A Mathematical Model for Latency Constrained Self-Organizing Application Placement in the Edge

Matteo Mordacchini
Institute for Informatics and Telematics, IIT-CNR
Pisa, Italy
matteo.mordacchini@iit.cnr.it

Emanuele Carlini
Institute for Information Science and Technologies, ISTI-CNR
Pisa, Italy
emanuele.carlini@isti.cnr.it

Patrizio Dazzi
University of Pisa
Pisa, Italy
patrizio.dazzi@unipi.it

ABSTRACT

The highly dynamic and heterogeneous environment that characterizes the edge of the Cloud/Edge Continuum calls for new intelligent methods for tackling the needs of such a complex scenario. In particular, adaptive and self-organizing decentralized solutions have been advanced for optimizing the placement of applications at the Edge. In this paper, we propose a probabilistic mathematical model that allows to describe one of such solutions. The goal of the model is twofold: i) to make it possible to demonstrate the convergence of the proposed solution; ii) to study the impact of the self-organizing solution without the need of an actual implementation or simulation of the system, allowing to evaluate the suitability of the solution in specific contexts. The paper presents the mathematical formulation of the proposed solution as well as the validation of the proposed model against a simulation of the system.

CCS CONCEPTS

• **General and reference** → Validation; *Empirical studies*; *Experimentation*; • **Mathematics of computing** → **Probabilistic algorithms**.

KEYWORDS

Cloud/Edge Continuum, Edge Intelligence, Mathematical Model, Self-organizing, Adaptive, Application Placement

ACM Reference Format:
Matteo Mordacchini, Emanuele Carlini, and Patrizio Dazzi. 2022. A Mathematical Model for Latency Constrained Self-Organizing Application Placement in the Edge. In *Proceedings of the 2nd Workshop on Flexible Resource and Application Management on the Edge (FRAME '22), July 1, 2022, Minneapolis, MN, USA*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3526059.3533620

1 INTRODUCTION

Clouds have represented a very disruptive innovation, allowing to exploit resources on a pay-per-use fashion, actually realising the utility computing paradigm. Nowadays, computing and storage services are shifting from clouds to network edges. Through edge servers, highly distributed on a wide geographical area, as a way to be as much close as possible to end-users, edge computing is endowed with high expectations to meet the demands of latency-sensitive applications [1, 11, 13]. Edge computing environments are becoming increasingly critical in current 5G and even the future 6G networks [20]. In order to be able to provide the requested services, edge resources need to host applications and data. Data and applications can be either provided by edge devices or by a remote cloud. As a matter of fact, such a scenario clearly promotes a direct provisioning of resources from edge servers, in an attempt to avoid congestion in the core network linking edge with remote cloud [18]. In a direct provisioning system, there is a need to allow edge devices to connect one each others, interacting and accessing and retrieve data and applications from each other [5]. In fact, in a Cloud/Edge architecture allowing direct edge interactions, once a user sends a request, such a request has to be managed by its nearest edge server. Then, in an attempt to satisfy the user request, the server either host the associated application locally, on the remote cloud or in other Edge servers. Due to the limitation of resources characterising edge devices, a single server can only host a minimal subset of all the applications that can run into the system. As a result, data and applications needed to satisfy user requests are not necessarily hosted in the resource that is closer to the end user, significantly increasing network load and request latency. When the latency increases, it also increases the risk that the quality of experience of end-users is hindered.

A properly designed application placement schema can significantly reduce, on average, the latency occurring when satisfying user requests. To this end, in the literature, there are many solutions for the actual placement of data repositories [4]. However, many of such solutions assume that both resources and end-users are statically distributed [2, 3, 7, 12, 14, 16, 19], and tend to be quite stable in size. Unfortunately, such kind of scenario does not fit with most of those applications that nowadays represent a relevant share of those having key advantages from the exploitation of Edge resources. In fact, Edge computing provide key benefits when resources at the edge of the network host "nextgen" applications, characterised by strong requirements in terms of latency and bandwidth [9]. Many of these applications are typically accessed by users from mobile and/or handheld devices. Thus, users can easily access applications from different locations [15], determining trends of footprints on resources that vary depending on user movements. Such dynamics in user behavior, and the consequent impact on resources, call for application placement solutions that could confront with the problem by adopting approaches targeting highly dynamic environments [8, 17]. From this regard it is fundamental to find the right

FRAME '22, July 1, 2022, Minneapolis, MN, USA

ACM ISBN 978-1-4503-9310-2/22/07…$15.00
https://doi.org/10.1145/3526059.3533620

Table 1: Table of Symbols.

Symbol	Meaning
$A = \{A_1, \ldots, A_n\}$	Set of all the applications in the system
$\varrho_i(t)$	Funct. returning the numb. of instances of app. i at time t
$\mu_i(t)$	Prob. to turn off an instance of app. i at time t
$p_i(t)$	Prob. to hold an active instance of app. i at time t
$\widetilde{W}$	Maximum weight sustainable by a generic node
$W(t)$	Weight of all the applications on a node at time t
$w_i(t)$	Weight of holding an instance of app. i at time t
w_i^{fix}	Fixed cost of running an instance of app. i
w_i^u	Cost per user of an instance of app. i
U_i	Number of users of app. i
$u_i(t)$	Users-per-instance of app. i at time t
$\widetilde{U}_i^L$	Expected max. numb. of users of A_i w.r.t. latency

trade-off between an approach that will focus only on matching end-user needs (e.g., QoE, latency requirements, etc.) and service providers' needs (limit resource usage to save money). An approach that would be focused only on controlling the resource usage could generate an application allocation schema that strongly limits the resource usage at the edge by deploying resources (almost) only in the Cloud (typically cheaper than Edges), with a potential impairment of the end-users QoE. Conversely, a solution that will prioritize QoE improvement could maximise the involvement of edge resources, implying very expensive deployments.

In a previous work [6], we presented a method that allows Edge resources to autonomously coordinate in order to optimally place the instances of the services running in the system. Specifically, using this scheme, the Edges attempt to limit the instances and at the same time guarantee that the users of the services will be served with a satisfactory QoE. This operation is done by taking into account the constraints posed by the latency requirements of both the users and the applications. The goal is to optimize the usage of the system resources, reducing the consumption needed for serving the actual users, and creating more space for potentially hosting new users and services. In this paper, we provide a mathematical model that describes this solutions.

The rest of this paper is organized as follows. In Section 2 we present the mathematical model of the proposed solution. Section 3 proposes a validation of the model against a simulation of the system. Finally, Section 4 concludes the paper.

2 MODEL DESCRIPTION

In this section, we present the mathematical model of our previously proposed solution. In the following, we use the symbols reported in Table 1. Following the description of the original algorithm, the model assumes that the system is serving a set of applications $A = \{A_1, \ldots, A_m\}$. The applications are run by nodes of the system known as *Edge Miniclouds* (EMCs). Each EMC is an entity that supervises smaller and simpler devices at the Edge. We assume to have n of such EMCs. The model is based on the *average* behaviour over time of a *generic* node. We assume that at each time interval t a node (the *initiator* of the exchange) interacts with another *generic* node (called *receiver*), trying to exchange the users of an application. An exchange implies that all the users that were served by an instance on the initiator are referred to use an instance on the receiver. The initiator can thus shut down its instance, contributing to limit the total number of active instances. Both the initiator and the receiver are based on a average behaviour and, thus, they both have the same characteristics. The model presented in this section allows to compute, for each application $A_i \in A$, the variation over time of the number of its instances running on the EMCs of the system. This quantity is computed, for an application A_i at a time t, using the function $\varrho_i(t)$.

To derive how these values are returned by ϱ_i, we need to determine the probability for an application A_i to see the number of its instances reduced. This fact can only happen as a result of an exchange of users between two different EMCs. In order to perform an exchange, the initiator randomly selects one of the applications for which it has a running instance. Let $p_i(t)$ be the probability for a generic node to have a service A_i at time t. The probability for the initiator to select an instance of A_i is then $\frac{p_i(t)}{\sum_k p_k(t)}$. The exchange can be performed only in case the receiver has a running instance of A_i, too. This is done to reduce the number of instances, avoiding to just transfer a service from one EMC to another. Obviously, the probability of the receiver to have such an instance is $p_i(t)$.

However, these probabilities are not enough to determine whether an exchange can be done or not. In particular, we have to consider the weight (in terms of resource usage) brought by each instance of a service. We assume that this weight is a combination of the application baseline requirements and the number of users served by an instance. Since each EMC has a limited number of resources, a node stops accepting new users for its running instances whenever it has exhausted its own available resources. To reflect this behaviour in the model, we assume that our generic node is characterised by a maximum weight it can support for hosting applications. This value corresponds to the expected value of the maximum weight sustainable by the nodes in the system, and it is denoted with $\mathbb{E}[W] = \widetilde{W}$.

The weight sustained by a node at time t is $W(t) = \sum_k w_k(t)p_k(t)$, where w_i is the total cost of executing an instance of A_i at time t, i.e. $w_i(t) = w_i^{fix} + w_i^u u_i(t)$. In this formula, w_i^{fix} is the fixed or baseline cost of running an instance of A_i, while w_i^u is the corresponding cost-per-user. Moreover, $u_i(t)$ is the number of users of an instance of A_i at time t. It is computed as the average number of users per instance at time t of A_i, i.e. $u_i(t) = U_i/\varrho_i(t)$, where U_i is the total number of users of A_i.

The probability for an exchange to be successful depends on the fact that the final weight of the receiver should not exceed the maximum weight. For each application A_i, we compute this probability using a function Δ_i. It evaluates whether the exchange at a time $t+1$ can be done by respecting the maximum weight of a node. It is defined as:

$$\Delta_i(t+1) = \begin{cases} 1 - \frac{W_i(t) + w_i^u u_i(t)}{\widetilde{W}} & \text{if } W_i(t) + w_i^u u_i(t) \leq \widetilde{W} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The increase in the weight is given by the increase of the number of users, since an instance of A_i should already be active, i.e. its fixed cost is already considered. The number of users to be added is computed from the previous time step. As can be seen, the higher the load of a node, the lower the probability to accept an exchange. The exchange is rejected in case the resulting final weight exceeds the maximum allowed.

The other factor that influences the possibility to make an exchange is related to the latency constraints. An EMC cannot serve users in case they could experience a latency above a given threshold. This threshold depends on the kind of services provided by the application requested by the users. Thus, each A_i has its own latency limit. We define with $\widetilde{U}_i^L$ the expected maximum number of users of A_i that can be served by a generic node on the basis of A_i's latency constraints. The possibility to perform an exchange for A_i is computed using the function Λ_i. It is defined as:

$$\Lambda_i(t+1) = \begin{cases} 1 & \text{if } \frac{U_i}{\varrho_i(t)-1} \leq \widetilde{U}_i^L \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The function consider whether the latency-based maximum number of users is exceeded by shutting down one instance of A_i (with respect to the situation at the previous time step) in an exchange. Whenever this fact could happen, the exchange is reject.

Finally, by putting together all the facts exposed so far, we can define the probability $\mu_i(t+1)$ to shut down a replica at time $t+1$ as:

$$\begin{aligned} \mu_i(t+1) &= \frac{p_i(t)}{\sum_k p_k(t)} p_i(t)\Delta_i(t+1)\Lambda_i(t+1) = \\ &= \frac{p_i(t)^2}{\sum_k p_k(t)} \Delta_i(t+1)\Lambda_i(t+1) \end{aligned} \tag{3}$$

Since an exchange involves a pair of nodes, we consider that the maximum number of exchanges h involving A_i that could happen at time $t+1$ is at most half the number of nodes hosting an instance of A_i, i.e $h = \lfloor \varrho_i(t)/2 \rfloor$. Each of these changes is an independent trial, with all the trails having the same success probability. We can describe these trials using a binomial distribution. As a result, we can compute the expected number of successful exchanges S_h^i (i.e., shut down instances) as $\mathbb{E}[S_h] = h \cdot \mu_i(t+1)$.

Using these notations, we can describe the evolution over time of ϱ_i as:

$$\varrho_i(t+1) = \varrho_i(t) - h\mu_i(t+1) \tag{4}$$

The evolution of p_i can be easily derived from the one of ϱ_i:

$$p_i(t+1) = \varrho_i(t+1)/n \tag{5}$$

Theorem 2.1. *ϱ_i is a monotonic decreasing function.*

Proof. From the definition of Equation 4, this fact can be easily proved by noting that $h = \lfloor \varrho_i(t)/2 \rfloor$ and $\mu_i(t+1) \in [0,1]$. Thus, $h\mu_i(t+1) \geq 0, \forall t$. As a result, $\varrho_i(t+1) \leq \varrho_i(t), \forall t$.

□

3 MODEL VALIDATION

In order to evaluate the accuracy of the proposed mathematical model, in this section we compare the results of the model with the ones of the original algorithm. To this end, we use a simulation of an Edge environment.

The simulation environment has been modeled using the well-known EUA Dataset[1], collected by Lai *et al.* [10]. As a consequence, the EMCs are placed on an area that reflects the locations of real edge servers in the Melbourne area, as communicated by the Australian Communications and Media Authority (ACMA). As in Lai *et al.*, we assume that each server covers an area whose range is randomly selected in the interval [450m,750m]. In order to exploit a realistic model of the applications' resource footprints on edge resources, we use the Alibaba cluster traces[2]. However, the Alibaba dataset is extremely vast and complex. We derive the description of the applications to use in our experiments by performing a clustering process over all the data. We use a standard k-means algorithm, setting $k = 10$. The centroids of the clusters resulting from this process has been used as the reference set of applications for our experiments. From this set, we extracted the applications used for the comparison.

In this target scenario, we have 125 EMCs. Each of these EMCs is characterized by just one type of resource: the available memory. This resource has the same limit for all the EMCs, that has been set to 8192 MB. In the following experiments, we consider 3 different types of applications. Their characteristics as listed in Table 2. In the table, we have assumed that each user impacts the resources for a quantity that is equal to 12.5% of the fixed cost of the application. Users are initially placed uniformly at random in the simulation area. An instance of their requested application is activated on their closest EMC, to which they are initially assigned.

Table 2: Fixed and Per-user Costs for 3 Applications

Type	Fix. Cost (MB)	Var. Cost (MB)
A_1	587	73.38
A_2	898	112.25
A_3	858	107.25

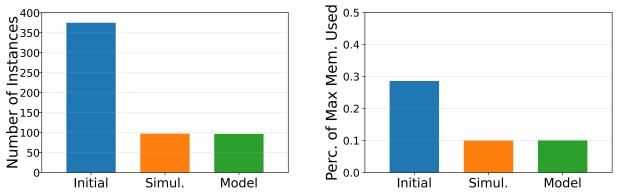
In the following, we propose a comparison using 125, 250 and 375 users per applications. This implies a total of 375, 750 and 1125 users in the system as a whole. For the latency constraints, we consider that the service instances can cover at most 3.9 clients on average with 375 users; 6.5 users with 750 users; 8.9 with 1125 users.

The results of the comparison are presented in Figures 1, 2 and 3. In all this figures, the left-hand side reports a comparison between the starting number of instances, the final result of the simulation and the result of the model. The right-hand side shows the corresponding results for the average resource consumption at each EMC. This quantity is reported as the ratio between the occupied resource and the maximum available space.

In all the figures, it is possible to observe that the result of the model is extremely close to the one of the simulation. This fact highlights the ability of the model to closely describe the behaviour of the simulated system. It is also worth noting that the adopted scheme allows the system to achieve a considerable results in terms of both the number of running instances and the percentage of occupied resources. Both these quantities are significantly reduced with respect to the initial configuration, thus highlighting the efficacy of the proposed solution.

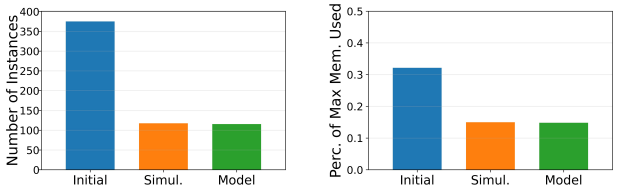
[1] https://github.com/swinedge/eua-dataset

[2] https://github.com/alibaba/clusterdata



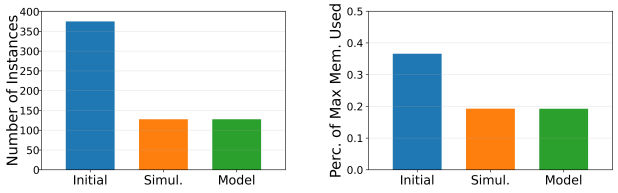
(a) Numb. of Instances **(b) Avg. Resource consumption**

Figure 1: Model validation with 125 users per app.



(a) Numb. of Instances **(b) Avg. Resource consumption**

Figure 2: Model validation with 250 users per app.



(a) Numb. of Instances **(b) Avg. Resource consumption**

Figure 3: Model validation with 375 users per app.

4 CONCLUSION

Next generation applications can benefit from a widely distributed computing infrastructure. An infrastructure that encompasses traditional cloud resources as well as edge devices. Such infrastructures have the potentiality to allow a service provisioning, to end-users, able to satisfy highly demanding QoE. In fact, when the application is hosted in a resource that is close to the end-user, the QoE offered tends to be higher. However, as edge resources are typically constrained in terms of performances and more expensive in cost, a relevant challenge in such a context is the ability of balance the usage of edge resources and the clouds one, while guaranteeing the level of QoE requested by end-users.

In a previous work, we presented an approach that in a fully decentralized way, is able to perform the balance we mentioned above. In this work we provide a mathematical formulation of our previously proposed solution as well as the validation of the proposed model against a simulation of the system. The preliminary results we achieved are promising, showing that the formulation we conceived is able to model the behavior of the target algorithm.

ACKNOWLEDGMENTS

This work has been partially supported by the EC, under the project ACCORDION (Grant agreement ID: 871793).

REFERENCES

[1] J. Altmann, B. Al-Athwari, E. Carlini, M. Coppola, P. Dazzi, A. J. Ferrer, N. Haile, Y.-W. Jung, J. Marshall, E. Psomakelis, et al. 2017. BASMATI: an architecture for managing cloud and edge resources for mobile users. In *Int. Conf. on the Econ. of Grids, Clouds, Syst, and Serv.* Springer, Cham, 56–66.

[2] F. Baiardi, A. Bonotti, L. Ferrucci, L. Ricci, and P. Mori. 2003. Load balancing by domain decomposition: the bounded neighbour approach. In *Proc. of 17th European Simulation Multiconference.* 9–11.

[3] Massimiliano Bertolucci, Emanuele Carlini, Patrizio Dazzi, Alessandro Lulli, and Laura Ricci. 2016. Static and dynamic big data partitioning on apache spark. In *Parallel Computing: On the Road to Exascale.* IOS Press, 489–498.

[4] E. Carlini, M. Coppola, P. Dazzi, M. Mordacchini, and A. Passarella. 2016. Self-optimising decentralised service placement in heterogeneous cloud federation. In *2016 IEEE 10th Int. Conf. Self-Adapt. Self-Organ. Syst. (SASO).* IEEE, 110–119.

[5] Marco Conti, Matteo Mordacchini, Andrea Passarella, and Liudmila Rozanova. 2013. A Semantic-based Algorithm for Data Dissemination in Opportunistic Networks. In *7th Int. Workshop on Self-Organ. Syst. (IWSOS 2013).* Springer, 14–26.

[6] Luca Ferrucci, Matteo Mordacchini, Massimo Coppola, Emanuele Carlini, Hanna Kavalionak, and Patrizio Dazzi. 2021. Latency Preserving Self-optimizing Placement at the Edge. In *Proceedings of the 1st Workshop on Flexible Resource and Application Management on the Edge.* ACM, 3–8.

[7] L. Ferrucci, L. Ricci, M. Albano, R. Baraglia, and M. Mordacchini. 2016. Multidimensional range queries on hierarchical Voronoi overlays. *J. Comput. System Sci.* (2016).

[8] Qiang He, Guangming Cui, Xuyun Zhang, Feifei Chen, Shuiguang Deng, Hai Jin, Yanhui Li, and Yun Yang. 2019. A game-theoretical approach for user allocation in edge computing environment. *IEEE Transactions on Parallel and Distributed Systems* 31, 3 (2019), 515–529.

[9] I. Korontanis, K. Tserpes, M. Pateraki, L. Blasi, J. Violos, Ferran Diego, E. Marin, N. Kourtellis, M. Coppola, E. Carlini, Z. Ledwoń, P. Tarkowski, T. Loven, Y. G. Rozas, M. Kentros, M. Dodis, and P. Dazzi. 2021. Inter-Operability and Orchestration in Heterogeneous Cloud/Edge Resources: The ACCORDION Vision. In *1st Workshop on Flexible Resource and Application Management on the Edge (FRAME '21).* ACM, 9–14.

[10] Phu Lai, Qiang He, Mohamed Abdelrazek, Feifei Chen, John G. Hosking, John C. Grundy, and Yun Yang. 2018. Optimal Edge User Allocation in Edge Computing with Variable Sized Vector Bin Packing. In *ICSOC.*

[11] Luyang Liu, Hongyu Li, and Marco Gruteser. 2019. Edge Assisted Real-Time Object Detection for Mobile Augmented Reality. In *The 25th Proc. annu. Int. Conf. Mob. Comput. Netw. (MobiCom '19).* ACM, Article 25, 16 pages.

[12] Deepthi K Madathil, Rajani B Thota, Paulina Paul, and Tao Xie. 2008. A static data placement strategy towards perfect load-balancing for distributed storage clusters. In *2008 IEEE Int. Symp. on Par. and Distr. Proc.* IEEE, 1–8.

[13] Antonios Makris, Abderrahmane Boudi, Massimo Coppola, Luís Cordeiro, Massimiliano Corsini, Patrizio Dazzi, Ferran Diego Andilla, Yago González Rozas, Manos Kamarianakis, Maria Pateraki, et al. 2021. Cloud for Holography and Augmented Reality. In *2021 10th IEEE Int. Conf. Cloud Comput.* IEEE, 118–126.

[14] Moreno Marzolla, Matteo Mordacchini, and Salvatore Orlando. 2006. A p2p resource discovery system based on a forest of trees. In *17th International Workshop on Database and Expert Systems Applications (DEXA'06).* IEEE, 261–265.

[15] Matteo Mordacchini, Marco Conti, Andrea Passarella, and Raffaele Bruno. 2020. Human-centric data dissemination in the IoP: Large-scale modeling and evaluation. *ACM Trans. Auton. Adapt. Syst.* 14, 3 (2020), 1–25.

[16] Matteo Mordacchini, Patrizio Dazzi, Gabriele Tolomei, Ranieri Baraglia, Fabrizio Silvestri, and Salvatore Orlando. 2009. Challenges in designing an interest-based distributed aggregation of users in p2p systems. In *2009 International Conference on Ultra Modern Telecommunications & Workshops.* IEEE, 1–8.

[17] Qinglan Peng, Yunni Xia, Zeng Feng, Jia Lee, Chunrong Wu, Xin Luo, Wanbo Zheng, Shanchen Pang, Hui Liu, Yidan Qin, et al. 2019. Mobility-aware and migration-enabled online edge user allocation in mobile edge computing. In *2019 IEEE International Conference on Web Services (ICWS).* IEEE, 91–98.

[18] Jinke Ren, Guanding Yu, Yinghui He, and Geoffrey Ye Li. 2019. Collaborative cloud and edge computing for latency minimization. *IEEE Transactions on Vehicular Technology* 68, 5 (2019), 5031–5044.

[19] Pedro Ruivo, Maria Couceiro, Paolo Romano, and Luis Rodrigues. 2011. Exploiting total order multicast in weakly consistent transactional caches. In *2011 IEEE 17th Pacific Rim International Symposium on Dependable Computing.* IEEE, 99–108.

[20] Yulei Wu, Sukhdeep Singh, Tarik Taleb, Abhishek Roy, Harpreet S Dhillon, Madhan Raj Kanagarathinam, and Aloknath De. 2021. *6G mobile wireless networks.* Springer.