either be a single input or a combination of inputs, the combination can be done using the logical operators "and" and "or". The consequent part deals with the output. In the context of this project, the rules are usually of the form "if Bandwidth is low then processing is local". The rules are generally defined by the user and are based on their knowledge of the system.

3.1.3 Aggregation

The aggregation is the process of combining the fuzzy outputs from the rules to get a single fuzzy output. This process relies on T-conorms, and must satisfy the following properties:

- Commutativity: $x * y = y * x$,
- Associativity: $x * (y * z) = (x * y) * z$,
- Monotony: $x \leq y \implies x * z \leq y * z$,
- Neutrality of 0: $x * 0 = x$ for $x \in [0,1]$.

They are also *positive reinforcement* operators:

$$f(x_1, \cdots, x_n) \leq max[x_i] \forall x_i \geq 0.5 \tag{3}$$

As opposed to T-norms which are *negative reinforcement* operators:

$$f(x_1, \cdots, x_n) \geq min[x_i] \forall x_i \leq 0.5 \tag{4}$$

The most common T-conorms are the maximum, the probabilistic sum and the bounded sum. They are defined as follows:

$$\text{Maximum: } x \oplus y = \max(x, y) \tag{5}$$

$$\text{Probabilistic sum: } x \oplus y = x + y - x \cdot y \tag{6}$$

$$\text{Bounded sum: } x \oplus y = \min(x + y, 1) \tag{7}$$

3.1.4 Defuzzification

Defuzzification is the process of converting a fuzzy output into a crisp value, there are several methods to do this, the most common one is the centroid method. The centroid method calculates the center of mass of the fuzzy output, this is done by taking the weighted average of the output values. The weighted average is calculated by taking the sum of the product of the output value and its membership value divided by the sum of the membership values. The formula for the centroid method is given by:

$$y = \frac{\sum_i \mu_i \cdot y_i}{\sum_i \mu_i} \tag{8}$$

Where y is the crisp output, μ_i is the membership value of the output value y_i. The centroid method is the most common method because it is simple and easy to implement. However, it is not always the best method, other methods like the mean of maximum and the largest of maximum can be used depending on the application.

3.2 Fuzzy system for offloading

3.2.1 Setting up the engine

To demonstrate the use of fuzzy logic in offloading, we wrote two simple programs in Python. The goal was to recreate the experiment done by Hari et al.[20]. The first program uses the pyfuzzylite[21] library to implement a fuzzy engine with all the variables and rules needed to determine the offloading decision. The second program uses the NumPy library to generate random values for the inputs and then uses the fuzzy engine to determine the offloading decision. The fuzzy engine is defined by the variables shown in table 1 and 2.

Name	Range	Fuzzy set	Membership function	Parameters
Bandwidth (in Mbps)	[0, 100]	bw_low	trapezoidal	0, 20, 30, 40
		bw_med	trapezoidal	35, 45, 60, 70
		bw_high	trapezoidal	65, 75, 90, 100
Data size (in KB)	[0, 600]	data_low	trapezoidal	0, 0, 230, 360
		data_med	trapezoidal	250, 350, 470, 590
		data_high	trapezoidal	450, 540, 600, 600
Residual battery charge (in %)	[0, 100]	bat_low	trapezoidal	0, 0, 25, 35
		bat_med	trapezoidal	25, 40, 60, 75
		bat_high	trapezoidal	60, 75, 100, 100
Load (in %)	[0, 100]	load_low	trapezoidal	0, 0, 25, 40
		load_med	trapezoidal	35, 45, 60, 70
		load_high	trapezoidal	65, 80, 100, 100
Memory (in %)	[0, 100]	mem_low	trapezoidal	0, 0, 25, 40
		mem_med	trapezoidal	35, 45, 60, 70
		mem_high	trapezoidal	65, 80, 100, 100
Virtual machines available	[0, 50]	vm_low	trapezoidal	0, 0, 15, 20
		vm_med	trapezoidal	15, 22, 37, 40
		vm_high	trapezoidal	30, 35, 50, 50
Number of concurrent users	[0, 100]	user_low	trapezoidal	0, 0, 25, 40
		user_med	trapezoidal	30, 40, 60, 70
		user_high	trapezoidal	60, 75, 100, 100

Table 1: Input variables for the fuzzy engine.

Name	Range	Fuzzy set	Membership function	Parameters
Offloading decision	[0, 100]	local	trapezoidal	0, 12, 24, 48
		remote	trapezoidal	36, 60, 72, 100

Table 2: Output variable for the fuzzy engine.

The original paper by Hari et al. did not provide all the rules used, so we had to come up with our own rules. We established them based on our understanding of the system and the variables. Unlike the original paper, where the rules only combined the bandwidth with one other variable, we decided to combine the bandwidth with all the relevant variables. The rules are shown in table 3.

	Rules
R1	IF Bandwidth is bw_low THEN Processing local_processing
R2	IF Bandwidth is not bw_low and Datasize is not data_low THEN Processing is remote_processing
R3	IF Bandwidth is not bw_low and Datasize is data_low and (Load is load_high or Memory is mem_low) THEN Processing is remote_processing
R4	IF Bandwidth is not bw_low and Datasize is data_low and (NB_concurrent_users is user_low or Memory is mem_low) THEN Processing is remote_processing
R5	IF Bandwidth is not bw_low and Datasize is data_low and NB_concurrent_users is not user_low and Memory is not mem_low and Load is load_low THEN Processing is local_processing
R6	IF Bandwidth is not bw_low and Datasize is data_low and NB_concurrent_users is not user_low and Memory is not mem_low and Load is not load_low THEN Processing is remote_processing

Table 3: Rules for the fuzzy engine.

From these rules we can see that two variables have no impact on the offloading decision, the residual battery charge and the number of virtual machines available. So we decided to remove them from the fuzzy engine.

3.2.2 Mean 3Π aggregation operator (M3Π)

The standard 3Π operator was introduced by Yager and Rybalov [22] in 1988, it is a generalization of the probabilistic sum. The 3Π operator is defined by the following formula:

$$\Pi(x_1, \cdots, x_n) = \frac{\Pi_{j=1}^{n} x_j}{\Pi_{j=1}^{n} x_j + \Pi_{j=1}^{n} (1 - x_j)} \tag{9}$$

Unlike T-norms and T-conorms, the 3Π operator is *full reinforced* which means that it satisfies the properties of both positive and negative reinforcement operators.

We decided to implement a M3Π aggregation operator which is derived from the 3Π operator, it is defined by the following formula:

$$M3\Pi(x_1, \cdots, x_n) = \frac{\Pi_{j=1}^{n} (x_j)^{(1/n)}}{\Pi_{j=1}^{n} (x_j)^{(1/n)} + \Pi_{j=1}^{n} (1 - x_j)^{(1/n)}} \tag{10}$$

The goal was to improve the decision-making process. However, we were unable to implement the M3Π operator as it would require to modify a large part of the *pyfuzzylite* library. We decided to shift our focus towards other parts of the project.

3.2.3 Partial results

Despite the lack of the M3Π operator, we still ran the fuzzy engine with three different aggregation operators: maximum, probabilistic sum and bounded sum. Samples of the results are shown in tables 4, 5 and 6.

Bandwidth	Data size	Number of concurrent users	Memory	Load	Processing	Fuzzy output
30	180	42	38	66	21.575	1.000/local_processing + 0.000/remote_processing
77	462	44	96	8	67.539	0.000/local_processing + 1.000/remote_processing
3	21	45	75	50	60.732	0.150/local_processing + 0.850/remote_processing
38	229	51	31	66	57.565	0.200/local_processing + 0.600/remote_processing
50	302	17	96	92	68.116	0.000/local_processing + 0.554/remote_processing

Table 4: Sample results from the fuzzy engine with the Maximum aggregation operator.

Bandwidth	Data size	Number of concurrent users	Memory	Load	Processing	Fuzzy output
30	180	42	38	66	21.575	1.000/local_processing + 0.000/remote_processing
77	462	44	96	8	67.539	0.000/local_processing + 1.000/remote_processing
3	21	45	75	50	60.443	0.150/local_processing + 0.850/remote_processing
38	229	51	31	66	61.27	0.200/local_processing + 0.904/remote_processing
50	302	17	96	92	68.314	0.000/local_processing + 0.863/remote_processing

Table 5: Sample results from the fuzzy engine with the Probabilistic sum aggregation operator (note: the library calls it Algebraic sum).

Bandwidth	Data size	Number of concurrent users	Memory	Load	Processing	Fuzzy output
30	180	42	38	66	21.575	1.000/local_processing + 0.000/remote_processing
77	462	44	96	8	67.539	0.000/local_processing + 1.000/remote_processing
3	21	45	75	50	60.275	0.150/local_processing + 0.850/remote_processing
38	229	51	31	66	61.855	0.200/local_processing + 1.000/remote_processing
50	302	17	96	92	67.903	0.000/local_processing + 1.000/remote_processing

Table 6: Sample results from the fuzzy engine with the Bounded sum aggregation operator.

3.3 AI-based task scheduler

4 Conclusion

Conclusion

Bibliography

[1] Flavio Bonomi et al. "Fog computing and its role in the internet of things". In: *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*. MCC '12. Helsinki, Finland: Association for Computing Machinery, 2012, pp. 13–16. ISBN: 9781450315197. DOI: 10.1145/2342509.2342513. URL: https://doi.org/10.1145/2342509.2342513.

[2] Muhammad Ehsan Rana and Nirase Abubacker. "Fog Computing Foundations". In: June 2023, pp. 1–20. ISBN: 9781668444665. DOI: 10.4018/978-1-6684-4466-5.ch001.

[3] Nirase Abubacker, Muhammad Ehsan Rana, and Mafas Raheem. "Fog Computing Applications". In: June 2023, pp. 30–58. ISBN: 9781668444665. DOI: 10.4018/978-1-6684-4466-5.ch003.

[4] Mohammed Al-Musawi, Dunya Alrseetmiwe, and Zahraa Saad. "Fog computing system for internet of things: Survey". In: 16 (Nov. 2023), 1_10.

[5] Javid Misirli and Emiliano Casalicchio. "An Analysis of Methods and Metrics for Task Scheduling in Fog Computing". In: *Future Internet* 16.1 (2024). ISSN: 1999-5903. DOI: 10.3390/fi16010016. URL: https://www.mdpi.com/1999-5903/16/1/16.

[6] Gaith Rjoub and Jamal Bentahar. "Cloud Task Scheduling Based on Swarm Intelligence and Machine Learning". In: *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*. 2017, pp. 272–279. DOI: 10.1109/FiCloud.2017.52.

[7] Claudia Canali and Riccardo Lancellotti. "GASP: Genetic Algorithms for Service Placement in Fog Computing Systems". In: *Algorithms* 12.10 (2019). ISSN: 1999-4893. DOI: 10.3390/a12100201. URL: https://www.mdpi.com/1999-4893/12/10/201.

[8] Adil Yousif, Mohammed Bakri Bashir, and Awad Ali. "An Evolutionary Algorithm for Task Clustering and Scheduling in IoT Edge Computing". In: *Mathematics* 12.2 (2024). ISSN: 2227-7390. DOI: 10.3390/math12020281. URL: https://www.mdpi.com/2227-7390/12/2/281.

[9] Mohammed Anis Benblidia et al. "Ranking Fog nodes for Tasks Scheduling in Fog-Cloud Environments: A Fuzzy Logic Approach". In: *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. 2019, pp. 1451–1457. DOI: 10.1109/IWCMC.2019.8766437.

[10] Hala S. Ali et al. "Real-Time Task Scheduling in Fog-Cloud Computing Framework for IoT Applications: A Fuzzy Logic based Approach". In: *2021 International Conference on COMmunication Systems & NETworkS (COMSNETS)*. 2021, pp. 556–564. DOI: 10.1109/COMSNETS51098.2021.9352931.

[11] Raad S. Alhumaima. "Energy efficiency and latency analysis of fog networks". In: *China Communications* 17.4 (2020), pp. 66–77. DOI: 10.23919/JCC.2020.04.007.

[12] Zhiming He et al. "Green Fog Planning for Optimal Internet-of-Thing Task Scheduling". In: *IEEE Access* 8 (2020), pp. 1224–1234. DOI: 10.1109/ACCESS.2019.2961952.

[13] Usman Mahmood Malik et al. "Energy-Efficient Fog Computing for 6G-Enabled Massive IoT: Recent Trends and Future Opportunities". In: *IEEE Internet of Things Journal* 9.16 (2022), pp. 14572–14594. DOI: 10.1109/JIOT.2021.3068056.

[14] Mohamed Abdel-Basset et al. "Energy-Aware Metaheuristic Algorithm for Industrial-Internet-of-Things Task Scheduling Problems in Fog Computing Applications". In: *IEEE Internet of Things Journal* 8.16 (2021), pp. 12638–12649. DOI: 10.1109/JIOT.2020.3012617.

[15] Yan Wang et al. "Service delay and optimization of the energy efficiency of a system in fog-enabled smart cities". In: *Alexandria Engineering Journal* 84 (2023), pp. 112–125. ISSN: 1110-0168. DOI: https://doi.org/10.1016/j.aej.2023.10.034. URL: https://www.sciencedirect.com/science/article/pii/S1110016823009365.

[16] Lijun Wu et al. "An Efficient Binary Convolutional Neural Network With Numerous Skip Connections for Fog Computing". In: *IEEE Internet of Things Journal* 8.14 (2021), pp. 11357–11367. DOI: 10.1109/JIOT.2021.3052105.

[17] Dexian Yang et al. "Energy saving strategy of cloud data computing based on convolutional neural network and policy gradient algorithm". In: *PLOS ONE* 17.12 (Dec. 2022), pp. 1–18. DOI: 10.1371/journal.pone.0279649. URL: https://doi.org/10.1371/journal.pone.0279649.

[18] Bing Jing and Huimin Xue. "IoT Fog Computing Optimization Method Based on Improved Convolutional Neural Network". In: *IEEE Access* 12 (2024), pp. 2398–2408. DOI: 10.1109/ACCESS.2023.3348133.

[19] L.A. Zadeh. "Fuzzy sets". In: *Information and Control* 8.3 (1965), pp. 338–353. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(65)90241-X. URL: https://www.sciencedirect.com/science/article/pii/S001999586590241X.

[20] Dhanya Hari et al. "Fuzzy-logic-based decision engine for offloading IoT application using fog computing". In: May 2018, pp. 175–194. DOI: 10.4018/978-1-5225-5972-6.ch009.

[21] Juan Rada-Vilela. *The FuzzyLite Libraries for Fuzzy Logic Control*. 2018. URL: https://fuzzylite.com.

[22] R.R. Yager and A. Rybalov. "Full reinforcement operators in aggregation techniques". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 28.6 (1998), pp. 757–769. DOI: 10.1109/3477.735386.