

Edge Computing Tasks Orchestration: An Energy-Aware Approach

Johan Løhde Thomsen,
Kristian Dragsbæk Schmidt Thomsen,
Rasmus B. Schmidt, Søren D. Jakobsgaard,
Thor Beregaard and Michele Albano
Aalborg University, Denmark

Sergio Moreschini
*Tampere University
Tampere, Finland*

Davide Taibi
*University of Oulu Oulu, Finland
Tampere University, Tampere, Finland*

Abstract—In this paper, we investigate experimentally the use of auctioning as a method for optimizing task orchestration in distributed computing systems by making selfish agents compete to execute computational tasks. Our goal is to find an approach that can improve the performance of these systems, using a deadline, fines, and reward limits in a reverse second-price sealed bid auction, to incentive and control the system, specifically in terms of improving task throughput and power consumption. With improvements to both energy consumption and task throughput, we have developed a promising approach, that is able to scale with the number of machines in the system. Results suggest that this type of auction may be useful for improving the implementation of these systems in a wide range of scenarios.

Index Terms—auction, deadline, game theory, task offloading, experiments

I. INTRODUCTION

The edge to cloud continuum [1] and cognitive cloud [2] introduce different challenges such as dynamic allocation of the computation [3], the approach robustness against edge nodes' faults [4], and the monitoring of the behavior of the system at runtime [5]. From a more pragmatic point of view, a client interested in executing a computational task might require a certain amount of computational power, and the task to be completed under a given deadline. Edge nodes, especially when owned by different parties, usually try to maximize their own benefits, for example by requiring payment in exchange for the execution of a computational task. The latest trend is to ensure fairness of computation resources by modeling the system following game theory.

The goal of this paper is to propose an orchestration method based on mechanism design, a branch of game theory that studies how to set up competitive games between selfish and rational players, such that a given global goal, expressed through a fitness function, is optimized. Differently from previous works, we do not focus on multiplayer based game theory but we implement a system based on a non-cooperative game theory: an auction.

The contribution of this paper is threefold:

- 1) We implement a game theoretical auction system to distribute the tasks to the edge nodes in order to achieve a fair balance of cost and reward. This involves a reverse Second Price Sealed Bid Auction, which pushes the

nodes to be truthful. Moreover, our auction mechanism allows for maximum costs, deadlines, and fines.

- 2) We make our implementation using a scalable and robust architecture with a focus on the quality of service achievable from the system, ensuring that the system could be useful in a real-world setting.
- 3) We perform experiments making the system use different strategies and rules to find how different parameters and environments affect the system and what system settings perform the best in different environments. All of this is done in a physical environment with multiple independent but networked machines, rather than in a simulation.

II. ORCHESTRATION USING AUCTION MECHANISMS

Offloading workload to edge devices is helpful when cloud communication latency is too high compared to the computational resources' speed. Fairness is vital when assigning workload, and efficiency is necessary to avoid issues from a long distribution time or excessive network traffic. Game theory is popular for modeling the system, given that task orchestration and offloading are multiplayer decision problems where entities may compete selfishly or inefficiently to maximize their benefit. [6]

Auctions are non-cooperative games that have been studied to support many aspects of resource management and pricing for edge and cloud computing [7], comprising task orchestration and offloading [8]. A mechanism based on auctions usually forces the bidders to be strategic [9], i.e. to compute the bid based on its own evaluation of the item to be bid on, and on other parameters that can be hard to assess. Thus, most potential bidders would refrain from participating in auctions that are prone to be strategic.

A particular kind of auction, called Second-price sealed bid auction, or reverse Vickrey auction, can be of particular interest for our solution. Within this framework, the winner of the auction is the bidder making the lowest bid, and it gets paid an amount corresponding to the second-lowest bid. The bid is sealed and therefore kept secret, not to provide confidential information (the bid) to the bidder's competitors. A very useful property of the second price sealed bid auction is that it is incentive compatible, meaning that each edge node will gain

the most from simply being truthful about its valuation, or in this case the costs it would have to endure, knowing that it will be paid at least that value, and possibly make a profit out of the process. This simplifies the bidding, and thus the auction process itself, and can attract more potential bidders to participate in the auction [10]. Moreover, this approach can be deemed a green orchestration method, since it matches each task with the edge device having the lowest operational costs to execute it, which is related to the energy consumed to execute the task.

III. REFERENCE ARCHITECTURE

The *clients* can be a number of different entities lacking the capacity to perform computational-intensive tasks. Tasks submitted could be computationally or data intensive, requiring powerful hardware to handle them in a reasonable time. We focus on computationally intensive tasks and leave the data-intensive ones to more network-focused studies.

The *API-Gateway*, is the entry point of the edge system. It exposes different APIs to the clients, that call them to execute specific tasks.

The *Orchestrator* is a centralized decision maker that receives the tasks sent out for execution. The orchestrator would be placed as close to the edge nodes as possible. Moreover, it decides if the edge nodes have the capacity to solve the tasks, otherwise, it sends them to a cloud service.

The *Edge nodes* are individual selfish edge nodes, with different hardware resources, owned by third parties, supplying the network with their computational resources, in exchange for payment.

IV. THE PROPOSED ORCHESTRATION METHODS

The goal of the orchestration algorithm is to allocate computational tasks over competing and selfish edge nodes, so as to identify the "best" possible task allocation. We define "best" as a combination of three Key Performance Indicators (KPI): Energy Consumption; Task Throughput; Time over Deadline. The algorithm is based on 7 steps:

- 1) **Tasks definition:** the clients provide (asynchronously) computational tasks to bid on and their expected complexities to the Frontend, which forwards them to the orchestrator; optionally, the clients can include a deadline and a fine to be paid if the deadline is missed, a maximum fee that the client agrees to pay for the

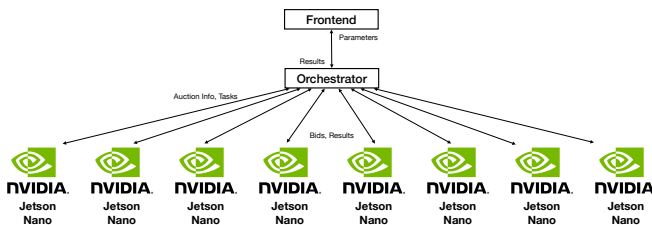


Fig. 1. Overall system setup

execution of the task, a minimum percentage of green energy to be used by the edge node.

- 2) **Tasks provisioning:** The orchestrator sends to the edge nodes one or more tasks to bid on and informs them of the parameters of the auction.
- 3) **Bidding:** The edge nodes can decide to send encrypted messages to the orchestrators containing a bid.
- 4) **Auction:** the orchestrator executes the auction processes to associate to each task one (or zero) edge nodes.
- 5) **Task assignment:** the orchestrator monitors the execution of the tasks by interacting periodically with the assigned edge nodes.
- 6) **Result computation:** the edge nodes send back to the orchestrator the result of the task.
- 7) **Results Availability:** the orchestrator makes the results available to the clients, computes the incentives for the edge nodes, and provides them to it.

We also investigate the additions of deadlines with corresponding fines for tasks, and having a maximum possible reward for each task. In the first case, the bids of the machines change depending on its own estimate of the task duration, and the cost of the fine. If the edge node estimates¹ it will take longer than the deadlines of the task to complete, its bid will include the cost of the fine. The bid in this case is:

$$\text{bid} = \begin{cases} p_c \cdot d + \text{fine} & \text{if } d > \text{deadline} \\ p_c \cdot d, & \text{otherwise} \end{cases}$$

where p_c is the power consumption and d is task duration. In the case where we instead have a maximum value, to limit unrealistic bids and stop workers from gaining large rewards from tasks that other workers find difficult, a machine would not bid at all if its computed bid is larger than the limit. These two cases can be combined so that the fines are counted in the total reward if the duration is higher than the deadline.

V. EXPERIMENTS AND RESULTS

The system needs to be able to accommodate different experimentation environments and quickly exchange strategies on the machines to ease the experimentation.

To represent this scenario, a setup is created using eight Nvidia Jetson Nanos² (the edge nodes), and a desktop computer equipped with an Intel Xeon W-1250P - 4.1GHz (the distributor). We limited the clock speed of the Jetson Nanos, to simulate machines with different power. The distributor gets orchestration parameters from the clients by means of a REST API. Then it sets up actions using these parameters and communicates with the edge nodes using websockets. Each machine is running a Python program capable of receiving the parameters for the orchestration experiments, including: **Amount of Tasks** sent into the system; **Range of dimensions of the matrices**, which then are generated randomly; **Frequency of tasks** that are sent to the distributor, which has

¹Estimate is either based on previous experience on the tasks or on the task complexity sent out by the client in Step 1 of the algorithm

²<https://developer.nvidia.com/embedded/jetson-nano-developer-kit>

Experiment #	1	2	3	4	5	6
Tasks	25	25	25	25	25	25
Deadline	True	True	True	False	False	False
Freq	Mid	Mid	Mid	Low	Mid	High
Fines	True	True	False	False	False	False
Max Reward	True	False	True	False	False	False
Energy (kWh)	0.007	0.010	0.011	0.012	0.007	0.011
Time Taken	5:50	6:20	7:00	7:30	4:35	7:30
%late tasks	68%	76%	65%	N/A	N/A	N/A
Avg task delay	57s	66s	63s	N/A	N/A	N/A
Max task delay	191s	223s	212s	N/A	N/A	N/A

TABLE I

THE MEAN OF RESULTS OVER THREE RUNS PER EXPERIMENT, ALL WITH 25 TASKS, AND DEADLINES BETWEEN 50 AND 70 SECONDS.

three options: 1 task/s, 5 tasks/s, and 10 tasks/s, later referred to as low, medium, and high frequency respectively; **Maximum reward**, either true or false, the reward is then defined automatically based on the size of the deadline, whether used or not, and the size of the matrix; **Deadlines**, being true or false, and if so the deadline will be a random number between a compilation-defined maximum and minimum value, and the fine being double the calculated max reward field.

A number of clients are simulated by a client process run on the desktop machine that generates tasks in accordance with parameters given. In our experiments, for the sake of simplicity, we chose to do let the tasks be simple matrix multiplication as this is a computation intensive task with a naive solution having a time complexity of $O(n^3)$.

We performed 6 different experiments. In each experiment, the number of tasks assigned was fixed to 25, while the rest of the parameters could be tweaked. Such parameters are deadline, frequency, fines, and max rewards.

From the analysis of such experiments we retrieved the values for the energy consumption (in KWh), the amount of time necessary to perform the tasks, the percentage of tasks which arrived late, the average delay for task completion and the maximum delay for task completion.

As per Table I, experiment 1 with all parameters enabled, appears to provide the best overall performance. It has the lowest energy consumption, the highest task throughput, and the smallest amount of delay. Experiment 2, where only deadlines and fines are enabled, shows similar throughput, but worse energy consumption and delay. This would seem to indicate that it is the max reward parameter that impacts the delay, which can be confirmed by experiment 3, showing low task delay, but very slow throughput.

The energy consumption scales with the time taken, since the operating system is still running in the background keeping the system outside a sleep state. Interestingly, experiment 5 and experiment 1 both show the same energy consumption even though there has lapsed 1 minute and 15 seconds in between. Therefore, relinquishing control of how the single tasks are executed (single tasks have long delays, maximum reward can grow dramatically) leads to a faster overall execution while incurring no energy penalty.

The last three experiments evaluated how task frequency affected the system. Low frequency is slow, as expected, with not enough tasks for all machines to work concurrently,

resulting in longer task times and higher energy consumption. The medium frequency experiment was much faster. The high-frequency experiment was surprisingly slow, caused by the task queue filling up too quickly. The experiments indicate that a medium task frequency, with no parameters enabled, yields the best completion time for the 25 tasks.

A medium task-frequency, with multiple machines running simultaneously and all parameters enabled, yielded the best results for the system across each KPI.

VI. CONCLUSION AND FUTURE WORK

In this work, we proposed an edge orchestration approach based on a second-price sealed bid auction. The auction proposed in this paper, with deadlines, fines, and a maximum reward limit, is a viable way to provide a faster total computation, at lower energy costs, and with lower total delay of tasks. By including deadlines, fines, and max reward, the approach provides an increase in both throughput and energy-saving.

In future work, we plan to use our system in a real-world setting, where tasks can be different methods of processing data, calculations of probability, and implement it into a serverless@edge platform [11].

ACKNOWLEDGMENTS

This work was partially supported by the Validation WP of the ERC Advanced Grant LASSO (Learning, Analysis, Synthesis and Optimization of Cyber-Physical Systems), by Industry X and 6GSoft projects funded by Business Finland.

REFERENCES

- [1] S. Moreschini, F. Pecorelli, X. Li, S. Naz, D. Hästbacka, and D. Taibi, "Cloud continuum: The definition," *IEEE Access*, vol. 10, pp. 131 876–131 886, 2022.
- [2] S. Moreschini, F. Pecorelli, X. Li, S. Naz, M. Albano, D. Hästbacka, and D. Taibi, "Cognitive cloud: The definition," in *Int. Conf. on Distributed Computing and Artificial Intelligence*, 2023.
- [3] N. Mehran, D. Kimovski, and R. Prodan, "A two-sided matching model for data stream processing in the cloud – fog continuum," in *Int. Symposium on Cluster, Cloud and Internet Computing*, 2021.
- [4] A. Droob, D. Morratz, F. L. Jakobsen, J. Carstensen, M. Mathiesen, R. Bohnstedt, S. Moreschini, D. Taibi, and M. Albano, "Fault tolerant horizontal computation offloading," in *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*. IEEE, 2023.
- [5] A. S. Klitgaard, A. Alexander Sønderby, H. S. Jørgensen, K. Walstrøm Petersen, J. Dongo, and M. Albano, "Resilience-focused monitoring framework for edge systems," in *2022 IEEE International Conference on Edge Computing and Communications (EDGE)*, 2022.
- [6] S. Noreen and N. Saxena, "A review on game-theoretic incentive mechanisms for mobile data offloading in heterogeneous networks," *IETE Technical Review*, vol. 34, no. sup1, pp. 15–26, 2017.
- [7] N. Sharghivand, F. Derakhshan, and N. Siasi, "A comprehensive survey on auction mechanism design for cloud/edge resource management and pricing," *IEEE Access*, vol. 9, pp. 126 502–126 529, 2021.
- [8] S. Smolka, L. Wißenberg, and Z. Á. Mann, "Edgedecap: An auction-based decentralized algorithm for optimizing application placement in edge computing," *Journal of Parallel and Distributed Computing*, 2023.
- [9] R. Wilson, "Strategic analysis of auctions," *Handbook of game theory with economic applications*, vol. 1, pp. 227–279, 1992.
- [10] R. Besharati, M. H. Rezvani, and M. M. G. Sadeghi, "An incentive-compatible offloading mechanism in fog-cloud environments using second-price sealed-bid auction," *J. Grid Comput.*, vol. 19, no. 3, p. 37, 2021.
- [11] N. El Ioini, D. Hästbacka, C. Pahl, and D. Taibi, "Platforms for serverless at the edge: A review," in *Advances in Service-Oriented and Cloud Computing*, 2021.