POLITECNICO DI TORINO
Repository ISTITUZIONALE

Scalable K-Nearest Neighbors Implementation using Distributed Embedded Systems

Original
Scalable K-Nearest Neighbors Implementation using Distributed Embedded Systems / DE SIO, C., Avignone, A., Sterpone, L., Chiusano, S.. - (2024), pp. 314-315. (CF' 24: 21st ACM International Conference on Computing Frontiers Ischia (ITA) May 7-9, 2024) [10.1145/3649153.3652994].

Availability:
This version is available at: 11583/2988853 since: 2024-05-19T21:25:04Z

Publisher:
ACM

Published
DOI:10.1145/3649153.3652994

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
ACM postprint/Author's Accepted Manuscript

(Article begins on next page)

18 November 2026

POSTER: Scalable K-Nearest Neighbors Implementation using Distributed Embedded Systems

Corrado De Sio, Andrea Avignone, Luca Sterpone, Silvia Chiusano
Department of Control and Computer Engineering (DAUIN)
Politecnico di Torino, Turin, Italy
corrado.desio@polito.it, andrea.avignone@polito.it, luca.sterpone@polito.it, silvia.chiusano@polito.it

ABSTRACT

The distributed embedded systems paradigm is a promising platform for high-performance embedded applications. We present a distributed algorithm and system based on cost-effective devices. The proof of concept shows how a parallelized approach leveraging a distributed embedded platform can address the computational of the Machine Learning K-Nearest Neighbors (K-NN) algorithm with large and heterogeneous datasets.

CCS CONCEPTS

•Computer systems organization~Embedded and cyber-physical systems~Embedded systems~Embedded software

KEYWORDS

Distributed Systems, Embedded Systems, K-NN, Machine Learning, SoC

ACM Reference format:

Corrado De Sio, Andrea Avignone, Luca Sterpone and Silvia Chiusano. 2018. Scalable K-Nearest Neighbors Implementation using Distributed Embedded Systems. In *Proceedings of 21st ACM International Conference on Computing Frontiers (CF'24). May 7-9, 2024, Ischia, Italy.* https://doi.org/10.1145/3649153.3652994

1 Introduction

In recent years, embedded systems have been established as versatile solutions for many application fields. The advent of Systems-on-Chip (SoC) paved the way for embedded platforms with good performance and high connectivity at a reduced cost. As a result, they soon attracted interest as a platform for Machine Learning (ML) algorithms. ML tasks often rely on dedicated hardware accelerators to enhance overall efficiency. Parallel computation has emerged as a valid strategy to overcome performance bottlenecks. The distributed embedded systems paradigm, where multiple SoCs collaborate, is a promising platform for scalable and highly efficient computation [1][2]. The advantages of distributed systems characterize such hardware architecture.

K-Nearest Neighbors (K-NN) is a fundamental *lazy* supervised machine learning algorithm for classification and regression tasks [3]. The central idea behind K-NN is to predict the class or value of a new data point based on the K nearest points of the training dataset to the new data point. K-NN's computational demand significantly increases with large datasets, thus triggering the need for optimized and distributed solutions [4].

This paper presents a distributed architecture and approach for speeding up a K-NN classifier. We show proof of how leveraging a cluster of cost-effective embedded devices makes it possible to address the computational challenges in the context of the K-NN algorithm.

2 Platform and Methodology

The approach relies on a distributed algorithm running on an embedded cluster system. Fig. 1 shows the general pipeline for distributing the K-NN algorithm. It consists of dynamically splitting the training dataset across the nodes currently available in the system. As a result, each node identifies the set of local neighbors, by computing the distances between the test data and each sample in its local portion of training data. Once the computation is complete, the nodes return their K candidates to be aggregated. Thus, the global nearest K points are detected, and the final classification is performed according to the majority class of the neighbors.

The proposed architecture involves different actors: (i) a catalog to manage the partitioning of the dataset among nodes, the registration of nodes, and the final aggregation of results for classification; (ii) N computational nodes to carry out the computation on the assigned dataset portion; (iii) web sockets, for handling the communication between nodes and catalog.

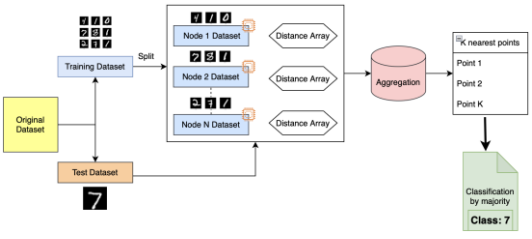


Fig. 1: Conceptual architecture of distributed K-NN.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
CF '24, May 7–9, 2024, Ischia, Italy
© 2024 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0597-7/24/05.
https://doi.org/10.1145/3649153.3652994

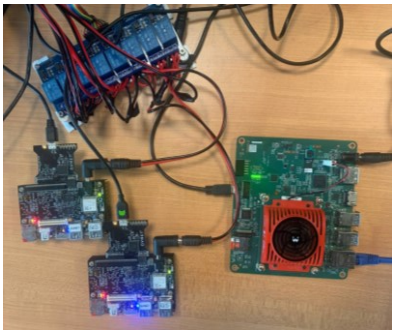


Fig. 2: Embedded Distributed Platform with three nodes.

The registration of nodes is plug-and-play. The system dynamically adapts and instruments the nodes accordingly with the nodes currently available in the system.

The hardware architecture is based on Zynq UltraScale+ MPSoC devices [5]. We used an heterogeneous cluster, including an XCK26 on a KV260 board as a catalog node. On the same node, a computation node instance runs as well, representing the first node of the system that is always present. Additional computation nodes are implemented using Ultra96v2 boards that embed a ZU3EG. KV260 has 4 Gb DDR, while Ultra96v2 is equipped with 2 Gb LPDDR. Ultra96v2 uses a IEEE 802.11 b/g/n WiFi interconnection, while KV260 uses 1GbE. A router is used as the single access point for the network. All the platforms use an OS based on Ubuntu 22.04. Fig. 2 shows an example of the system with three nodes.

3 Experimental Results

Two different datasets have been used as case studies. (i) MNIST dataset [6], providing a set of grayscale images of handwritten digits with ten classes (i.e., numbers from 0 to 9). The training set consists of 60,000 images. (ii) Covertype dataset [7], focusing on the classification of forest cover type according to cartographic data (e.g., elevation, slope). The training set consists of 464,809 samples.

They both achieve good classification results with standard K-NN (accuracy > 90%), but they are prone to longer computation times due to their size and heterogeneity.

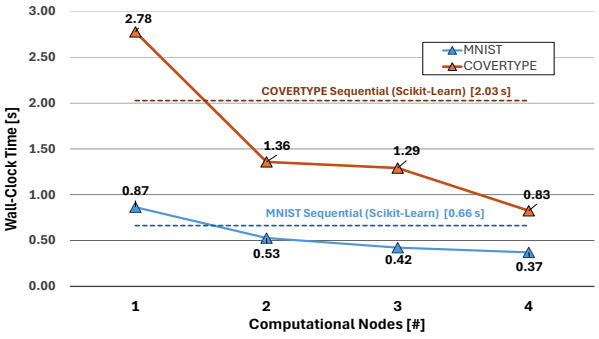


Fig. 3: Measured Wall-Clock time with a different number of computational nodes for 5-NN. The sequential Scikit-Learn solution is showed as a reference.

Fig. 3 reports the obtained results in terms of Wall-Clock time varying N for an inference of 10 samples with our distributed architecture (K=5). For reference, we included the wall clock time using a traditional implementation based on Python Scikit-Learn. For completeness, we also included the performance of our system when relying on a single node. Using a single node, we have a degradation of performance since the overhead introduced by the architecture negatively impacts the computational time. Instead, relying on the distributed architecture (N>1), we observe a progressive performance improvement, up to 57% for MNIST and 70% for the Covertype dataset with N=4. Clearly, our solution does not influence the classification accuracy which is preserved.

4 Conclusions and Future Works

We introduced a promising architecture and methodology to exploit embedded distributed clusters based on cost-effective devices to significantly reduce the time required for K-NN classification by distributing the overall effort. In the future, we want to explore how to increase further performance of different ML models by relying on programmable hardware available on the SoCs, combining the high flexibility provided by the software approach with performance that can be provided by custom hardware.

ACKNOWLEDGMENTS

This study was partially carried out within the FAIR - Future Artificial Intelligence Research - and received funding from the European Union Next-GenerationEU (PNRR MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 D.D. 1555 11/10/2022, PE00000013) and partially supported by the SmartData@PoliTO center on Big Data and Data Science. This paper reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

REFERENCES

[1] L. Bozzoli et al., "EuFRATE: European FPGA Radiation-hardened Architecture for Telecommunications," 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE), Antwerp, Belgium, 2023, pp. 1-6, doi: 10.23919/DATE56975.2023.10137035.

[2] O. Knodel, A. Georgi, P. Lehmann, W. E. Nagel and R. G. Spallek, "Integration of a Highly Scalable, Multi-FPGA-Based Hardware Accelerator in Common Cluster Infrastructures," 2013 42nd International Conference on Parallel Processing, Lyon, France, 2013, pp. 893-900, doi: 10.1109/ICPP.2013.106.

[3] T. Cover and P. Hart, "Nearest neighbor pattern classification," in IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, January 1967, doi: 10.1109/TIT.1967.1053964.

[4] W. Zhang, X. Chen, Y. Liu, and Q. Xi, "A Distributed Storage and Computation k-Nearest Neighbor Algorithm Based Cloud-Edge Computing for Cyber-Physical-Social Systems," in IEEE Access, vol. 8, pp. 50118-50130, 2020, doi: 10.1109/ACCESS.2020.2974764.

[5] AMD. Zynq UltraScale+ Device Technical Reference Manual, UG1085, v2.4, December 2023.

[6] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 29(6), 141–142.

[7] Blackard,Jock. (1998). Covertype. UCI Machine Learning Repository. https://doi.org/10.24432/C50K5N.