A Do-It-Yourself Single-Board-Computer Cluster Used for Teaching High-Performance-Computing

Niklas Schlegel; Silas Della Contrada; Lukas Lange; Marcus Soll; Louis Kobras; Daniel Versick; Jan Haase

2025 IEEE 4th German Education Conference (GECon), 2025/2026

IEEE Xplore: https://ieeexplore.ieee.org/document/11369378

DOI: 10.1109/GECon64629.2025.11369378

A Do-It-Yourself Single-Board-Computer Cluster Used for Teaching High-Performance-Computing

Niklas Schlegel*, Silas Della Contrada*, Lukas Lange*, Marcus Soll†, Louis Kobras†, Daniel Versick†, Jan Haase†
* NORDAKADEMIE gAG Hochschule der Wirtschaft
25337 Elmshorn, GERMANY
Email: {niklas.schlegel.t21a, silas.della_contrada.a21a, lukas.lange.i21c}@nordakademie.org
† NORDAKADEMIE gAG Hochschule der Wirtschaft
25337 Elmshorn, GERMANY
Email: {marcus.soll, louis.kobras, daniel.versick, jan.haase}@nordakademie.de

***Abstract*—Technologies like containerization and parallel computing grow increasingly important. Where containerization might be used to facilitate service availability or development and deployment processes, parallelization can be utilized to tackle problems that are too large to be computed sequentially. Thus, these concepts – pertaining to scalability, availability, and high-performance computing – need to be taught to students. This paper presents a computing cluster that was constructed as a cost-efficient Do-It-Yourself student project which is currently used in High-Performance Computing education. The cluster utilizes Banana Pi Single Board Computers as computing nodes and employs Kubernetes for container management and Ansible for task automation. As the cluster is used in teaching in Computer Science, several educational aspects and applications are discussed.**

***Index Terms*—Parallel Computing, Container Orchestration, Banana Pi, Computing Education, Computing Cluster, DIY, Kubernetes, High-Performance Computing**

I. Introduction

The world has evolved from monolithic desktop applications and sequential calculations to cloud applications and distributed and parallel computing. This is made evident by two exemplary perspectives:

Certainly one of the most challenging issues of our time is climate change. Climate models are exceedingly complex and thus require vast amounts of computational power to simulate. The German Climate Computing Center (*Deutsches Klimarechenzentrum*, DKRZ), employs a computing cluster called *Levante*, which, in total, features 2832 computing nodes, each with two processors à 64 processing cores. This results in a total capacity of 14 petaFLOPS, supplemented by about 800 terabytes of RAM and an additional 60 GPU nodes for even more power.[1] To properly utilize such a system, optimized software and specialized skills are required.

JetBrains' 2023 *State of the Developer Ecosystem* report [2] features a *DevOps and Cloud*[2] section where it is stated that about half of all developers who responded to the survey utilize some form of containerization for development or deployment. In this survey 23% of respondent users noted some form of employment of Kubernetes and about half of all respondents noted that they run several containerized applications at the same time – thus indicating that container orchestration is a relevant technique for current-day development practices.

With container orchestration and parallel computing having this kind of impact, it is therefore imperative to prepare Computer Science students to handle these technologies as part of their available tools. Another place where such learning experiences fit well into is the laboratory, as it enhances theoretical knowledge with practical, hands-on experience through a form of enactive learning.[3] Laboratory education, while not necessarily restricted to physical laboratory spaces, see [4], has long been employed as part of education – Hofstein and Lunetta mention a history that goes back at least as far as the 1890s (cf. [5, p. 202]) – and has been used, for example, for cyber security [6], aerodynamics [7], titration [8], or any other STEM subject that profits from hands-on experience.

To address this educational need, a computing cluster was constructed as a one-time Do-It-Yourself (DIY) part of a student project. The cluster, which is now used as-is as regular teaching device with regards to container orchestration and parallel computing, consists of two banks of Banana Pi M5[4] Single Board Computers (SBCs), and can be seen in Fig. 1. The cluster makes use of industry-standard technologies such as Ansible[5] and Kubernetes to manage nodes and distribute jobs, thus giving students first-hand experience with technologies they are likely to encounter later in their professional life. The cluster was constructed as a student project and has found several applications in teaching concerning the aforementioned

This research was part of the project *Flexibel kombinierbare Cross-Reality Labore in der Hochschullehre: zukunftsfähige Kompetenzentwicklung für ein Lernen und Arbeiten 4.0* (CrossLab) [1], which is funded by the *Stiftung Innovation in der Hochschullehre*, Germany.

[1]https://www.dkrz.de/en/projects-and-partners/projects/focus/levante-spotlight, last accessed 2025-03-27.

[2]https://www.jetbrains.com/lp/devecosystem-2023/devops/, last accessed 2025-03-12.

[3]See [3] for an example of the effect of enactive learning in general.

[4]https://docs.banana-pi.org/en/BPI-M5/BananaPi_BPI-M5, accessed 2025-03-13.

[5]https://docs.ansible.com/ansible/latest/index.html, accessed 2025-03-13.

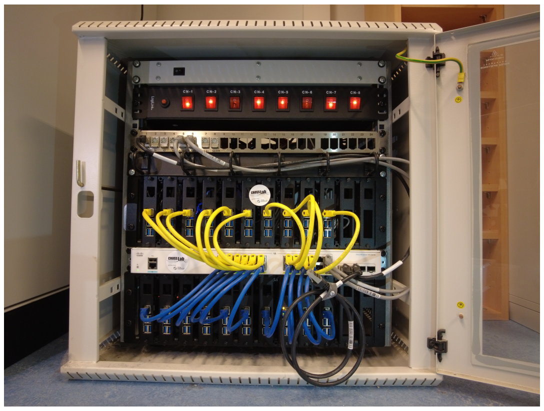

Figure 1. The Banana Pi Cluster. Two banks with each ten Banana Pi M5 and two empty slots are connected to a switch. Each node – courtesy of it being a self-contained Single Board Computer – can be exchanged individually in case of hardware fault.

topics of containerization and parallel processing. Since both the construction of computing clusters in competitions (such as SC's Student Cluster Competition[6]), using SBCs (e.g., [9]), and the use of containerization to deploy microservice infrastructures (e.g., [10]) are known concepts by now, this paper will instead focus on the possible teaching applications of the cluster and the student perspective of the cluster construction process.

II. Requirements for Constructing the Cluster

To achieve the objective of providing inexpensive cluster technology for educational purposes, several requirements must be met to ensure a cost-effective yet sufficiently powerful, effective, and easily maintainable cluster.

The proposed cluster currently consists of 20 Banana Pi M5 (BPI M5) nodes, providing a total of 80 processor cores. The fundamental decision to use Banana Pi M5s as the primary execution units was simply driven by the limited availability of other well-known SBCs, such as the Raspberry Pi 4B, due to supply shortages caused by the COVID-19 pandemic.

Nevertheless the BPI M5 nodes were found to be well-suited for the proposed cluster architecture. Besides offering four processor cores and 4 GB of memory per node, this platform also provides a 1 Gbit/s network interface. As network performance can be a limiting factor in cluster scaling — becoming even more apparent as the number of nodes increases — the availability of appropriately dimensioned hardware is a crucial requirement, as the scalability of the cluster directly impacts its overall efficiency [11]. In this context, network saturation may lead to internode communication becoming the primary bottleneck [12]. This is particularly relevant in educational scenarios, where insufficient network interfaces could impose limitations on the analysis of parallel computing constraints, as network performance may dominate and obscure the intended focus on parallelization efficiency.

[6]https://sc24.supercomputing.org/students/student-cluster-competition/, last accessed 2025-06-02.

The SBCs use the lightweight Linux operating system (OS) Armbian[7], specifically a variant of the Ubuntu Linux distribution built specifically for the BPI M5. Choosing this image was primarily driven by Armbian's ability to support a broad range of different SBCs, thereby enhancing the flexibility of the proposed cluster architecture. As the cluster management is entirely based on Ansible (see Sec.IV), the setup allows for the exchange of SBCs as long as an identical OS is installed. This enhances the cluster's portability by leveraging the flexibility of the chosen OS and its uniform management, ensuring consistent system environments across divergent hardware configurations and thereby increasing the level of hardware abstraction.

Besides this, the proposed cluster also emphasizes a portable yet expandable hardware implementation, featuring uniform off-the-shelf 19-inch rackmounts designed to accommodate credit card-sized SBCs, such as the Raspberry Pi or the utilized BPI M5. These are mounted using standardized, pre-drilled mounting holes within an 3U server rack. In addition, the physical cluster networking is implemented using a dedicated network switch equipped with 1 Gbit/s access ports, enabling the inter-nodes communication. The switch is also connected to the external network (the rest of the lab environment) via 10 Gbit/s uplink ports, providing remote access to the cluster's capabilities and facilitating its remote management.

As mentioned, the cluster is comparably cheap in construction. With an upper estimate of around 4.200€ the acquisition cost is comparable to a high-end desktop system[8] with the added benefit of the much higher processor count and comparatively more memory. Next to the SBCs themselves, the largest cost points were the Switch (a refurbished Cisco CBS350-24P-4X for about 800€) and the server cabinet. We used a cabinet we already had on hand but a comparable unit can be bought for around 600€. Additional but comparatively lower costs came from rack mounts for the SBCs (180€ per bank), power supply (20 Raspberry Pi power supply units, several coupler strips, as well as a main power switch for a total of about 500€), and the required Ethernet cables (28 pieces in total).

III. Teaching Applications of a DIY Computing Cluster

By providing a cluster-based lab environment, a broad range of teaching applications emerges, including concepts related to parallel computing, container orchestration [13], and infrastructure services such as distributed storage systems like CephFS[9] or MinIO[10] [14]. Since the underlying project focuses primarily on distributed parallel execution using the Message Passing Interface (MPI)[11] and container orchestration with Kubernetes, the following section will concentrate on these specific application areas.

[7]https://www.armbian.com/, last accessed 2025-04-04.

[8]At time of writing, an Alienware Area-51 Gaming Desktop starts at 4500€.

[9]https://docs.ceph.com/en/quincy/cephfs/index.html, last accessed 2025-03-19.

[10]https://github.com/minio/minio, last accessed 2025-03-19.

[11]https://www.mpi-forum.org/, last accessed 2025-03-19

A. *Parallelization*

While desktop computer systems have been found to potentially outperform SBC clusters, such as the 16-node Odroid C2 [12] cluster proposed in [11], the primary advantage of using such clusters lies in their ability to leverage multiple distributed processors rather than the available processing power, therefore focusing on runtime efficiency in a distributed manner. This characteristic allows students to gain hands-on experience with cluster computing by emulating the architecture of large-scale systems, thereby exposing them to challenges that a single operating system on a desktop computer might abstract away [11]. Even though the performance drawback compared to desktop systems has narrowed with the emergence of more powerful next-generation SBCs – opening up new applications for SBC clusters, such as in edge computing – their educational value remains a primary focus [15].

Building on the previously mentioned capabilities for distributed computation, the use of the MPI framework makes a significant contribution to the educational applications of such lab environments. By using MPI, workloads can be parallelized by distributing processes across multiple cluster nodes via a communication medium such as a network, thereby enhancing the level of parallelism [9]. This enables students to actively engage with fundamental aspects of distributed computing, including interprocess communication, synchronization, and workload distribution, which are central elements of the MPI standard [16]. These cluster systems offer a cost-effective platform that covers a wide range of high-performance computing (HPC) aspects, allowing students to explore the fundamental principles of HPC in a practical setting [12].

A major aspect of education concerning parallel computing involves providing insights into fundamental laws of parallel computing, such as Amdahl's Law and Gustafson's Law [14]. Essentially, these laws concern the limits of parallelization. Amdahl's Law emphasizes that the speedup of a computation is fundamentally constrained by its serial proportion, regardless of the number of processors. It highlights the challenge of achieving significant performance gains when only part of a program can be parallelized [17]. In contrast, Gustafson's Law opposes Amdahl's pessimistic view by arguing that practical workloads scale with the available computing power [18]. Gustafson [18] suggests that instead of assuming a fixed problem size, as more processors become available, larger problems can be tackled within the same execution time. In this context, the parallel or vector part of a program scales with the problem size, meaning that the amount of work that can be done in parallel increases with the number of processors. Gustafson's Law emphasizes that speedup should be therefore measured by scaling the problem to the number of processors, not by fixing the problem size.

Teaching these fundamental laws of parallel computing through conventional approaches can be supported using a

[12]https://wiki.odroid.com/odroid-c2/odroid-c2, last accessed 2025-03-19.

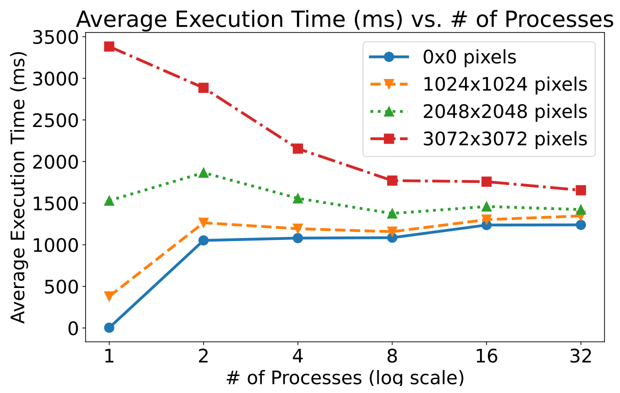

Figure 2. Observed execution times for Mandelbrot set calculation. Four different problem sizes (pixel resolution) were used to benchmark the parallelization. The Figure displays the average calculation times for each problem size as the number of available processes increases. For small problem sizes, the cost of the parallelization overhead exceeds the speedup of the calculation, thus the total calculation actually takes longer with more processes. At the same time, the calculation time converges for all problem sizes as there are some parts of a calculation that cannot be parallelized, which acts as a lower bound.

cluster lab environment, as it provides the necessary resources to study these principles through real-world applications. Such a lab environment can potentially offer a sufficient number of processing units to demonstrate the acceleration achieved through a distributed, parallelized approach, whereas conventional desktop processors are often limited to only a few cores. Furthermore, using conventional desktop systems might obscure the impact of critical factors such as communication between processing units, which plays a crucial role in parallel and distributed computing especially when it comes to message passing in a network-driven cluster environment. An example that demonstrates the application and limits of parallelization for students is calculating and visualizing the Mandelbrot set across varying problem sizes and process counts. By adjusting the process count and problem size — defined by the image resolution (width × height) — students can gain insights into the challenges and benefits of distributed parallelization.

An experiment was conducted in the proposed cluster lab environment, where the number of processes was varied as powers of two (with 1 process representing the serial version, and additional processes being used for parallel execution), ranging from 1, 2, 4, 8, 16, and 32, while the problem size was adjusted to 0, 1024, 2048, and 3072 pixels. The observed average execution times are shown in Fig. 2. As the results show, students were able to observe effects described by Amdahl's and Gustafson's laws regarding execution time through parallelization.

For small problem sizes (0 and 1024 pixels), not only was no performance improvement observed, but execution time significantly deteriorated as the number of processes increased. This effect can be attributed to the overhead introduced by parallelization when there is little to no computational workload, as the problem size must be large enough to justify internode communication [19].

At 2048 pixels, execution times initially worsened with the introduction of parallelization but then decreased significantly at higher process counts, falling noticeably below the serial execution time before stagnating. This marked the first problem size where parallelization provided a measurable speedup, while also illustrating Amdahl's law, as the speedup remained fundamentally limited by the sequential fraction of the workload. However, at 3072 pixels, performance improved consistently as the number of processors increased, aligning with Gustafson's law. The larger problem size allowed the parallelizable fraction of the workload to dominate, making parallel execution more efficient and mitigating the impact of communication overhead.

These valuable insights provide students with a deeper understanding of the fundamental aspects of HPC as defined by these laws. Moreover, they make such a lab environment a valuable supplement to traditional curricula, which often rely solely on textual or mathematical representations of these concepts.

B. Container Orchestration and Scalability

Besides their application in teaching HPC concepts, cluster lab environments may also serve as valuable educational platforms for exploring container orchestration technologies such as Kubernetes. With the rise of cloud computing, Kubernetes has become a widely adopted orchestration system and an industry standard for application deployment, playing a crucial role in managing containerized workloads in modern infrastructures [20]. To the best of the authors' knowledge, aspects of container orchestration are rarely considered as a potential teaching application in SBC-based cluster environments in literature. However, given the increasing significance of this technology in contemporary IT infrastructures, effectively integrating container orchestration concepts into laboratory education represents a valuable addition to lectures covering containerization and especially its orchestration.

While SBC clusters have been found to be inappropriate for parallel frameworks such as OpenStack, Apache Hadoop, or Spark — primarily due to limitations like insufficient per-node memory, high-latency network links, and SD card corruption [13] — new fields of application are emerging, particularly with the development of lightweight Kubernetes distributions. One such example is *K3s*,[13] a lightweight Kubernetes distribution specifically designed for resource-constrained environments.[14] With optimizations for both ARM64 and ARMv7 architectures, it provides a solution for running Kubernetes on SBC systems like Raspberry Pis. The availability of such distributions, which prioritize runtime efficiency on small-scale hardware, further enables the integration of container orchestration platforms into SBC cluster-based lab environments.

Although K3s focuses on minimizing resource consumption for small-scale hardware, it still retains key production-grade features, such as high availability. These capabilities further enhance its value in educational settings by enabling the demonstration of essential platform and operational aspects, such as fault tolerance and scalability, within a controlled laboratory environment. Therefore, in addition to addressing only application-related aspects of container orchestration, such as the design and deployment of microservice applications, students can engage in lab exercises that explore the underlying Kubernetes platform itself and its core mechanisms. These exercises are typically only feasible in real cluster environments, distinguishing them from single-node lab setups, such as Kubernetes installations on local desktop systems. While local setups for application-related lab experiences are not necessarily inferior in terms of performance —- since the aforementioned performance constraints inherent in SBC clusters also apply here —- SBC-based cluster environments provide the additional advantage of simulating multi-node setups that closely resemble real-world cluster environments. This enables SBC cluster-based lab environments to cover platform-related aspects in a practical manner, making them a valuable tool for educating students on critical topics that may be essential in their future professional careers.

To illustrate the practical applications of these concepts in a laboratory setting, several concrete examples can be introduced, demonstrating how SBC cluster-based environments can be used to teach key aspects of container orchestration and Kubernetes in an educational context. One concrete example from a student project that showcased the practical application of the proposed cluster was the analysis of the scalability of a self-developed cloud-native application. The project utilized the benchmarking tool *wrk*[15] to assess the application's performance under certain load conditions, approaching real-world workloads. By evaluating the response throughput across multiple replication levels of the individual service components, the experiment provided valuable insights into the application's scalability. This contributed to the overall understanding of how the application, proposed by the students, can handle dynamic workloads in a multi-node Kubernetes cluster environment.

Another example from a student project studies the impact of horizontal scaling on the performance of the Kubernetes control plane. The experiment aimed to analyze how increasing the number of control plane nodes affects the system's responsiveness and efficiency. Using the benchmarking framework *K-Bench*,[16] various control plane scalability scenarios were evaluated to measure performance indicators such as pod creation throughput, API response times, and CRUD operation latencies on Kubernetes resources. While this particular study was not conducted within the proposed cluster lab environment, such an analysis would be well-suited for it, as the previously mentioned distribution K3s supports high-availability cluster setups. This further highlights the relevance of the lab environment for examining platform aspects commonly

[13]https://k3s.io/, last accessed 2025-03-24.

[14]https://docs.k3s.io/, last accessed 2025-03-24.

[15]https://github.com/wg/wrk, last accessed 2025-04-10.

[16]https://github.com/vmware-tanzu/k-bench, last accessed 2025-04-10.

encountered in production environments.

Further educational lab activities could address key architectural and operational aspects of Kubernetes in multi-node environments, such as analyzing scheduling behavior (e.g., topology-aware vs. default), exploring service abstraction through the deployment of service resources, investigating the network model including the roles of kube-proxy and CNI plugins or examining fault tolerance and self-healing by simulating node failures to observe self-healing mechanisms and evaluate high-availability strategies.

IV. Experiences Made Constructing the Cluster

The educational capabilities of cluster-based environments are not only limited to scenarios where the cluster itself is used for teaching as described in Sec. III. As stated in literature [11], [13], the experience gained by building the cluster is a valuable benefit for students complementing subsequent operations on this cluster. The construction of such clusters often emphasizes relevant experiences, including hardware installation — such as racks, network components, and cabling — as well as software frameworks covering aspects like Linux, containers, MPI, or Kubernetes [13]. As mentioned earlier, the proposed DIY cluster was built as part of a student project, allowing for the derivation of insights from the experiences gained during its construction which are covered below.

A key experience gained from constructing the cluster was the emphasis on automation to minimize repetitive tasks, which is particularly useful in large-scale scenarios and enhances both scalability and expandability [13]. This project employs Ansible as a configuration management tool, which is well-suited for managing systems that are already running a base operating system [13]. Within this project, Ansible was primarily used to deploy the cluster environment by executing the necessary steps to configure the cluster nodes according to the desired cluster setup. This includes tasks such as network configuration, software installation, and SSH key management. Furthermore, Ansible is utilized for subsequent administrative tasks, including applying updates, implementing new requirements or fixing bugs.

The versatility of the cluster architecture in supporting diverse cluster setups to facilitate the proposed teaching applications (see Sec. III) emerged as a crucial aspect during its construction as well. Building on the fundamental aspect of automation, this cluster architecture enables the deployment of such diverse scenarios by leveraging a modular design of the underlying Ansible project. This modularity is achieved through establishing a framework based on Ansible's role-based approach, allowing the base cluster deployment to be extended with specific Ansible roles tailored to each cluster scenario (e.g. mapping MPI configuration through a dedicated MPI role). This approach not only ensures adaptability to changing teaching requirements – allowing for quick and automated transitions between MPI and Kubernetes cluster setups and vice versa – but also enhances the project's long-term extensibility by enabling the integration of new teaching applications and their associated concepts, thereby fostering continuous learning opportunities.

Another critical aspect, closely tied to the use of configuration management, is an architectural design keeping expandability and scalability in mind. By maintaining a scalable deployment approach, the proposed cluster architecture allows for dynamically adding or exchanging nodes, flexibly separating them into multiple distinct clusters, or aggregating all nodes to utilize the full extent of available resources. This flexibility is achieved through Ansible's grouping mechanism, which enables the organization of nodes into hierarchical inventory structures, allowing for adaptation to different deployment scalings. This flexibility is particularly useful for enhancing the cluster's capabilities by dynamically expanding the node count, simultaneously supporting different teaching applications — such as MPI and Kubernetes – or maximizing resource utilization.

Besides the previously discussed software-related experiences, hardware-related considerations also emerged. Although performance is not the primary focus of these cluster environments (see Sec. III), sufficient per-node performance is still essential to ensure proper cluster operation. Raw computing power is thereby relevant to a certain extent as the cluster must still meet the computational demands of its applications [11]. During the cluster's construction, it became evident that its architecture is highly dependent on the available hardware, requiring a certain level of both quality and quantity to be maintained. In this DIY project, these constraints were evident in the requirement for the SBC nodes to support the hardware requirements of the software stack associated with K3s,[17] which they successfully met. Furthermore, as mentioned in Sec. II, the SBCs network performance proved to be a another critical factor in their suitability.

Another significant limitation that arose was the restricted persistent storage, as the integrated eMMC storage of the Banana Pi devices was limited to 16GB. This constraint led to a rather pessimistic storage management approach, highlighting the need for further investigation into higher-capacity yet performant block storage solutions. Further investigations concerning the proposed DIY cluster could focus on storage integration via high-speed interfaces such as integrated USB 3.0 or PCIe, which is supported by newer SBC generations, such as the Raspberry Pi 5.[18] This becomes particularly evident as K3s performance is strongly dependent on its database performance. Therefore, it is recommended to use an external SSD instead of SD cards or eMMC storage.[19]

V. Conclusion

Constructing a computing cluster from a set of SBCs is not only feasible, it also proved to be a cost-efficient solution to supplement our current spectrum of teaching-learning-activities. Students can use the cluster to practice container

[17]https://docs.k3s.io/installation/requirements, last accessed 2025-04-06

[18]https://www.raspberrypi.com/products/raspberry-pi-5, last accessed 2025-06-02

[19]https://docs.k3s.io/installation/requirements, last accessed 2025-04-06

management using Kubernetes, the automation of networked jobs through Ansible, and the pros and cons of parallelization, e.g., with respect to Amdahl's Law and Gustafson's Law as well as handling parallelization tasks with OpenMPI. Our cluster offers two banks of ten Banana Pi M5 SBCs each, for a total of 40 processing cores per bank or 80 cores in total, with two currently empty slots per bank remaining. Each bank can be used standalone, or both can be utilized together.

Each Banana Pi is powerful enough to host several containers simultaneously. Though the on-board Mali-G31 GPU is in itself quite weak, as it was designed specifically for mobile devices, the fact that each node in the cluster features its own graphics unit opens the way for possible laboratory applications for example in machine learning or artificial intelligence, especially with regards to federated learning – however, as we didn't employ the cluster in ML teaching as of yet, we don't have any applicable teaching scenarios available. As each node is a standalone SBC, it is also easy to exchange a node in case of failure, as simply a singular SBC needs to be fitted with an appropriate software image and plugged into the free cluster slot in place of the faulty one. This coupled with automated node deployment allows for an easily maintainable learning environment that can facilitate several teaching-learning-scenarios with minimal overhead.

One limitation inherent to this design lies with performance. While this cluster lends itself well to teaching-learning-scenarios for parallel computing and container orchestration, it severely lacks in processing power compared to commercial clusters and thus is not well suited for actual real-world scenarios (e.g., climate simulation). One expansion that somewhat addresses this limitation is the (gradual) exchange of nodes with the more modern Raspberry Pi 5 (other other comparably new) SBCs that feature more processing power. A similar limitation with bandwidth and storage capacity also pertains to Big Data projects: SBCs are not suitable to handle Big Data, and not even clustering them can mitigate this.

References

[1] I. Aubel *et al.*, "Adaptable Digital Labs - Motivation and Vision of the CrossLab Project," in *2022 IEEE German Education Conference (GeCon)*. Berlin, Germany: IEEE, 2022, pp. 1–6.

[2] JetBrains s.r.o., "The State of Developer Ecosystem 2023," Available: https://www.jetbrains.com/lp/devecosystem-2023/, last accessed 2025-03-12, JetBrains s.r.o., Tech. Rep., 2023.

[3] A. T. Stull, M. J. Gainer, and M. Hegarty, "Learning by enacting: The role of embodiment in chemistry education," *Learning and Instruction*, vol. 55, p. 80–92, jun 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0959475217302311

[4] D. May, C. Terkowsky, V. Varney, and D. Boehringer, "Between hands-on experiments and cross reality learning environments – contemporary educational approaches in instructional laboratories," *European Journal of Engineering Education*, vol. 48, no. 5, pp. 783–801, 2023. [Online]. Available: https://doi.org/10.1080/03043797.2023.2248819

[5] A. Hofstein and V. N. Lunetta, "The role of the laboratory in science teaching: Neglected aspects of research," *Review of Educational Research*, vol. 52, no. 2, p. 201–217, jun 1982. [Online]. Available: https://journals.sagepub.com/doi/10.3102/00346543052002201

[6] C. Chhetri, ""it was a one of a kind experience:" student experiences and pedagogical design of a project-based hands-on cybersecurity pen-testing course," in *The 24th Annual Conference on Information Technology Education*. Marietta GA USA: ACM, oct 2023, p. 22–27. [Online]. Available: https://dl.acm.org/doi/10.1145/3585059.3611402

[7] D. Aurich, C. Lehr, M. M. Wolny, K. Boettcher, and A. Brümmer, *Immersive Airfoil Student Laboratory: Augmented Reality Application with Real-Time Measurement Data Access*. Cham: Springer Nature Switzerland, 2024, vol. 1135, p. 283–299. [Online]. Available: https://link.springer.com/10.1007/978-3-031-70771-1%5F14

[8] J. Garcia and L. D. Schultz, "Determination of sulfate by conductometric titration: An undergraduate laboratory experiment," *Journal of Chemical Education*, vol. 93, no. 5, p. 910–914, may 2016. [Online]. Available: https://pubs.acs.org/doi/10.1021/acs.jchemed.5b00941

[9] T. Klinger and C. Madritsch, "Parallel computing for education using single-board computers," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Opatija: IEEE, may 2018, p. 0616–0619. [Online]. Available: https://ieeexplore.ieee.org/document/8400116/

[10] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, "Deploying microservice based applications with kubernetes: Experiments and lessons learned," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. San Francisco, CA, USA: IEEE, jul 2018, p. 970–973. [Online]. Available: https://ieeexplore.ieee.org/document/8457916/

[11] P. J. Basford *et al.*, "Performance analysis of single board computer clusters," *Future Generation Computer Systems*, vol. 102, pp. 278–291, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X1833142X

[12] A. M. Pfalzgraf and J. A. Driscoll, "A low-cost computer cluster for high-performance computing education," in *IEEE International Conference on Electro/Information Technology*, 2014, pp. 362–366.

[13] S. J. Johnston *et al.*, "Commodity single board computer clusters and their applications," *Future Generation Computer Systems*, vol. 89, pp. 201–212, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18301833

[14] C. Baun, H.-N. Cocos, and R.-M. Spanou, "Erfahrungen beim Aufbau von großen Clustern aus Einplatinencomputern für Forschung und Lehre [Experiences with the Assembly of large Clusters of Single Board Computers for Research and Teaching]," *Informatik-Spektrum*, vol. 41, no. 3, p. 189–199, jun 2018. [Online]. Available: http://link.springer.com/10.1007/s00287-017-1083-9

[15] Z. Krpić, L. Loina, and T. Galba, "Evaluating performance of sbc clusters for hpc workloads," in *2022 International Conference on Smart Systems and Technologies (SST)*, 2022, pp. 173–178.

[16] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard Version 4.1*, Nov. 2023. [Online]. Available: https://www.mpi-forum.org/docs/mpi-4.1/mpi41-report.pdf

[17] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, ser. AFIPS '67 (Spring). New York, NY, USA: Association for Computing Machinery, 1967, p. 483–485. [Online]. Available: https://doi.org/10.1145/1465482.1465560

[18] J. L. Gustafson, "Reevaluating amdahl's law," *Commun. ACM*, vol. 31, no. 5, p. 532–533, may 1988. [Online]. Available: https://doi.org/10.1145/42411.42415

[19] D. Culler *et al.*, "Logp: towards a realistic model of parallel computation," *SIGPLAN Not.*, vol. 28, no. 7, p. 1–12, jul 1993. [Online]. Available: https://doi.org/10.1145/173284.155333

[20] Sameer, S. De, and R. Prashant Singh, "Selective analogy of mechanisms and tools in kubernetes lifecycle for disaster recovery," in *2022 IEEE 2nd International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, 2022, pp. 1–6.

[21] B. P. Abbott *et al.*, "Einstein@home search for periodic gravitational waves in early s5 ligo data," *Phys. Rev. D*, vol. 80, p. 042003, Aug 2009. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevD.80.042003

[22] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "Seti@home: an experiment in public-resource computing," *Commun. ACM*, vol. 45, no. 11, p. 56–61, nov 2002. [Online]. Available: https://doi.org/10.1145/581571.581573

[23] E. Gamess and M. Parajuli, "Performance evaluation of the docker technology on different raspberry pi models," in *Proceedings of the 2023 5th International Electronics Communication Conference*. Osaka City Japan: ACM, jul 2023, p. 27–37. [Online]. Available: https://dl.acm.org/doi/10.1145/3616480.3616485

[24] M. Soll, "What exactly is a laboratory in computer science?" in *2023 IEEE Global Engineering Education Conference (EDUCON)*, 2023, pp. 1–9.