A Node-Level Managing System for ML-based Autonomic Operation

Alexandros Patras*, Foivos Pournaropoulos*

* *Department of Electrical and Computer Engineering, University of Thessaly, Greece*

ABSTRACT

The management of a computation node that is a member of a large cluster spanning the cloud-edge continuum is a cumbersome task for human operators. There are numerous decisions that must be made, considering many variables that come from different environments. We propose an autonomic management system for the single node that uses machine learning methods to optimize various configuration options. We have developed a prototype that includes the necessary configuration mechanisms for many management aspects, including a custom middleware for the utilization of acceleration units. We also have integrated a telemetry system to collect data for machine-learning models. This prototype offers the necessary study ground to explore and develop machine learning models that will interact with the configuration mechanisms and the node environment.

KEYWORDS: machine learning; cloud-edge computing; Internet of Things; autonomic management;

1 Introduction

The advancement of Edge Computing and the broader use of IoT devices have increased the management complexity of the underlying infrastructure. Applications may be deployed on a cluster of nodes that are heterogeneous and physically distributed. Different hardware architectures and unique characteristics of each node present the biggest challenges for application development, deployment, and system management. Container technology was developed to handle the issues of the application lifecycle and many tools have emerged around the container ecosystem that implement different solutions, managing high-level aspects, such as the application version that is deployed to specific nodes. We adopt this technology and assume that the applications run in containers. There are also lower-level configuration mechanisms, that reach the level of the physical operation properties of the system. We try to focus on configuration options between those two boundaries.

There are works in the literature that target specific configurations parameters, such as the frequency and voltage of a CPU [KKA+19], [KALB22] and DRAM [KAB+20]. Another aspect of the node level that has been studied, is the exploitation of container-specific configurations like the CPU-pinning functionality, to improve performance and cache behavior [SDBS20]. The existence of hardware accelerators on edge devices has also been a topic that

[2]E-mail: { patras, spournar } @uth.gr

has been studied, and some works try to optimize their usage in the container environment [PCK+21].

To the best of our knowledge, no work tackles the problem of configuring multiple options in tandem. An indicative example would be the following: for a given application with specific requirements we must decide on (i) the application version; (ii) the CPU affinity of the application container; (iii) the use of an accelerator from a container; (iv) the frequency of the computation and accelerator units. It becomes apparent that a human operator might not be able to cope with the complexity of the management, especially in the case of many different managed nodes. An autonomic system, on the other hand, can adapt to any environmental or application changes without the intervention of a human operator, keeping the system operational.

In this work, we consider a computing system that includes heterogeneous nodes spanning the cloud-edge continuum, and focus on the operation of a single node. The main contribution of our work is the development of a managing system for the node that uses Machine Learning for its operation. The design of the system has been inspired by the paradigm of an autonomic system that follows 4 steps: (i) monitor the managed system; (ii) analyze the data; (iii) plan a new configuration if needed; and (iv) execute the new configuration if possible [GSC09]. The analysis and planning steps include the use of machine learning models. The managing system uses and sends commands to available configuration mechanisms for the different subsystems and is also designed to include necessary interfaces to interact with different machine learning models. Due to being part of a cluster, the managing system also receives input from the external environment.

2 An ML-ready managing system

Figure 1 shows the 3 main entities of the managing system: i) Telemetry Collector, ii) Machine Learning Models, and iii) Configuration Mechanisms. The workflow of the system starts by collecting telemetry data from different sources, including the application, transforming, and then exporting it to suitable machine learning models. The models analyze the data and generate a new plan for the configuration mechanisms to apply.

Our approach is to use different machine learning models that produce configuration plans for a single or a group of configuration mechanisms and is based on the fact that actions from a specific mechanism are best correlated with different features that are extracted from specific telemetry data, compared to a one-size fits all alternative. Machine learning models need adequate data quantity and quality for their training. A computing system can produce telemetry data from every layer of its software and hardware stack, containing a lot of noise and must be normalized to ease the ingestion into an ML model. We use a suitable telemetry implementation that includes the necessary data collection and transformations [OTE]. Our system includes arbitrary application-level metrics that record the application performance.

A general overview of the proposed mechanisms that can be used in a single node is also illustrated in Figure 1. The main element is the general computation unit (e.g. CPU). Our work focuses on exploring the energy efficiency space by configuring the frequency or the power cap of those units.

Hardware acceleration units are widely used on many kinds of edge nodes and not only in data centers. Our work explores the smart and transparent use of hardware accelerators.

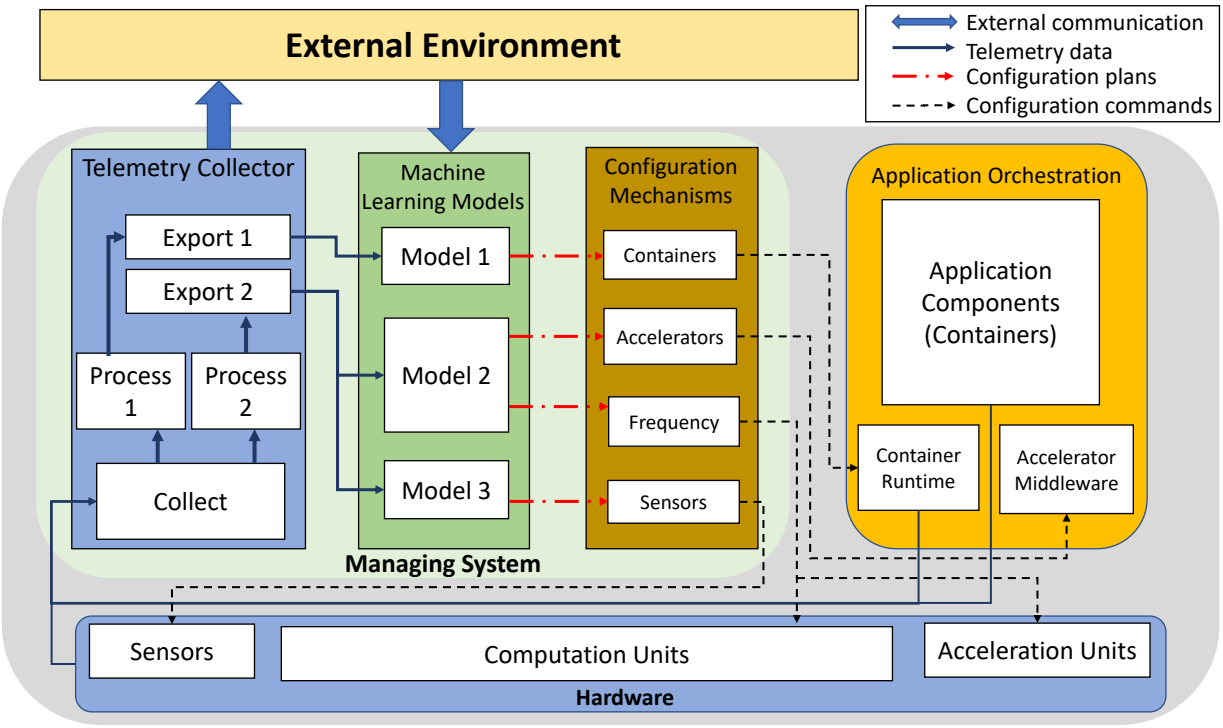

Figure 1: Node-level architecture of the proposed managing system

This choice is made initially when a container is placed on the node, and then necessary changes might be made at runtime. We use a custom middleware, that abstracts out the actual implementation of a computation task for different hardware accelerators (GPU, FPGA), and offer a generic API to the application. The application code calls a generic method that performs the required functionality, and the middleware decides in every call what accelerator will be used. This assumes that the actual hardware acceleration implementations of each computation kernel is available and installed in each node.

Another aspect of container orchestration that can be managed is the affinity of a container and its resource requirements. Affinity enforcement mechanisms can pin a container to specific CPU cores. This functionality can help to improve performance for specific tasks by leveraging better cache behavior. Resource requirements, on the other hand, are usually managed from the orchestrator, where an application container explicitly defines its own hardware requirements, such as the CPU core count and memory quantity. The proposed system provides the necessary functionality to make changes at runtime, changing the given resources for each container.

Sensors are available mostly on smart edge nodes with different characteristics and usages. The kind of sensors can vary from a simple camera module to environmental conditions sensing modules. The configuration mechanisms contain the necessary functionality to offer abstract configuration options, such as the sensing frequency, to the machine learning models.

3 Future work

We have developed a prototype where the proposed managing system can be tested with the appropriate software stack implementing the necessary telemetry pipeline and configuration mechanisms, as shown in Section 2. The next step is to explore, develop and test suitable machine-learning models that will exploit the presented mechanisms. Another aspect that needs further research, is the fact that the node-level operation is not only affected by the node-level system and application requirements but also by an external environment.

Acknowledgments

This work has been supported by the European Community's Horizon Europe Program under Grant Agreement nr. 101092912, project MLSysOps.

References

[GSC09] David Garlan, Bradley Schmerl, and Shang-Wen Cheng. *Software Architecture-Based Self-Adaptation*, pages 31–55. 04 2009.

[KAB+20] Christos Kalogirou, Christos D. Antonopoulos, Nikolaos Bellas, Spyros Lalis, Lev Mukhanov, and Georgios Karakonstantis. Increasing the profit of cloud providers through dram operation at reduced margins. *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pages 549–558, 2020.

[KALB22] Christos Kalogirou, Christos D. Antonopoulos, Spyros Lalis, and Nikolaos Bellas. Dynamic management of cpu resources towards energy efficient and profitable datacentre operation. In *Job Scheduling Strategies for Parallel Processing*, 2022.

[KKA+19] Christos Kalogirou, Panos K. Koutsovasilis, Christos D. Antonopoulos, Nikolaos Bellas, Spyros Lalis, Srikumar Venugopal, and Christian Pinto. Exploiting cpu voltage margins to increase the profit of cloud infrastructure providers. *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 302–311, 2019.

[OTE] Opentelemetry. `https://opentelemetry.io`.

[PCK+21] Jung-Gi Park, Un-Sook Choi, Seungwoo Kum, Jaewon Moon, and Kyungyong Lee. Accelerator-aware kubernetes scheduler for dnn tasks on edge computing environment. *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 438–440, 2021.

[SDBS20] Davood Ghatreh Samani, Chavit Denninnart, Josef Bacik, and Mohsen Amini Salehi. The art of cpu-pinning: Evaluating and improving the performance of virtualization and containerization platforms. *Proceedings of the 49th International Conference on Parallel Processing*, 2020.