An Edge Computing Sizing Tool for Robotic Workloads

Ahmad Rzgar Hamid
arh@mmmi.sdu.dk
University of Southern Denmark
Maersk Mc-Kinney Moller Institute
Odense, Denmark

Mikkel Baun Kjærgaard
mbkj@mmmi.sdu.dk
University of Southern Denmark
Maersk Mc-Kinney Moller Institute
Odense, Denmark

ABSTRACT

Robots of today are equipped with lightweight computing resources used merely to make the robot function. However, proportional advancements in associated data processing and algorithms are needed, given the significant advances in robots' sensing and programming capabilities and the increasingly complex tasks they must complete.

Yet, such advancements require additional hardware resources to function as intended. In many robotic applications, cloud computing is not an option; therefore, edge computing must be embraced.

This paper proposes a sizing tool for benchmarking workloads against pre-written tasks to determine optimal edge computing hardware candidates used to deploy said workloads efficiently without wasting resources.

Preliminary results show that the right combination of hardware resources has an impact on workload execution.

CCS CONCEPTS

• **General and reference** → **General conference proceedings**; • **Hardware** → **Hardware test**; • **Computer systems organization** → **Heterogeneous (hybrid) systems**; • **Software and its engineering** → **Software infrastructure**; **Software architectures**; **Software performance**.

KEYWORDS

Edge Computing, Workload Benchmarking, Hardware Comparison

ACM Reference Format:
Ahmad Rzgar Hamid and Mikkel Baun Kjærgaard. 2024. An Edge Computing Sizing Tool for Robotic Workloads. In *2024 ACM/IEEE 6th International Workshop on Robotics Software Engineering (RoSE '24), April 15, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3643663.3643969

1 INTRODUCTION

Even though robotics, as seen today, has been a significant advancement in automating physical, labour-intensive workloads, novel visions flourish to go even further. One of these visions is to automate high-mix, low-volume productions at scale within a single manufacturing setting. To accomplish this, production lines must support vast numbers of configurations, even in the production of single products.

Increasing automation cases in high-mix, low-volume productions where robots are financially viable requires innovations in the flexibility of collaborative robots that can operate close to humans. New sensors, data storage, and AI-based data processing are key technologies that enable this. However, these technologies also introduce a need for more computing and data storage capabilities in robotic systems.

Most robots of today are equipped with minimal resources intended to support the robot lifecycle, its world model and lightweight static instructions provided by the user. Robot manufacturers acknowledge this and suggest augmenting completely automated Artificial Intelligence (AI) solutions with additional computing power to ensure reliable operation [14].

The Industry 4.0 (I4.0) vision presents apparent adaptation problems for future robotics, as I4.0 imposes fluidity in production and needs highly data-driven approaches tightly tied to the real-time state of the production line. To overcome said problems, one overarching limitation has to be solved to enable deploying holistic, technically complex, and data-driven workloads. The limited resources available on the robots and the manufacturing floor primarily impose this limitation.

Executing these compute loads requires appropriate hardware. To ensure that the system is not limited by the available hardware, the go-to way of deploying said systems is to acquire an over-the-top hardware setup, in which there is no doubt that enough resources will be available. However, this approach is not cost-effective, as the abundance of resources results in underutilised hardware, increasing the initial purchase and maintenance cost. This ineffective hardware utilisation is the case even for large manufacturing settings that benefit from economies of scale.

Following the already flourishing cloud computing trend, a possible solution to overcome such limitations is intersecting cloud utilisation and robotics. This idea has been named Cloud Robotics [6], which utilises core cloud objects, such as cloud computing and cloud storage. Here, robots have been equipped with a lightweight on-board client that acts as an interface between the robotic edge node and a cloud instance, which implements all the computing and storage capabilities needed to use the robot. With cloud robotics, the need for on-board resources diminishes, often resulting in physically smaller hardware and substantial price decreases.

Yet, further research has identified flaws with the cloud-first strategy, as processing using far-away infrastructure requires the inefficient, high latency, low bandwidth connectivity offered by the Wide Area Network. This proposes notable overheads that are intolerable for real-time or distributed systems in which vast

Permission to make digital or hard copies of all or part 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 components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICSE 2024, April 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0566-3/24/04…$15.00
https://doi.org/10.1145/3643663.3643969

amounts of data must be transferred between nodes. Further attributes, such as sustainability and energy efficiency, fuel the idea that cloud-first might not be the silver bullet it is currently perceived as [15]. Moreover, the inherent trust and control issues of using "someone else's computer" give no guarantees of how data are handled and no absolute control, as the black-box approach, used by cloud providers, enables excellent client isolation but no underlying consumer explanation [8].

As a response to the centralised nature of cloud computing, a new, decentralised computing paradigm, edge computing, has emerged, which adopts the cloud-native way of thinking, with deployment on edge nodes in proximity to the data source [12]. The key arguments for the edge computing paradigm are increased bandwidth with much lower chances of network congestion, low latency communication, and increased control over data management and computing [13, 15].

Edge computing is especially suitable for distributed systems, as the close-by nature of edge computing reduces both the physical and the logical end-to-end distance [8]. System distribution further increases reliability, availability and performance as loads are distributed amongst nodes, reducing the chances of operational failures.

Instead, tools that accurately determine the hardware configuration for deploying such software systems to execute as expected, given predefined conditions, are needed. Implementing such strategies introduces a potential reduction in both initial costs, as the required hardware is often far less than anticipated, and operation costs, as a reduction in hardware resources, decreases the power needed to operate.

This paper aims to introduce a tool that supports such decision-making based on observing software execution and evaluating it against quality attribute-based test cases. The tool monitors the system behaviour, state, and resource usage under given loads and introduces data-driven decision-making.

2 DISTRIBUTION OF COMPUTING LOADS

Realising that all workloads cannot be deployed close to the data source is nothing new. The realisation came to be as the workloads aimed for resource-limited end devices became more and more hardware-intensive. In 2002, research showed improved Quality of Service (QoS) for applications utilising distant servers, to which some of the workloads were offloaded [1]. This new paradigm introduced the idea that not all workloads have to be monoliths deployed on a singular device but could easily be split up into multiple parts, in which each part has specialised responsibilities. As a result, the concept of cloud computing was born, in which specialised cloud providers deploy numerous cloud farms worldwide for consumers to use.

With the rise of cloud computing, new challenges arise, as the dynamic nature of the cloud and the associated load require rapid scaling capabilities to serve value to the consumers within acceptable response time. For this, Virtual Machines (VMs) were not suited, as VMs require time to start, following its virtualisation of a complete operating system with virtual resources. Moreover, the full-fledged operating systems within VMs take up an amount of space equivalent to that of an operating system deployed on bare-metal infrastructure.

As an alternative to deploying VMs in the cloud, a cloud-native approach was formed, in which workloads were isolated using a newborn technology: containers. These were especially suited for deployment on the edge and in the cloud, as they naturally have a low startup time of milliseconds and share kernel with the host operating system. Especially the kernel-sharing nature proved to be a fundamental feature, which reduced the size notably to a few megabytes [10].

This opened up an innovative wave of strategies for both automated, dynamic provisioning of resources and dynamic computation placement [16, 17].

While applicable in the cloud — where resources are numerous — such strategies are less usable at the edge as resources are limited and often chosen to match the load anticipated for such workload. To bridge this gap, the strategies have been augmented with control theory-based methods to extend the strategies underlying predictions by monitoring and comparing the actual load with the predicted, thus regulating even further [5].

Furthermore, a plethora of existing deployment tools have been developed, with the ability to uphold a limited set of quality attributes, such as Kubernetes [7], Nomad [4], OpenStack [9] and OpenShift [11]. These systems have features that provide attributes such as availability and deployability through horizontal scaling and automated rollout and rollback.

However, even with the appealing capabilities of containerised workloads deployed in the cloud, there are underlying drawbacks that suggest considerations should be made before utilising the cloud as is. Utilising a distributed edge-cloud infrastructure assumes that connectivity and bi-directional communication between components deployed at both the edge and the cloud must be established, thus imposing a security risk. Likewise, sharing workload and generated data with cloud providers implies a level of trust in cloud providers that is not there.

Current deployment tools cannot take a quality attribute-based task as input, use the task on an appropriate workload, and output a performance report suggesting hardware characteristics needed to deploy the given case.

While provisional and scaling strategies in edge computing have been extensively researched, little to no attention has been given to the initial determination of usable hardware for a given workload. This determination is essential — especially for manufacturing settings — that purchase hardware engross. Purchasing underfitting hardware results in repurchasing hardware that fits the purpose, creating unnecessary costs. The purchase of overfitting hardware results in a higher-than-necessary purchase price and subsequent higher energy consumption. Inspired by the previous research, this paper proposes a tool that, based on a given workload and an accompanying task, can determine hardware characteristics that must be met to execute the workload adequately.

3 ASSESSMENT TOOL

To design a tool for addressing the problem, we interacted with several robotics researchers and engineers to scope out requirements.

A proof of concept was built from these requirements to deploy and monitor relevant robotic AI workloads.

3.1 Proof of Concept

To get an in-depth understanding of robotic AI workloads and their associated resource usage, a Proof of Concept (PoC) has been built. The PoC utilised heterogeneous clusters of three Kubernetes variants: Vanilla Kubernetes [7], K3s [3] and MicroK8s [2] on which a series of distributed computer vision training tasks were deployed. While running, the executions were monitored to get insights into the container orchestrator's effect on resource- and workload-bound metrics. The workloads contained training of increasingly larger and more complex models with larger datasets, increasing overall resource consumption. From the PoC, several lessons have been learned, for which future development will be considered.

Resource- and workload-bound metrics indicate categories of metrics for which monitoring gives insights into workload execution. Resource-bound metrics are infrastructure-bound, i.e. directly tied to hardware. Examples are CPU Usage, Memory Usage, Disk R/W and Network Tx/Rx. Workload-bound metrics are metrics specified for the execution of a specific workload. For ML training workloads, accuracy and precision could be of interest. For ML inference, accuracy and throughput could be of interest.

At execution, workloads will fill out whatever resources are available as workloads, and the arbitrary frameworks they are built upon are designed to optimise the usage of available resources. This often results in similar readings of resource-bound metrics, indicating similar resource consumption in percentage, regardless of the underlying infrastructure. For some workloads, the extra resources are well-spent and utilised to generate more precise or faster outputs. But for other workloads, the extra resources result in accurate outputs, with response times similar to that without the extra resources. This observation highlights the importance of measuring both resource- and workload-bound metrics to get a holistic understanding of workload execution.

The underlying infrastructure on which a workload is deployed significantly impacts workload output. Even though Kubernetes, K3s, and Microk8s all provide the same functionality, they each have substantial differences in their resource footprint. Further observations show that deployed workloads on said infrastructure yield different results, both in resource-bound metrics, like memory usage, and workload-bound metrics, like model accuracy.

3.2 Requirements

We distilled the lessons from the PoC into the following requirements for creating a technically usable and user-friendly solution.

(1) The tool should contain interfaces for commonly used operating systems, virtual machine managers and container runtimes
(2) The tool should implement an optimisation algorithm such as linear programming, moving average, a search algorithm or a machine learning algorithm to find the optimal hardware
(3) The tool should contain interfaces that allow for other custom optimisation algorithms to be used
(4) The tool should compile a report summarising all the runs and present the candidate hardware
(5) The tool should identify the candidate hardware, assuming the max load equivalent to the tasks generated load

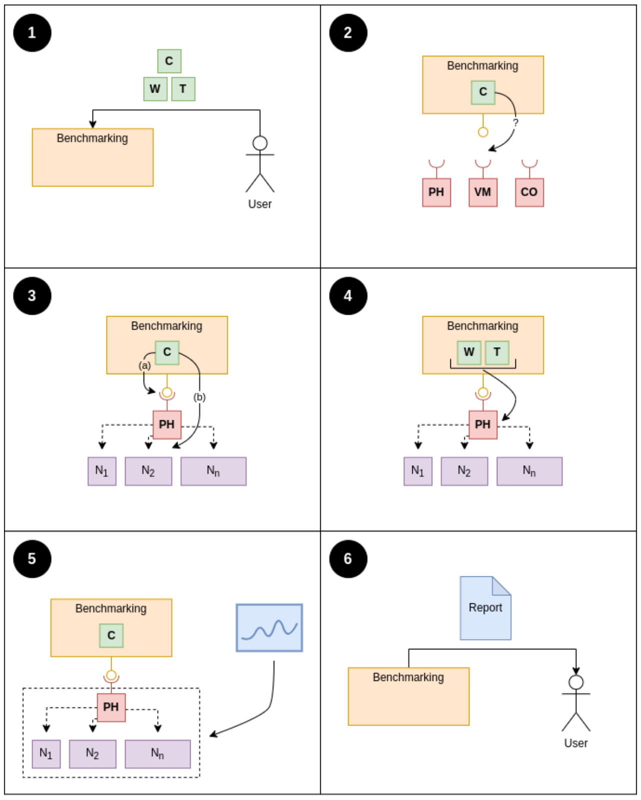

Figure 1: Storyboard, depicting the general workflow.

3.3 Tool Workflow

Given these requirements, a sizing tool would support the general workflow depicted in Figure 1. The workflow assumes that a manager has been installed on a single machine that acts as the interface for the tool. Further assumptions are that connectors for physical — bare metal — infrastructure, virtual machine managers and container runtimes have been installed.

(1) The user submits a workload for benchmarking. The submission consists of three parts: A configuration file containing configurations for the specific benchmarking (C); The workload, which is the system that is going to be benchmarked (W); and The task, that is, a system generating load that the workload is benchmarked against (T).
(2) The tool reads the configuration file and retrieves information about which connector will be used. This specifies if the workload is to be benchmarked on either *PH:* Physical infrastructure, meaning that the workload will be executed on an operating system, installed directly on bare metal, *VM:* Virtual Machine, meaning that the workload will be executed on operating systems utilising virtual resources, *CO:*

Containers, meaning that the workload will be executed using the host kernel.

(3) The tool searches the configuration file for information about affinity to any hardware configurations. *(a)* If the user has specified information about specific hardware configurations to benchmark the workload on, nodes (N_1, N_2, …, N_n) that either directly match or are closest to the specified hardware, are selected. *(b)* If no affinity is provided, the tool chooses a subset of nodes from the available nodes to benchmark the workload.

(4) The tool determines the number of available nodes from the set of chosen nodes and deploys the workload to the nodes. Furthermore, a matching number of tasks are deployed on isolated infrastructure to prevent cross-contamination.

(5) While the task is processing, and the workload is under load, the tool monitors both resource- and workload-bound resources.

(6) When the tool has executed the workload on the chosen set of nodes, a report is returned to the user, summarising the deployments on all nodes. The report contains *A)* A list of all workload deployments and their associated metrics, *B)* A graph depicting the metrics over execution time, and *C)* A single candidate is chosen as the optimal hardware configuration, assuming a maximum load equivalent to the task-generated load.

4 RESEARCH CHALLENGES

Augmenting such sizing tool with further capabilities proposes numerous vital challenges to consider and mitigate for the tool to function as intended.

Firstly, there is a challenge in identifying relevant optimisation algorithms that — based on the resource consumption of a given hardware configuration — can determine which configuration is most suited as a candidate. Here, conflicting attributes, such as performance, energy efficiency and cost, must be included in the optimisation.

Secondly, there is a challenge in bridging between hardware configurations and understanding what effect different configurations have on quality attributes. This bridging has to be semantically explainable to describe the relation between infrastructure and quality attributes. Furthermore, semantics must be specified for defining the relation between workloads, tasks, and associated workload-bound metrics.

Thirdly, there is a challenge in monitoring, logging and reporting the task-generated pressure on any given workload concerning a specified quality attribute.

Lastly, there is a challenge in sectioning and distributing workloads, especially at granularity levels with components as large as containers. Here, concerns arise about the ease with which a workload can be distributed while minimising developer disturbance.

5 CONCLUSION

In this article, we have argued that informed hardware decisions are becoming increasingly important with the rapid increase in robotics implementations, especially in manufacturing. By nature, robots are equipped with lightweight, low-resource computers specifically designed to run the core systems needed for the robot to function, as well as small, static consumer scripts used to customise the robotic behaviour to the given use case.

To enable further augmentation of robotics applications, appropriate hardware that is resourceful enough to execute the given workloads while being cost-effective to deploy at a large scale is needed.

This paper proposes the development of a tool for static workload benchmarking, which will help users make informed choices about deploying a given workload on hardware configurations. The tool utilises established optimisation algorithms combined with a data-driven approach and load-based assessments to determine candidates for optimal hardware configurations for deploying said workload.

ACKNOWLEDGMENTS

This work is supported by DIREC – Digital Research Centre Denmark.

REFERENCES

[1] Rajesh Balan, Jason Flinn, M. Satyanarayanan, Shafeeq Sinnamohideen, and Hen I. Yang. 2002. The case for cyber foraging. *Proceedings of the 10th Workshop on ACM SIGOPS European Workshop, EW 10* (2002), 87–92. https://doi.org/10.1145/1133373.1133390

[2] Canonical. [n. d.]. MicroK8s - Zero-ops Kubernetes for developers, edge and IoT. https://microk8s.io/

[3] Cloud Native Computing Foundation. [n. d.]. K3s. https://k3s.io/

[4] HashiCorp. [n. d.]. Nomad. https://www.nomadproject.io/

[5] Shihong Hu, Weisong Shi, and Guanghui Li. 2022. CEC: A Containerized Edge Computing Framework for Dynamic Resource Provisioning. *IEEE Transactions on Mobile Computing* (2022). https://doi.org/10.1109/TMC.2022.3147800

[6] Ben Kehoe, Sachin Patil, Pieter Abbeel, and Ken Goldberg. 2015. A Survey of Research on Cloud Robotics and Automation. *IEEE Transactions on Automation Science and Engineering* 12, 2 (4 2015), 398–409. https://doi.org/10.1109/TASE.2014.2376492

[7] Kubernetes. 2018. Kubernetes Components. https://kubernetes.io/docs/concepts/overview/components/

[8] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. 2015. Edge-centric Computing: Vision and Challenges. *ACM SIGCOMM Computer Communication Review* 45, 5 (9 2015), 37–42. https://doi.org/10.1145/2831347.2831354

[9] OpenStack. [n. d.]. OpenStack: Open Source Cloud Computing Infrastructure. https://www.openstack.org/

[10] Claus Pahl, Antonio Brogi, Jacopo Soldani, and Pooyan Jamshidi. 2019. Cloud container technologies: A state-of-the-art review. *IEEE Transactions on Cloud Computing* 7, 3 (7 2019), 677–692. https://doi.org/10.1109/TCC.2017.2702586

[11] Red Hat. [n. d.]. OpenShift enterprise Kubernetes container platform. https://www.redhat.com/en/technologies/cloud-computing/openshift

[12] Mahadev Satyanarayanan. 2017. The emergence of edge computing. *Computer* 50, 1 (1 2017), 30–39. https://doi.org/10.1109/MC.2017.9

[13] Weisong Shi and Schahram Dustdar. 2016. The Promise of Edge Computing. *Computer* 49, 5 (5 2016), 78–81. https://doi.org/10.1109/MC.2016.145

[14] Universal Robots. [n. d.]. UR+ | Jetson AGX Xavier Developer Kit for Universal Robots. https://www.universal-robots.com/plus/products/nvidia/jetson-agx-xavier-developer-kit/

[15] Blesson Varghese, Eyal De Lara, Aaron Yi Ding, Cheol Ho Hong, Flavio Bonomi, Schahram Dustdar, Paul Harvey, Peter Hewkin, Weisong Shi, Mark Thiele, and Peter Willis. 2021. Revisiting the Arguments for Edge Computing Research. *IEEE Internet Computing* (2021). https://doi.org/10.1109/MIC.2021.3093924

[16] Tian Ye, Xue Guangtao, Qian Shiyou, and Li Minglu. 2017. An Auto-Scaling Framework for Containerized Elastic Applications. *Proceedings - 2017 3rd International Conference on Big Data Computing and Communications, BigCom 2017* (11 2017), 422–430. https://doi.org/10.1109/BIGCOM.2017.40

[17] Ruiting Zhou, Zongpeng Li, and Chuan Wu. 2018. Scheduling Frameworks for Cloud Container Services. *IEEE/ACM Transactions on Networking (TON)* 26, 1 (2 2018), 436–450. https://doi.org/10.1109/TNET.2017.2781200

Received 07 December 2023; accepted 16 January 2024