Article

Not peer-reviewed version

Evaluating ARM and RISC-V Architectures For High-Performance Computing With Docker and Kubernetes

Vedran Dakić , Leo Mršić * , Zdravko Kunić , Goran Đambić

Posted Date: 13 August 2024

doi: 10.20944/preprints202408.0867.v1

Keywords: HPC; Docker; containers; performance; evaluation; heterogenous computing

Preprints.org is a free multidiscipline platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Evaluating ARM and RISC-V Architectures For High-Performance Computing With Docker and Kubernetes

Vedran Dakić [1], Leo Mršić [2,*], Zdravko Kunić [2] and Goran Đambić [3]

[1] Department of Cybersecurity and System Engineering, Algebra University, 10000 Zagreb, Croatia
[2] Department of Information Systems and Business Analytics, Algebra University, 10000 Zagreb, Croatia
[3] Department of Software Engineering, Algebra University, 10000 Zagreb, Croatia
* Correspondence: leo.mrsic@algebra.hr (L.M.)

Abstract: This paper thoroughly assesses the ARM and RISC-V architectures in the context of High-Performance Computing (HPC). It includes an analysis of Docker, Kubernetes, and KVM virtualization integration. Our study aims to evaluate and compare these systems' performance, scalability, and practicality in a general context and then assess the impact that might have on special use cases, like HPC. ARM-based systems exhibited better performance and seamless integration with Docker and Kubernetes, underscoring their advanced development and effectiveness in managing high-performance computing workloads. On the other hand, despite their open-source architecture, RISC-V platforms presented considerable intricacy and difficulties in coordinating with Kubernetes, which hurt their overall effectiveness and ease of management. The results of our study offer valuable insights into the practical consequences of implementing these architectures for HPC, highlighting ARM's preparedness and the potential of RISC-V while acknowledging the increased complexity involved and significant trade-offs at this point.

Keywords: HPC; Docker; containers; performance; evaluation; heterogenous computing

1. Introduction

HPC has played a leading role in pushing technological improvements, particularly in scientific research, weather forecasting, financial modeling, and other sectors that rely on extensive computational capabilities. Throughout history, the x86 architecture, introduced by Intel and AMD, has been the prevailing force in the field of HPC. The advancement of this design, characterized by ongoing enhancements in computational capability, parallel processing, and energy efficiency, has facilitated the creation of some of the most formidable supercomputers globally. The Summit supercomputer at Oak Ridge National Laboratory and the Frontera at Texas Advanced Computing Center illustrate the exceptional performance of x86 architecture in HPC.

Nevertheless, the increasing need for computational capacity, along with the necessity for energy-saving solutions, have stimulated interest in alternate architectures. ARM, initially developed for energy-efficient use in mobile devices, has gained attention in HPC due to its energy efficiency and expanding processing capabilities. ARM debuted in the HPC field by introducing the Fujitsu A64FX processor in the Fugaku supercomputer. This processor has achieved remarkable success in terms of both performance and power efficiency, leading it to secure the top position on the TOP500 list of the world's most powerful supercomputers.

In addition to ARM's increasing popularity, RISC-V, an open-source instruction set architecture (ISA), has emerged as a highly attractive option for HPC applications. RISC-V was created for academic and research reasons. It provides the advantage of customization and innovation, enabling developers to adapt the architecture to meet specific HPC requirements. The HPC community has shown great interest in the possibilities for customization and the advantages of an open-source paradigm. Nevertheless, the intricate nature of incorporating RISC-V into contemporary software

© 2024 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

ecosystems, including containerization technologies such as Docker and orchestration tools like Kubernetes, poses distinct difficulties.

The introduction of containerization technologies like Docker and Kubernetes is fundamentally transforming the management and deployment of HPC workloads. These technologies provide the capacity to quickly move and adapt applications to different computer environments while improving efficiency. By integrating these technologies with ARM and RISC-V platforms, HPC can achieve higher levels of performance and efficiency. However, combining these systems has challenges, mainly due to the variances in architecture and varied levels of software support.

Bruhn et al. (2015) present a low-power, fault-tolerant heterogeneous computer featuring multi-core CPUs, GPUs, and FPGAs for small satellite applications. The system's x86 CPU allows for extensive software compatibility in space environments. Integrating GPU and FPGA enhances computational performance, achieving TFLOP-level processing power. Challenges include ensuring radiation tolerance and safety-critical operations, previously unaddressed for x86 processors in space. Future research could focus on further improving radiation tolerance and developing more robust software ecosystems for space applications [1].

Reichenbach et al. (2018) explore the integration of FPGAs into Heterogeneous System Architecture (HSA) compliant systems, addressing the challenge of interfacing FPGAs with CPUs. The LibHSA IP library simplifies this integration, allowing FPGA accelerators to utilize high-level language toolchains. Demonstrations with image processors show significant performance improvements. Problems include the lack of FPGA models in HSA standards and the complexities of interfacing. Future work could aim to standardize FPGA models in HSA and further simplify FPGA-CPU integration [2].

Feng et al. (2016) introduce HeteroSim, a simulator for heterogeneous CPU-FPGA systems. It supports architectural exploration by simulating x86 multi-cores connected to FPGAs, allowing for performance analysis and optimization of memory hierarchies. Challenges include the absence of integrated simulators for system-level architectural exploration. Future research may enhance simulation accuracy and extend the tool to support a broader range of architectures and applications [3].

Chang et al. (2017) survey integrated heterogeneous systems and collaborative computing techniques, focusing on CPU-GPU and CPU-FPGA systems. The paper evaluates OpenCL's effectiveness for programming these systems, highlighting limitations and suggesting future programming languages. Problems include inadequate support for memory coherence and shared virtual memory in current programming interfaces. Future research could improve programming models to enhance collaboration between heterogeneous devices [4].

Mittal and Vetter (2015) review CPU-GPU heterogeneous computing techniques, discussing workload partitioning and the design of CPU-GPU systems. They address performance and energy efficiency challenges and review benchmark suites for evaluating such systems. Critical problems include the complexity of optimizing workload distribution and the need for better performance models. Future research may focus on developing more effective optimization techniques and comprehensive benchmark suites [5].

Prongnuch and Wiangtong (2016) propose a performance evaluation of APIs and partially reconfigurable hardware accelerators on heterogeneous computing platforms. The authors use the Parallella single-board computer to compare matrix-vector multiplication performance across different accelerators. They find PR hardware accelerators the most efficient for increasing data processing. Challenges include optimizing API performance and managing hardware reconfiguration. Future work could improve API efficiency and explore new applications for PR hardware accelerators [6].

Rethinagiri et al. (2015) present platforms combining CPUs, GPUs, and FPGAs for high-performance and embedded systems. These platforms achieve significant speed-ups and energy savings compared to dual-device systems. Challenges include managing data transfers and maximizing parallelism. Future research could optimize workload distribution and explore new high-performance applications for trigeneous platforms [7].

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

Kurth et al. (2017) introduce HERO (The Open Heterogenous Research Platform), a research platform combining RISC-V cores on FPGAs with ARM host processors. HERO includes a comprehensive software stack supporting OpenMP and enabling rapid exploration of heterogeneous systems. Key challenges involve integrating and scaling PMCA (Programmable ManyCore Accelerators) architectures. Future research could expand HERO's capabilities and apply it to more diverse application domains [8].

Parnassos et al. (2017) propose a programming model that supports approximation semantics in heterogeneous architectures comprising CPUs, GPUs, and FPGAs. The model allows for dynamic performance-quality trade-offs. Challenges include ensuring seamless execution and effective approximation control. Future research may refine the approximation model and expand its application to other domains [9].

Lopez-Novoa et al. (2015) review performance tools for heterogeneous systems, focusing on GPUs and Intel's Xeon Phi. They describe development frameworks and performance models for accelerator-based systems. Key issues include the need for more accurate simulators and comprehensive performance tools. Future research could enhance performance modeling techniques and develop more robust simulation tools for heterogeneous computing [10].

A notable study introduces a synergistic computing framework built on the Halide programming model, designed to enhance performance in heterogeneous systems by leveraging both CPUs and GPUs. The framework addresses critical issues like data coherence, workload partitioning, and job dispatching. Despite significant performance gains, the complexity of writing cooperative Halide programs remains challenging, necessitating further simplifications and optimizations in future research [11]. FPGA implementations have shown significant potential in portable and energy-efficient applications. An innovative solution for portable DNA sequencing using a base calling hardware architecture based on RISC-V demonstrated a 1.95x energy efficiency improvement over x86 and 38% over ARM. Despite these advancements, further miniaturization and computational load reduction are required for broader adoption [12].

Another study explores the integration of novel computing architectures, including ARM and RISC-V CPUs and FPGAs, into HPC education at Georgia Tech. The collaborative workflow highlighted the benefits and challenges of integrating these architectures into educational curricula. Key sticking points included the complexity of managing diverse hardware and software environments. Future research should develop more streamlined tools and methodologies to facilitate the integration of heterogeneous computing architectures in educational settings [13]. An FPGA-based research platform, HEROv2, combines RISC-V cores with ARMv8 or RV64 host processors, enabling significant application speedups. HEROv2's complexity suggests a need for further simplifications in hardware-software integration. Future research could enhance the platform's compiler capabilities and expand its application range [14].

PCS, a productive computational science platform, aims to unify multiple programming models for cluster-scale heterogeneous computing, emphasizing FPGA acceleration for graph-centric workloads. The platform presents significant advancements but faces challenges in optimizing FPGA designs for specific applications and ensuring compatibility with diverse hardware. Future research should focus on developing more adaptable FPGA models and enhancing integration [15]. Combining multiple CPU and CGRA cores is explored, presenting implementation results for digital audio and machine learning applications. Despite the promising results, the absence of commercial CGRAs poses challenges regarding widespread adoption. Future research should aim at improving performance metrics and exploring commercial applications of CGRA-based architectures to validate their practical viability [16].

Heterogeneous computing utilizing FPGAs has been demonstrated to significantly reduce the effort required to integrate FPGAs into existing systems. The IP library, LibHSA, simplifies FPGA integration into HSA-compliant systems, enabling user-space memory access and low-latency task dispatch. Challenges remain in optimizing FPGA designs for specific applications and ensuring compatibility with diverse hardware [17]. The study on accelerating elliptic curve cryptography on NVIDIA Tegra X2 embedded GPU platforms achieved significant speedups in cryptographic

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

operations, outperforming ARM CPUs and FPGA implementations in terms of power efficiency. The reliance on specific hardware platforms limits the generalizability of the results, suggesting future research should explore broader hardware compatibility and further optimization techniques [18].

Heterogeneous computing is crucial in AI and Big Data applications within high-energy physics. Integrating GPUs, FPGAs, and other specialized processors presents challenges in performance optimization and energy efficiency. Future research should focus on developing unified frameworks and tools to manage heterogeneous systems effectively, ensuring seamless integration and optimal resource utilization across various computational tasks [19]. The FLIA architecture abstracts heterogeneous computing for mobile GPUs and FPGAs, supporting task partition, communication, and synchronization. Despite impressive performance gains, challenges include the complexity of developing applications for heterogeneous architectures and managing inter-processor communication. Future research should simplify the development process and enhance the proposed architecture’s scalability for broader applications [20].

Another significant contribution is the development of Molecule, a serverless computing system utilizing heterogeneous computers, including general-purpose devices like Nvidia DPUs and domain-specific accelerators such as FPGAs and GPUs. By leveraging heterogeneous hardware, Molecule significantly improves function density and application performance. Key innovations include the XPU-Shim for multi-OS systems and vectorized sandbox for hardware abstraction. Despite the performance improvements, challenges remain in optimizing startup and communication latency. Future research should refine these optimizations and expand the system’s compatibility with additional hardware types [21].

Energy efficiency is another critical focus area. A Xilinx Zynq MPSoC device approach demonstrates significant energy savings and performance improvements by simultaneously executing tasks and applying adaptive voltage scaling. Integrating GPUs and FPGAs as specialized hardware units showcases the potential of heterogeneous computing for energy-efficient applications. However, optimizing the balance between performance and energy consumption across different tasks remains challenging. Future research should enhance the adaptability of voltage scaling techniques and explore additional use cases for reconfigurable MPSoCs [22].

Heterogeneous computing also plays a crucial role in deep learning. Combining FPGAs and GPUs for accelerating deep neural networks in embedded systems demonstrates that direct hardware mapping of CNNs on FPGAs outperforms GPU implementations regarding energy efficiency and execution time. However, the resource-intensive nature of FPGA-based implementations necessitates a hybrid approach. The study shows that heterogeneous FPGA-GPU acceleration reduces energy consumption and latency compared to GPU-only solutions. Challenges include managing communication overheads and optimizing the division of tasks between FPGAs and GPUs. Future research should focus on refining hybrid acceleration techniques and expanding the range of supported neural network architectures [23].

Security concerns in heterogeneous systems are also highlighted. A survey on electrical-level attacks on CPUs, FPGAs, and GPUs discusses how vulnerabilities in individual components can affect the overall security of heterogeneous architectures. Ensuring the security of multitenant environments and developing robust defense mechanisms are vital challenges. Future research should investigate the potential for cross-component attacks in heterogeneous systems and develop comprehensive security frameworks to mitigate these risks [24].

The demand for computing power and the diversity of computational problems have led to exploring the non-uniform decomposition of data domains to improve fluid flow simulation performance on hybrid architectures. Evaluations of systems combining x86 CPUs with GPUs and ARM CPUs with FPGAs show performance improvements of up to 15.15% with non-uniform partitioning. Despite the benefits, challenges include managing the complexity of data decomposition and optimizing the collaboration between different hardware components. Future research should refine data partitioning techniques and expand the applicability of collaborative execution methods to other computational problems [25].

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

Scheduling frameworks for heterogeneous computing platforms combining CPU and FPGA resources effectively increase performance and reduce energy consumption. Extending a high-level C++ template-based scheduler to the Xeon+FPGA platform demonstrates performance improvements of up to 8x compared to CPU-only solutions. Key challenges include optimizing the scheduler for dynamic and adaptive task allocation. Future research should enhance the framework's adaptability to different application domains and explore additional HPC benchmarks to validate its effectiveness [26]. The study on accelerating coupled-cluster calculations with GPUs presents an algorithm adapted for heterogeneous computing platforms with multicore CPUs and GPUs. The authors demonstrate significant speedups on pre-exascale and exascale supercomputers. Challenges include managing the limited memory space of GPUs and minimizing CPU-GPU data transfers. Future research should optimize data tiling strategies and expand the algorithm's applicability to other computational chemistry problems [27].

Another study reviews discussions from critical workshops on heterogeneous computing, highlighting the integration of multicore CPUs with GPUs, FPGAs, and Intel Xeon Phis in HPC clusters. The workshops emphasized challenges such as maximizing efficiency, resource utilization, and energy optimization. Future research should aim to develop specialized programming environments and tools to address these challenges and foster innovation in heterogeneous computing [28]. The development of Xar-Trek, a compiler and runtime framework, allows execution migration between heterogeneous ISA CPUs and FPGAs at run-time. By compiling applications for multiple CPU ISAs and FPGA acceleration, Xar-Trek enables dynamic and transparent migration. Challenges include optimizing scheduling policies and managing diverse hardware configurations. Future research should focus on refining scheduling heuristics and expanding the framework's support to additional hardware architectures for broader applicability [29].

A historical perspective on heterogeneous computing highlights the complexity of integrating diverse processors like RISC, GPUs, TPUs, and FPGAs. It emphasizes the need for advanced compilers to map programming languages to various hardware platforms while ensuring optimal performance. Key challenges include maintaining compatibility and efficiency across heterogeneous systems. Future research should focus on developing robust compiler technologies and enhancing the interoperability of different processing units to exploit the potential of heterogeneous computing [30] entirely.

Advancements in heterogeneous computing have revolutionized HPC platforms, with significant developments in both hardware and software. In their paper, Wyrzykowski and Ciorba (2022) present algorithmic and software development advancements, emphasizing the need for novel programming environments to harness the potential of heterogeneous systems. They note the increasing adoption of GPU accelerators and highlight the challenges in achieving performance portability and energy efficiency across diverse architectures. Future research could explore further integrating new accelerator technologies and improving programming models to enhance efficiency and performance [31].

In their study, Hagleitner et al. (2021) discuss how heterogeneous computing systems are crucial for complex scientific discovery workflows. They highlight the transition from homogeneous to heterogeneous architectures as a response to the limits of Moore's law, with a focus on sustainable computing through domain-specific hardware. However, the increased complexity in system design and the need for better integration of multidisciplinary workflows are significant challenges. Potential future research could aim to optimize these systems' design to balance performance and energy efficiency [32].

Mavrogeorgis (2021) addresses the challenges of heterogeneous migration between x86 and ARM machines, focusing on techniques to simplify the migration process. The research highlights the significant overhead induced by transforming the execution state during migration, which offsets the benefits. The goal is to create a uniform address space to simplify migration, but performance and energy efficiency remain problematic. Future research could explore more efficient methods of state transformation and address space unification [33].

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

Thomadakis and Chrisochoides (2022) introduce a runtime framework for performance portable programming on distributed heterogeneous systems. Their work demonstrates substantial performance improvements and scalability by hiding hardware idiosyncrasies and optimizing resource utilization. Despite these advancements, programming such architectures remains challenging due to their increased complexity. Future research could focus on further reducing this complexity and enhancing the portability of performance across diverse systems [34].

Nikov et al. (2020) present a methodology for simultaneous heterogeneous computing using a quad-core ARM Cortex-A53 CPU and FPGA accelerator. They achieve significant performance improvements through a heterogeneous scheduler that optimally distributes tasks. However, integrating diverse computing units introduces task scheduling and synchronization complexity. Future research could aim to develop more sophisticated scheduling algorithms to enhance performance and energy efficiency further [35].

In their paper, Fuentes et al. (2022) discuss the introduction of heterogeneous computing in undergraduate education using DPC++. They highlight the importance of teaching modern computing architectures, including GPUs and FPGAs, to bridge the gap between academic knowledge and industry requirements. The challenge lies in simplifying complex hardware concepts for students. Future research could explore more effective teaching methods and tools to improve student understanding and engagement in heterogeneous computing [36].

Thomadakis and Chrisochoides (2023) present a runtime support framework for performance portability on heterogeneous distributed platforms. Their framework significantly improves performance and scalability, particularly in distributed memory environments. However, achieving efficient inter-node communication among diverse devices remains a challenge. Future research could further focus on optimizing communication protocols and exploring new abstractions to enhance performance portability [37].

Kavanagh et al. (2020) explore energy-aware self-adaptation for applications on heterogeneous parallel architectures. They develop a framework that automates application configuration and deployment to improve energy efficiency. Despite these advancements, managing the complexities of heterogeneous devices and achieving consistent energy savings remains challenging. Future research could investigate more adaptive algorithms and techniques to optimize energy usage dynamically [38].

Yu et al. (2022) developed a methodology to characterize uncertainties in Earth system modeling with heterogeneous many-core architectures. They address the challenges of non-bit-for-bit reproducibility and numerical perturbations in simulations, which can blend with coding errors. Their methodology provides a way to distinguish platform-induced perturbations from software bugs, enhancing model reliability. Future research could focus on refining and applying this methodology to a broader range of applications to ensure robust modeling on new architectures [39].

Cheng et al. (2022) describe constructing and applying a large-scale ARM computing cluster for high-energy physics experiments. They highlight the cost-effectiveness of ARM processors compared to traditional x86 processors. However, they also note the performance limitations of ARM processors, especially in memory-bound operations. Future research could explore optimizing ARM-based clusters for specific scientific applications and reducing reliance on single-chip architectures to enhance performance [40].

In their study, Kamaleldin and Göhringer propose AGILER, a tile-based many-core architecture for RISC-V processors designed for adaptability and modularity in heterogeneous computing environments. The architecture supports 32-bit and 64-bit RISC-V ISAs with scalable network-on-chip communication, enabling high system scalability. Evaluations on Xilinx FPGAs showed scalable performance up to 685 MOPS for 32-bit tiles and 316 MOPS for 64-bit tiles, with reconfiguration times of 38.1 ms per tile. The key challenges include managing the complexity of modularity and ensuring efficient reconfiguration. Future research could focus on optimizing the reconfiguration manager and expanding the architecture's application domains [41].

In their survey, Nicholas, Gui, and Saqib analyze SoC platform security across ARM, Intel, and RISC-V architectures, highlighting RISC-V's potential for customizable security extensions. They

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

discuss various hardware and software security attacks and compare RISC-V's capabilities to traditional architectures. The tradeoff between performance and security remains a significant challenge. They suggest further research into enhancing RISC-V's security features without compromising performance, mainly focusing on developing robust security frameworks and exploring new attack vectors to improve defense mechanisms [42].

Wang and colleagues introduce xBGAS, an extension to the RISC-V ISA aimed at enhancing HPC by enabling direct access to remote shared memory. This reduces inter-process communication overhead by 69.26% and achieves an average 21.96% performance gain. Challenges include ensuring seamless integration with existing software infrastructures and maintaining low latency in data access. Future research might explore further optimization of the xBGAS design and its application in various HPC scenarios, potentially expanding its adoption in large-scale distributed systems [43].

Tornero and his team describe an open-source FPGA platform for exploring heterogeneous many-core architectures, focusing on integrating custom accelerators with standard RISC-V cores. The platform's coherent shared memory model enhances programmability and communication efficiency. Preliminary results indicate significant benefits from using systolic accelerators. The main challenges are optimizing the network and memory subsystems and managing resource usage. Future work could involve refining the platform to support more complex accelerators and improving the coherence mechanisms to enhance overall system performance [44].

Gómez-Sánchez et al. examine using RISC-V in genomics-based workloads, benchmarking the Variant-Interaction Analytics use case. They highlight the potential of RISC-V for HPC in large-scale scientific environments. The study reveals challenges in achieving comparable performance to x86 architectures, particularly in data processing and system integration. Future research should optimize RISC-V implementations for specific scientific applications, improve performance and scalability, and address integration issues to facilitate broader adoption in genomics and other data-intensive fields [45].

Stoyanov, Kakanakov, and Marinova developed a secure heterogeneous RISC-V system featuring a protection-dedicated core for establishing root-of-trust and monitoring execution. This architecture enhances system security by providing hardware mechanisms for control and monitoring. The primary challenge lies in integrating these security features without significantly impacting performance. Future research could explore advanced security protocols and mechanisms to strengthen system integrity further and optimize the balance between security and performance in heterogeneous RISC-V systems [46].

Gonzalez and colleagues present a heterogeneous RISC-V SoC, integrating high-performance out-of-order cores, energy-efficient in-order cores, and specialized accelerators in a low-power 22nm FinFET process. The SoC achieves substantial performance and efficiency gains, with up to 286x MOPS/W improvement. Challenges include managing the complexity of integrating diverse components and optimizing power consumption. Future research could focus on refining the integration process, exploring new accelerator designs, and reducing power consumption while maintaining high performance across varied workloads [47].

Kamaleldin, Hesham, and Göhringer propose a modular RISC-V-based many-core architecture for FPGA accelerators designed for flexibility and scalability. The architecture features multiple processing clusters connected via a network-on-chip, supporting dynamic and partial reconfiguration. Evaluations demonstrate scalable performance and memory bandwidth. Key challenges include managing reconfiguration complexity and ensuring efficient intra-cluster communication. Future research might explore enhancing the flexibility of reconfiguration processes and expanding the architecture's applicability to a broader range of applications, improving performance and energy efficiency [48].

Jia and colleagues explore a programmable heterogeneous microprocessor based on bit-scalable in-memory computing (IMC), addressing energy and throughput trade-offs in accessing data. The architecture integrates a 590-Kb IMC accelerator, digital near-memory computing (NMC) accelerator, and RISC-V CPU. It achieves high energy efficiency and performance in deep neural network tasks. Challenges involve maintaining computation signal-to-noise ratio (SNR) and ensuring robust

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

hardware and software integration. Future research could improve IMC and NMC integration, enhance SNR, and extend the architecture's application to more complex and varied workloads [49].

Docker and Kubernetes have been thoroughly covered in one of our previous papers [50]. The only notable addition to this technology overview is that Kubernetes and Docker are still complicated to work with on the RISC-V platform. We will cover this topic later in this paper.

This paper is organized as follows: in the next section, we'll go through some basics of ARM and RISC-V platforms and the installation process for Docker and Kubernetes. After those two sections, we'll discuss our experimental test and setup environment, followed by the performance evaluations, discussion about performance and feasibility of being used by HPC environments, future works, and conclusion.

2. ARM as a Platform for Docker and Kubernetes

ARM processors are increasingly used to deploy Docker and Kubernetes on Ubuntu because of their energy efficiency, scalability, and cost-effectiveness. This is particularly advantageous in cloud computing and edge contexts. ARM's RISC architecture is highly efficient at processing high-throughput workloads while consuming less power. This makes it an excellent option for running containerized apps using Docker and orchestrating them with Kubernetes. These benefits are especially noticeable when energy economy and cost-effectiveness are crucial, such as in extensive cloud data centers and dispersed edge computing configurations.

Utilizing ARM processors with Docker on Ubuntu enables developers to generate compact and adaptable containers capable of operating on many platforms, hence offering versatility in deploying applications. The ARM architecture is compatible with multiple Linux distributions, such as Ubuntu, making it a flexible choice for developers who want to utilize containerization technologies. Docker is highly efficient on ARM processors because of their capacity to manage concurrent operations with reduced energy requirements compared to standard x86 processors. Docker packages apps and their dependencies into containers. Efficiency is paramount when implementing services that must be scaled over several nodes, as shown in extensive cloud infrastructures or distributed networks [51].

Kubernetes boosts the functionality of ARM processors by effectively managing and orchestrating Docker containers in a scalable manner. It enables the automatic deployment, scaling, and management of application containers across groups of hosts, offering a framework that guarantees the reliability and resilience of applications. Integrating ARM processors and Kubernetes on Ubuntu provides a robust solution for delivering microservices and other cloud-native applications necessitating comprehensive orchestration. Kubernetes' capacity to scale and oversee containers over a wide range of nodes, including those utilizing ARM processors, guarantees effective deployment and management of applications, even in different environments [52,53].

Furthermore, researchers have conducted several experiments to investigate the integration of Kubernetes with ARM processors to enhance performance and optimize resource consumption. An example is research conducted on the KubCG platform, which showcased the effectiveness of a dynamic Kubernetes scheduler in enhancing container deployment in clusters with diverse architectures, such as those including ARM processors. The utilization of ARM processors in managing containerized workloads using Kubernetes has demonstrated a notable decrease in job completion time, highlighting the potential for enhanced efficiency. A different research study emphasized the utilization of ARM-based fog computing platforms that employ Docker and Kubernetes for effective data processing at the network edge, further confirming the appropriateness of ARM processors in situations that need both scalability and low latency [54,55].

The combination of ARM processors, Docker, and Kubernetes is seen in the implementation of distributed file systems, which are crucial for efficiently handling data over extensive clusters. Studies have demonstrated that deploying distributed file systems such as CephFS and Lustre-ZFS on ARM-based Kubernetes clusters can enhance the flexibility of data management and the dependability of services. This is especially advantageous in contemporary data centers and cloud environments requiring fast data transfer rates and reliable operations [56].

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

Using Docker and Kubernetes on Ubuntu operating on ARM processors offers a resilient and effective solution for contemporary cloud computing and edge scenarios. The combination utilizes ARM's energy-efficient and scalable technology, Docker's containerization capabilities, and Kubernetes' powerful orchestration to provide high-performance, cost-effective, and scalable solutions for various applications. This is why cloud providers are partially switching to ARM-based platforms for Kubernetes environments in their offering, as they offer excellent performance for the vast majority of everyday applications while being more efficient than x86 platforms.

Deployment processes on the TuringPi2 platform were quite complex – that's down to our hardware choice due to the lack of availability of Ampere-based servers, for example. But we had to:

- Flash the image to the module using the TuringPi web interface (for RK1) or the Seeed Studio Development Board Kit connected to an Ubuntu-based laptop with a specific Ubuntu release and NVIDIA's SDK Manager;
- Power on the module to enter the default installation;
- Configure output on TuringPi to output from the module via HDMI.

TuringPi has HDMI output available, so we could use GUI if required. This is much more convenient than using the USB serial console, which is easy to break physically on our RISC-V platform.

2.1. Docker Deployment

The situation with Docker deployment on the ARM platform is straightforward. A set of Docker packages is available in repositories for Ubuntu for ARM. Hence, the installation process for Docker is as simple as issuing the following apt command on the Ubuntu Server:

```
apt -y install docker.io
```

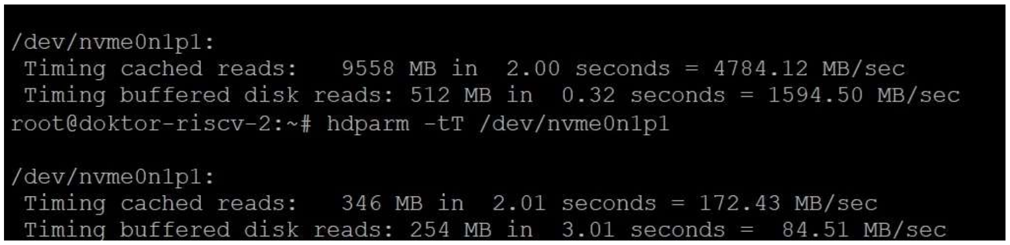
Even if there were no packages, the compilation process for Docker isn't a big challenge. It takes a couple of hours, but Docker does work after it. After this, Docker containerization features are fully available, feature-par with features available on the x86 platform. This also makes the deployment experience on par with our experience for years on x86 platforms. Let's now see if the same applies to the process of deploying Kubernetes on the ARM platform.

2.2. Kubernetes Deployment

Kubernetes deployment has always been more involved than Docker deployment, which is reasonable, as it's a much bigger platform. However, deployment on ARM closely resembles the deployment process on x86 counterparts. The detailed installation procedure is available online [57]. With a few tweaks here and there, it's the same for all our ARM platforms.

3. RISC-V as a Platform for Docker and Kubernetes

The RISC-V architecture has become increasingly popular in recent years because of its open-source nature, which enables more customization and freedom in designing processors. This architecture's scalability and cost-effectiveness make it suitable for cloud computing, IoT, and edge computing applications. However, there are several obstacles and constraints to the RISC-V story. The first one we will mention is the very slow NVMe controller. Figure 1 clearly shows the difference in performance between the NVMe controller on the TuringPI2 platform versus the RISC-V platform with the same SSD:



```
/dev/nvme0n1p1:
 Timing cached reads:   9558 MB in  2.00 seconds = 4784.12 MB/sec
 Timing buffered disk reads: 512 MB in  0.32 seconds = 1594.50 MB/sec
root@doktor-riscv-2:~# hdparm -tT /dev/nvme0n1p1

/dev/nvme0n1p1:
 Timing cached reads:   346 MB in  2.01 seconds = 172.43 MB/sec
 Timing buffered disk reads: 254 MB in  3.01 seconds =  84.51 MB/sec
```

Figure 1. NVMe SSD speed comparison, Turing RK1 vs SiFive platform.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

Operationally speaking, SiFive's RISC-V platform has one big issue - it's unable to boot from NVMe - it only boots from microSD, which is much slower. For reference, we're talking about 50MB/sec cached reads and 1.51 MB/s buffered disk reads, which makes it unusable except for the initial boot and a bit of configuration to make the platform use NVMe as an Ubuntu root partition drive. Even the regular package deployment processes can become unusably slow if we were to go down that route, which is not recommended. This would be a huge issue if we wanted to run containers from a local disk.

Implementing Docker on RISC-V is reasonably seamless, thanks to the flexibility of Linux as a platform, which serves as the foundation for Ubuntu and Docker's containerization technologies. Nevertheless, difficulties arise in the process of coordinating these containers using Kubernetes. Kubernetes is essential for efficiently managing large-scale containerized applications commonly found in cloud computing settings. Regrettably, there is no complete and officially endorsed version of Kubernetes available for the RISC-V architecture, and there's also a significant lack of available RISC-V-compatible Docker containers with which to work. For example, there's no official Ubuntu RISC-V image available at the time of writing this paper. Therefore, the potential for implementing Kubernetes in a production setting on RISC-V processors is significantly restricted [52,56].

The sole existing binary package of Kubernetes for RISC-V is version 1.16, providing solely fundamental services. As a result of this constraint, certain sophisticated functionalities of Kubernetes, including automatic recovery, scalability, and gradual upgrades, may not operate as intended or necessitate substantial adjustments and customization. Furthermore, the absence of support from the upstream source means that any upgrades or security patches must be done manually, making it more complicated and increasing the risks involved in maintaining a Kubernetes cluster on RISC-V [58].

Notwithstanding these obstacles, endeavors have been made to narrow the divide. An orchestration platform called KubeEdge-V has been created explicitly for RISC-V computers. This platform establishes the essential elements necessary to facilitate the fundamental functionalities of containerization and orchestration. It has undergone testing on a prototype system utilizing SiFive processors. Nevertheless, this solution is under development and does not give Kubernetes a complete array of functionalities on well-established architectures such as x86 or ARM [59,60].

RISC-V processors present promising opportunities for open-source hardware and software ecosystems. The utilization of Docker and Kubernetes on these processors, particularly on Ubuntu, is still at an early stage of development. The absence of a comprehensively endorsed Kubernetes version and the restricted capabilities of the current binary package are substantial obstacles to extensive adoption. Continued progress and assistance from the community will be essential in overcoming these obstacles and fully harnessing the capabilities of RISC-V in cloud-native settings.

First, Linux must be deployed on the set of RISC-V nodes. The deployment process for these platforms is more involved than using an x86 platform. That's partially due to the hardware choices we made and partially due to the immaturity of these platforms. Deployment for the SiFive-based RISC-V platform was as painless as possible:

- Download the Ubuntu Server 24.04 RISC-V image;
- Unpack the image and flash it on an SD card for installation by using Raspberry Pi Imager;
- Connect the serial console and follow the standard Ubuntu boot procedure.

After that, it would be prudent to make the board boot the root filesystem from the NVMe drive—it's much faster than the microSD. We had to change a few settings in the /etc/fstab file and some u-boot configuration files. After that, a system-wide upgrade to the latest packages is recommended (apt-get -y upgrade), and a mandatory reboot after the new kernel has been deployed. The following steps involve installing Docker and Kubernetes (if possible), which we will do in the following two sub-sections.

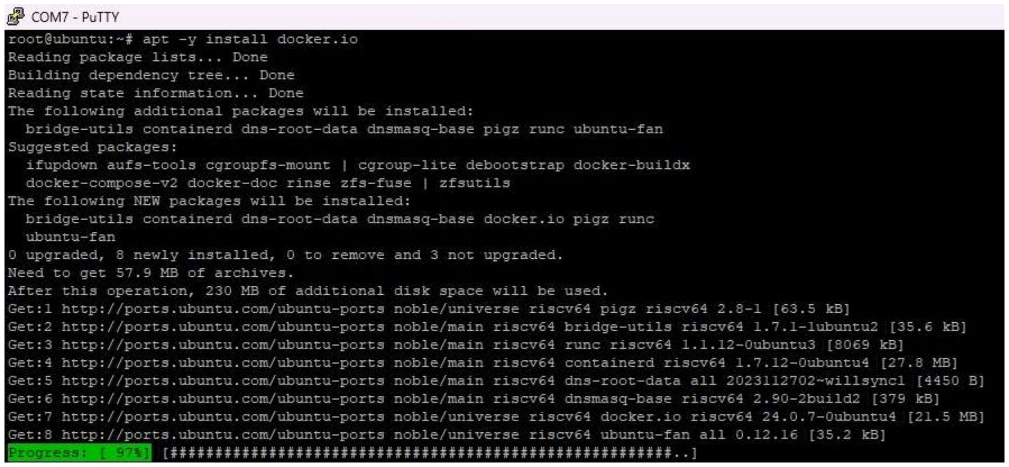
3.1. Docker Deployment

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

Since we started finishing this paper a couple of months ago, the situation with Docker deployment has improved immensely. There's a set of available Docker packages available in repositories for Ubuntu 24.04 for RISC-V, so the installation process for Docker is straightforward:

apt -y install docker.io

This is a recent development in the Ubuntu/RISC-V world, as these packages were unavailable when we started writing this paper a couple of months ago. Results are visible in Figure 2:



```
root@ubuntu:~# apt -y install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx
  docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc
  ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 3 not upgraded.
Need to get 57.9 MB of archives.
After this operation, 230 MB of additional disk space will be used.
Get:1 http://ports.ubuntu.com/ubuntu-ports noble/universe riscv64 pigz riscv64 2.8-1 [63.5 kB]
Get:2 http://ports.ubuntu.com/ubuntu-ports noble/main riscv64 bridge-utils riscv64 1.7.1-1ubuntu2 [35.6 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports noble/main riscv64 runc riscv64 1.1.12-0ubuntu3 [8069 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports noble/main riscv64 containerd riscv64 1.7.12-0ubuntu4 [27.8 MB]
Get:5 http://ports.ubuntu.com/ubuntu-ports noble/main riscv64 dns-root-data all 2023112702~willsync1 [4450 B]
Get:6 http://ports.ubuntu.com/ubuntu-ports noble/main riscv64 dnsmasq-base riscv64 2.90-2build2 [379 kB]
Get:7 http://ports.ubuntu.com/ubuntu-ports noble/universe riscv64 docker.io riscv64 24.0.7-0ubuntu4 [21.5 MB]
Get:8 http://ports.ubuntu.com/ubuntu-ports noble/universe riscv64 ubuntu-fan all 0.12.16 [35.2 kB]
Progress: [ 97%] [##########################################################..]
```

Figure 2. Docker deployment on the RISC-V platform is now available.

Even if there were no packages, the compilation process for Docker isn't a big challenge. It takes a couple of hours, but Docker does work after it.

3.2. Kubernetes Deployment

Unfortunately, Kubernetes still doesn't have current upstream packages for RISC-V. In this sense, there are three available options:

1. To compile Kubernetes from source code (a very complex task that won't necessarily end up being successful);
2. To use the only available binary package for Kubernetes, version for RISC-V (1.16 from September 2019), available online [61];
3. To install and run k3s.

K3s is a version of K8s with a much smaller footprint; it uses fewer resources, the configuration is more straightforward, albeit with a limited set of options, and it's not meant to be scalable and highly available for production-level environments. It is also much more limited in features and extensions while offering limited compatibility with standard K8s tools and extensions. We used three RISC-V nodes based on SiFive HiFive Unmatched boards. We deployed the available, minimal Kubernetes v1.16 package to evaluate whether using RISC-V as a platform makes sense for k8s workloads. But we also must make note of one simple fact - this package doesn't contain the full k8s environment with all modules and addons - it just contains the minimum services, like:

- set of required services and binaries, like kubectl, kubeadm, etc.;
- apiserver;
- controller-manager;
- scheduler;
- proxy;
- pause (for pod network namespace);
- etcd;
- coredns.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

First and foremost, a couple of dependencies must be deployed before the k8s v1.16 package deployment. We need to employ a set of commands as described on Carlos Eduardo's GitHub page [62]. Since this GitHub page was made, many new Docker versions have been released, so it's expected to get some warnings about k8s v1.16 not being compatible with, for example, Docker 24.0.7.

After the package deployment on our three nodes, the Kubernetes cluster works, and we can do some performance evaluation with it. But we also need to point out the fact that this package version is five years old and it's missing a whole bunch of new features that were introduced during that time, such as:

- changes to Ingress controller (1.18);
- better CLI support, logging, new APIs, CSI health monitoring (v1.19);
- docker deprecation (v1.20);
- changes to kubelet logging, storage capacity tracking (v1.21);
- external credential providers support (v1.22);
- dual-stack IPv4/IPv6 networking, HorizontalPodAutoscaler v2 changes (v1.23);
- removal of Dockershim from kubelet, changes in storage plugins (v1.24);
- cgroups v2 support, further changes in storage plugins (v1.25);
- API changes (v1.26);
- iptables performance improvements (v1.27);
- changes to Ceph support (removal of the CephFS plugin in favor of CephFS CSI driver) (v1.28), etc.

Furthermore, many stability issues exist when deploying Kubernetes from the binary package on Ubuntu 24.04. The Kubelet service times out occasionally (even during the cluster initialization phase), containers sometimes fail to start, issues with networking and firewalling, problems with the cgroups v2 subsystem, etc. However, we got it up and running and ran some tests to understand how this platform performs compared to ARM-based platforms.

4. Experimental Setup and Study Methodology

When we started working on this paper a couple of years ago, the priority was to get access to hardware to do real-life performance evaluations, not to write about theory and technical marketing. Years later, these platforms are still challenging to get, especially in volume. The availability of ARM servers in the EU region is poor. RISC-V is even worse, although it has been years since various vendors promised that they'll be available. It is a bit better in 2024. Still, no high-performance RISC-V processors are available, and - for example - ARM Ampere-based multicore system availability isn't much better.

Ultimately, we've opted to do our software and performance evaluations based on readily available platforms – a set of TuringPI2 platforms plus a selection of ARM-based compute modules for ARM systems and SiFive HiFive Unmatched Rev B for RISC-V. For TuringPi compute modules, we acquired Turing RK1, CM4-based Raspberry Pi CM4 modules, and NVIDIA Jetson TX2 NX. Turing RK1s based on Rockchip RK3588 are by far and away the most performant modules within the price envelope. At the time of writing this paper, the TuringPI2 cluster board plus an RK1 price was comparable to SiFive Unmatched Rev B with a RISC-V CPU if we add the cost of memory that was an extra cost for the RISC-V board. Price similarity gave us a good baseline with which to work.

Regarding performance evaluations, we focused on a stack of CPU, memory, and disk evaluations implemented by a set of custom containers managed by Kubernetes. This means that all the scores will be from the perspective of an Alpine container with the necessary tools (stress-ng, sysbench, etc.) installed inside. There was no point in using any GPU tests as GPUs are far from being supported on the RISC-V platform, making the comparison moot. However, we will reflect on that in our Discussion section to provide the correct information.

For HPC performance evaluations, we decided to use a standard set of performance evaluations based on HPCC (High-Performance Challenge), as it has different test suites and gives us a broad performance evaluation for various types of workloads. First and foremost, HPCC needed to be compiled for every one of these platforms. For that, we also had to compile OpenBLAS (Open Basic Linear Algebra Subprograms) library, then compile HPCC (which required a custom Makefile per

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

platform, and then all that was merged into a per-platform Docker image container to keep the methodology constant across all performance evaluations. We used the latest OpenBLAS library (v0.3.28) and the latest version of HPCC (1.5.0). Also, as we used Ubuntu Linux across all of our platforms, we had to install some dependencies, which was done via the same command on all platforms:

apt -y install build-essential hwloc libhwloc-dev libevent-dev gfortran libblas-dev liblapack-dev mpich libopenmpi-dev make

On our GitHub page dedicated to this paper [63], we published the procedure for compiling OpenBLAS, installing these dependencies, and finishing Makefiles for HPCC for all platforms. Configuration and compilation processes for these utils take quite a while, so we're publishing these configuration details for transparency reasons in case someone needs them for verification.

5. ARM and RISC-V Performance Evaluation

We used a set of standardized tests for performance evaluation, like stress-ng and sysbench, where available (sysbench is not supported on RISC-V architecture). We focused on CPU and memory performance paired with power usage, as this seemed like a reasonable scenario—these platforms should be efficient compared to x86 platforms. We used an HP ProLiant Gen8 server based on an Intel Xeon E5-2680 CPU for context reasons—we wanted to see how the performance of all these RISC-V and ARM platforms stacks up against a similarly priced x86 CPU, no matter the fact that E5-2680 is a twelve-year-old CPU. The RK3588 processor mentioned in the performance evaluations is the CPU on the Turing RK1 compute module.

Let's start with single-core performance, as this is very important when dealing with various types of workloads based on containers.

5.1. Single-Core Performance

In single-core performance tests, the Turing RK1 ARM-based system wins considerably. What's surprising is that all other ARM-based platforms and RISC-V-based U740 are nowhere to be found in that respect. We do have to note, though, that NVIDIA Jetson-based TX2 NX does have a built-in GPU with 256 CUDA (Compute Unified Device Architecture), which is one of the reasons why the CPU part of it is losing by such a margin, as can be seen in Figure 3:

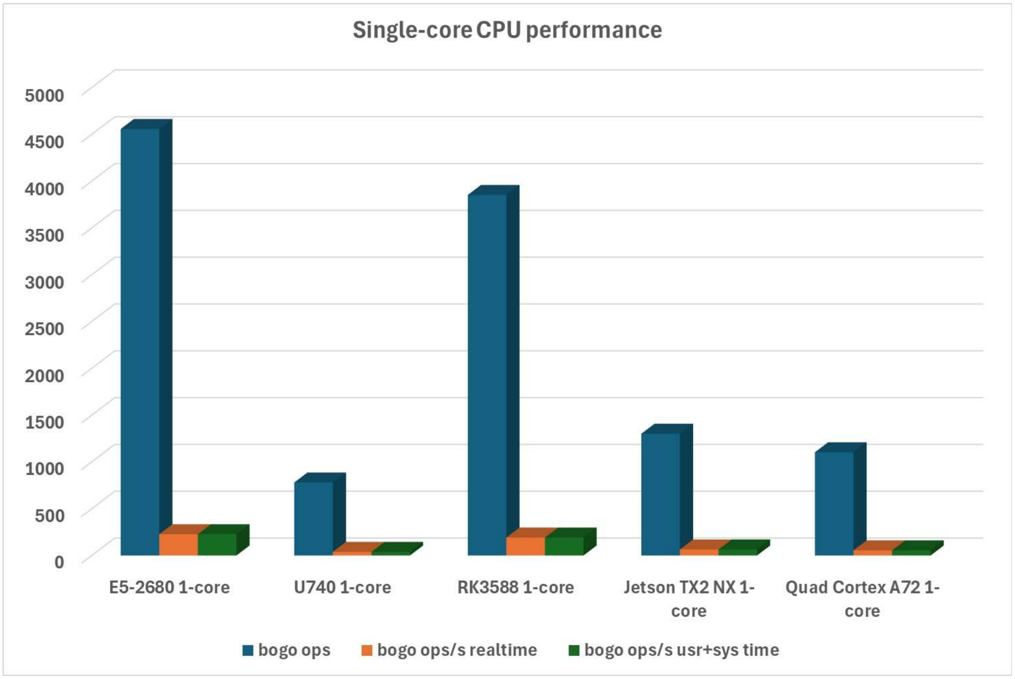


Figure 3. Single-core CPU performance for all platforms.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

Regarding memory performance, the Turing RK1 ARM-based system is miles ahead of everything else. But the exciting part is the fact that it's also significantly faster than our x86-based system, as can be seen in Figure 4:

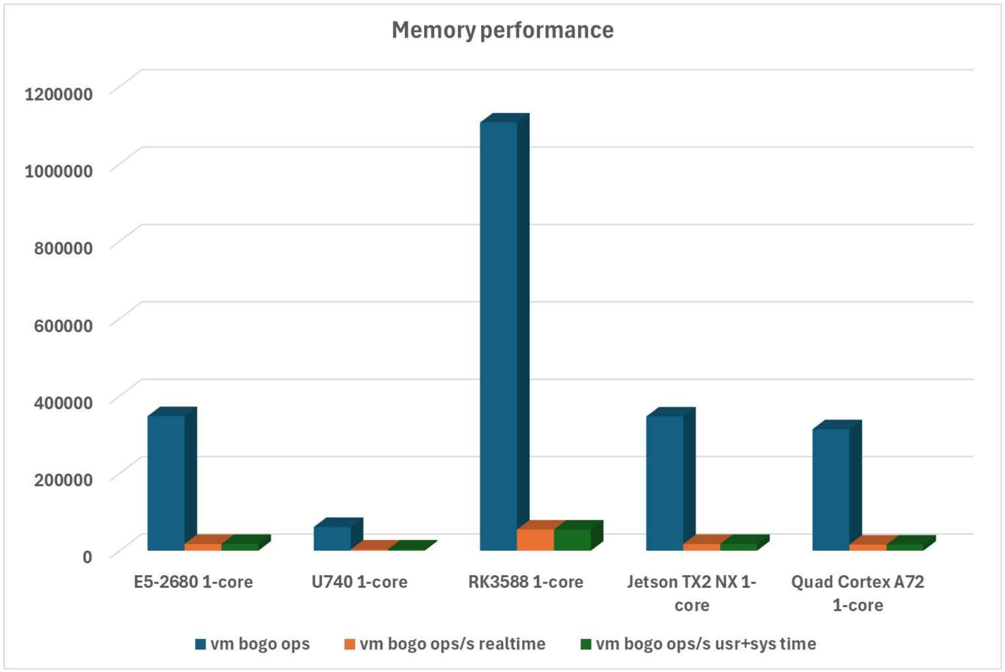


Figure 4. Memory performance with a single CPU core used for all platforms.

The RISC-V-based systems' score (U740) is a bit misleading in the following performance chart because we cannot do a sysbench latency test on it, so - it didn't post any scores. If we exclude that result, we can see that Turing RK1 is still much better than anything else, including other ARM CPUs that also have memory built-in, as can be seen in Figure 5:

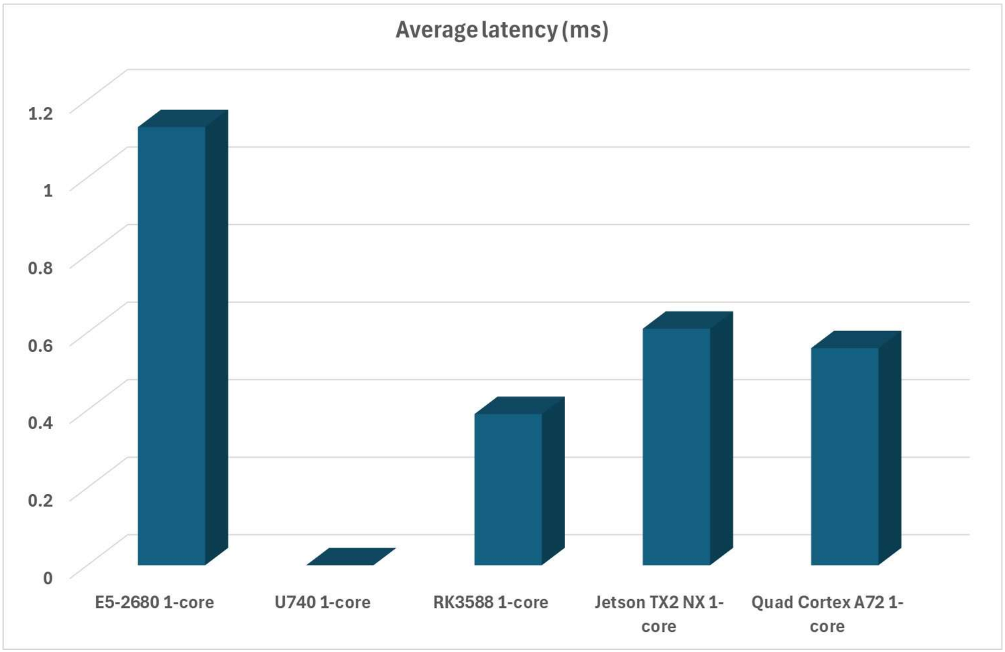


Figure 5. Average compute latency of all available platforms in single-core scenario.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

We'll continue our performance evaluations with multi-core performance tests to see how performance scales across all available cores. E5-2680 has 32 cores with HyperThreading enabled, RK3588 is an 8-core CPU, and all the other CPUs have four cores. We can expect this to impact performance results significantly, but that's the whole point—we need a complete overview of a platform's performance.

5.2. All-Core Performance

With 32 available x86 cores and significantly faster frequency, it's no wonder that E5-2680 is far ahead of every other CPU - but that's also not the point. If we evaluate all different platforms, we can again see the Turing RK1 compute module being far ahead of all other assessed platforms, as can be seen in Figure 6:

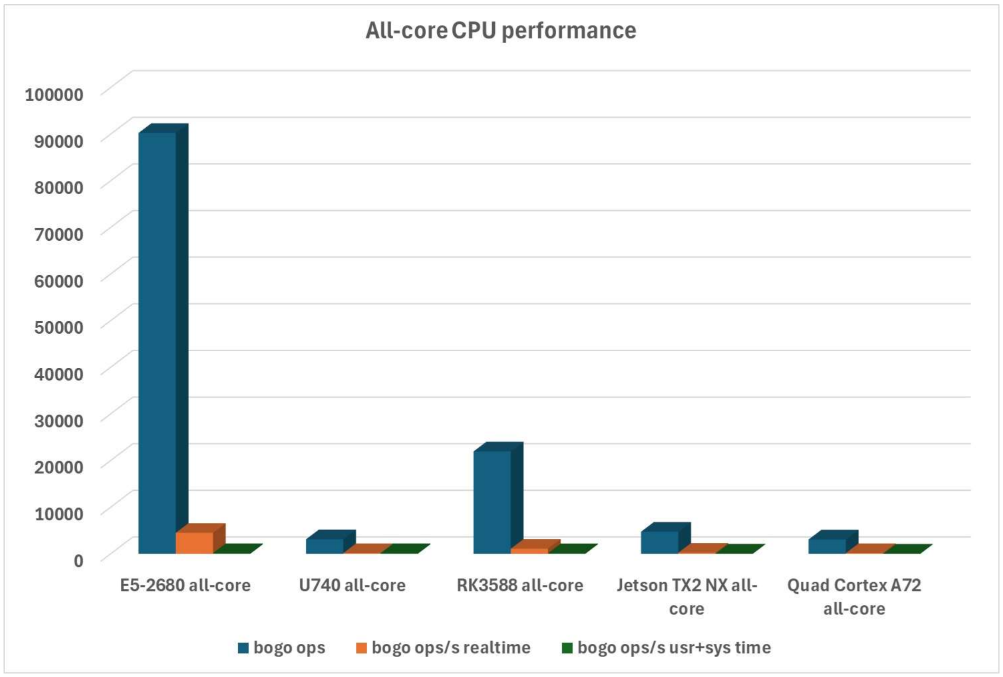


Figure 6. All-core CPU performance for all platforms.

Memory performance for the all-core scenario continues the same trend, as the Rockwell RK3588-based Turing RK1 compute module still has a significant lead compared to RISC-V and other ARM platforms, as can be seen in Figure 7:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

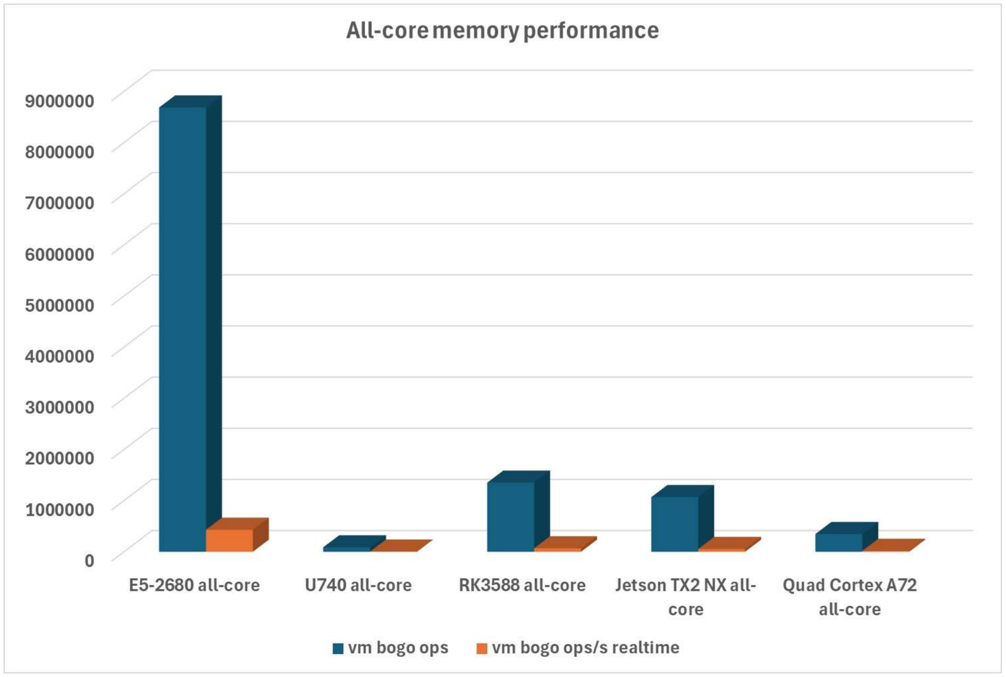


Figure 7. Memory performance with all CPU cores used for all platforms.

Again, ignoring that we can't do latency testing on the RISC-V platform (U740 chip), Turing RK1 still does very well, although the A72-based CPU has a tiny bit less latency (0.57 ms vs 0.58 ms). But this time, we need to take note and compare this latency to the x86-based system, as there's a world of difference between them, with the x86-based system having almost three times the latency as the ARM-based platforms. This is one of the fundamental issues with x86 platforms in general - memory is too far away from the CPU to be less latent, and the built-in caches cannot make that much of a difference compared to CPUs with memory on-chip as ARM chips have. Intel and AMD announced that this issue will be addressed in some of the future x86 chips in the next couple of generations as this design feature has the most detrimental influence on performance. The average latency of all platforms can be seen in Figure 8:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

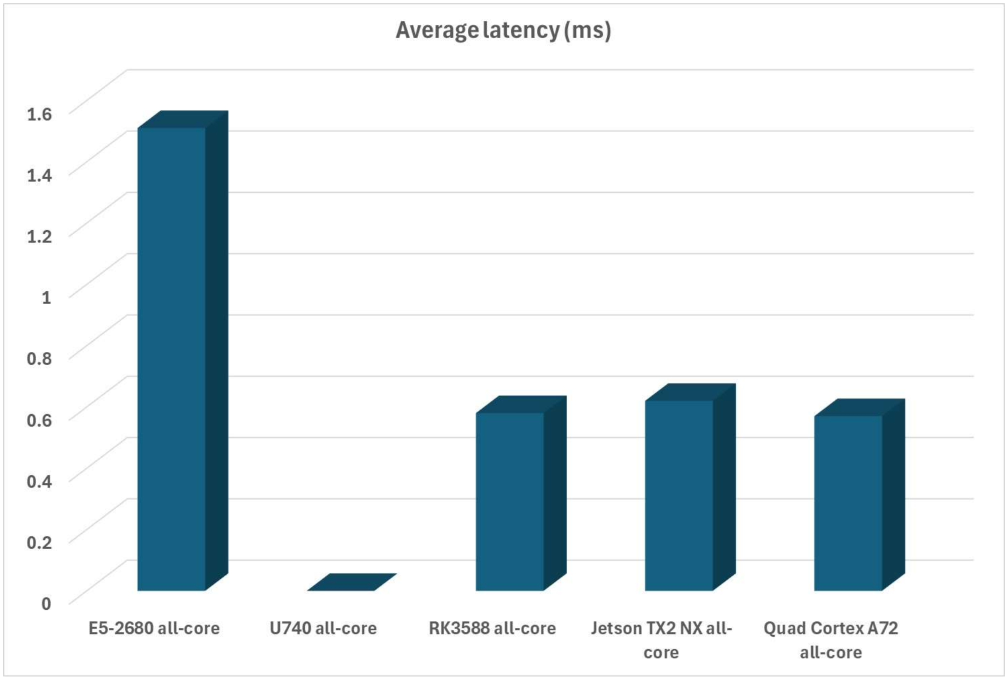


Figure 8. Average compute latency of all available platforms in an all-core scenario.

ARM-based systems are a much better choice for CPU or memory-intensive workloads. It's surprising how much faster they are, especially compared to the similarly priced RISC-V platform. Let's now do some essential HPC-related evaluations to see what performance we can expect from these platforms.

5.3. HPC Performance

The first stack of tests that we did for our HPC performance evaluation was related to HPL (High-Performance Linpack) in terms of available TFLOPS (Tera Floating Operations Per Second) and HPL time (time required to finish the evaluation). The TFLOPS evaluation, which tells us the story of how much faster ARM-based platforms are than RISC-V platform (especially Turing RK1 vs RISC-V), can be seen in Figure 9:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

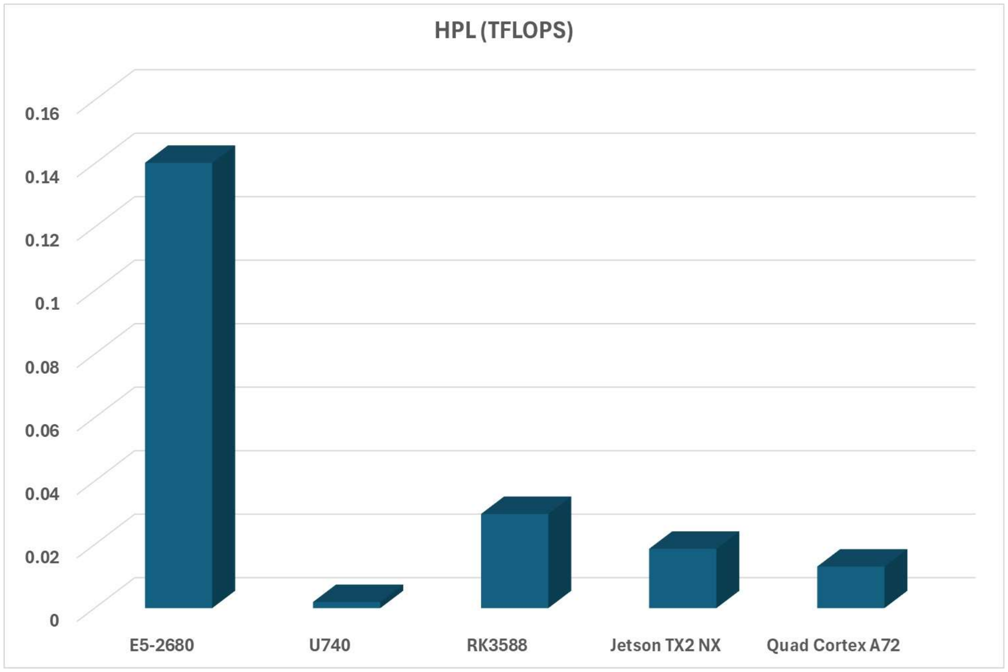


Figure 9. HPL TFLOPS evaluation for all platforms.

HPL time measures the amount of time needed to finish the benchmark, which is inversely proportional to the TFLOPS measurement, as can be seen in Figure 10:

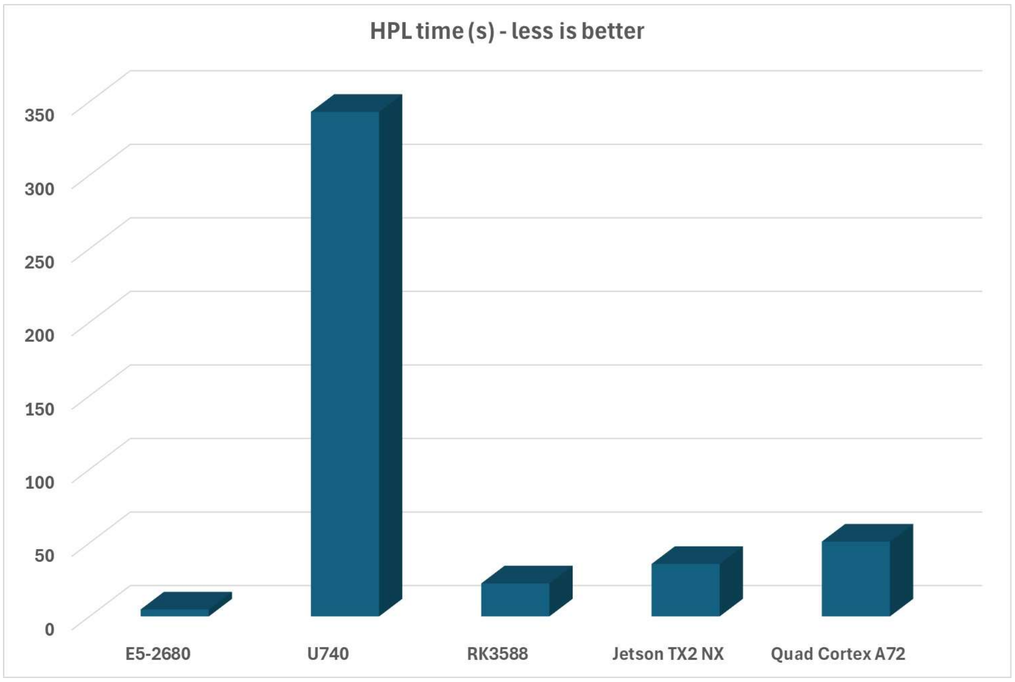


Figure 10. HPL evaluation in terms of time required for HPL evaluation on all platforms.

DGEMM, part of the HPCC benchmark, measures floating point and double-precision matrix-to-matrix multiplication performance. These performance evaluations, as well as some that are coming up next, should scale similarly to the HPL GFLOPS scores, and they do, as can be seen in Figure 11:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

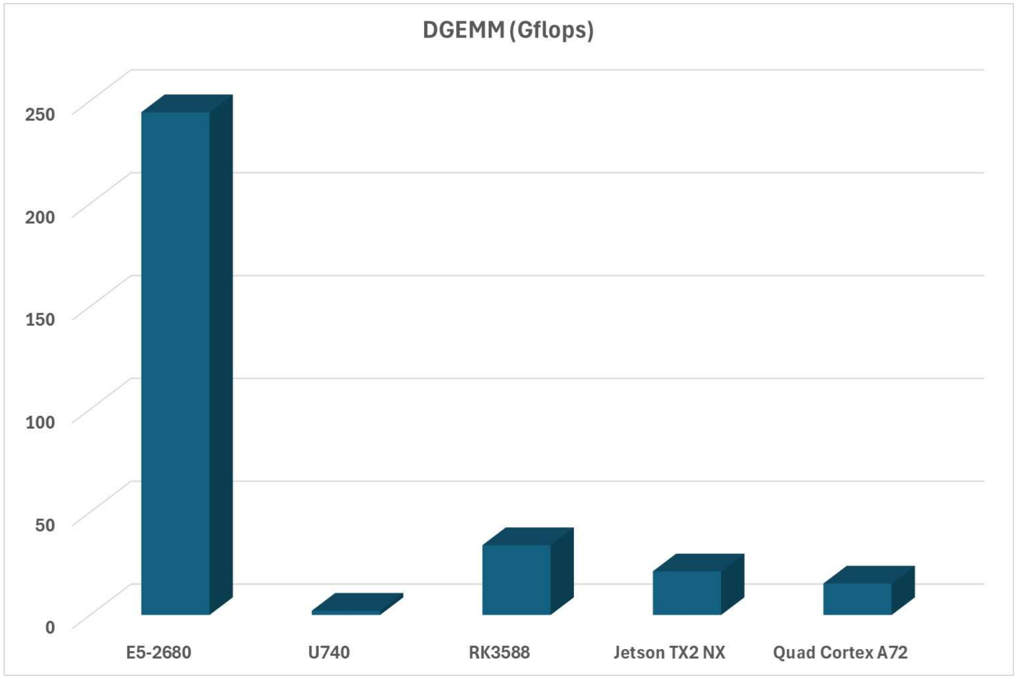


Figure 11. DGEMM double precision scores.

PTRANS measurement evaluates the parallel matrix transpose capabilities of our platforms. RandomAccess is a part of that evaluation, measuring random access performance for large data arrays in multicore scenarios. Evaluation results can be seen in Figure 12:

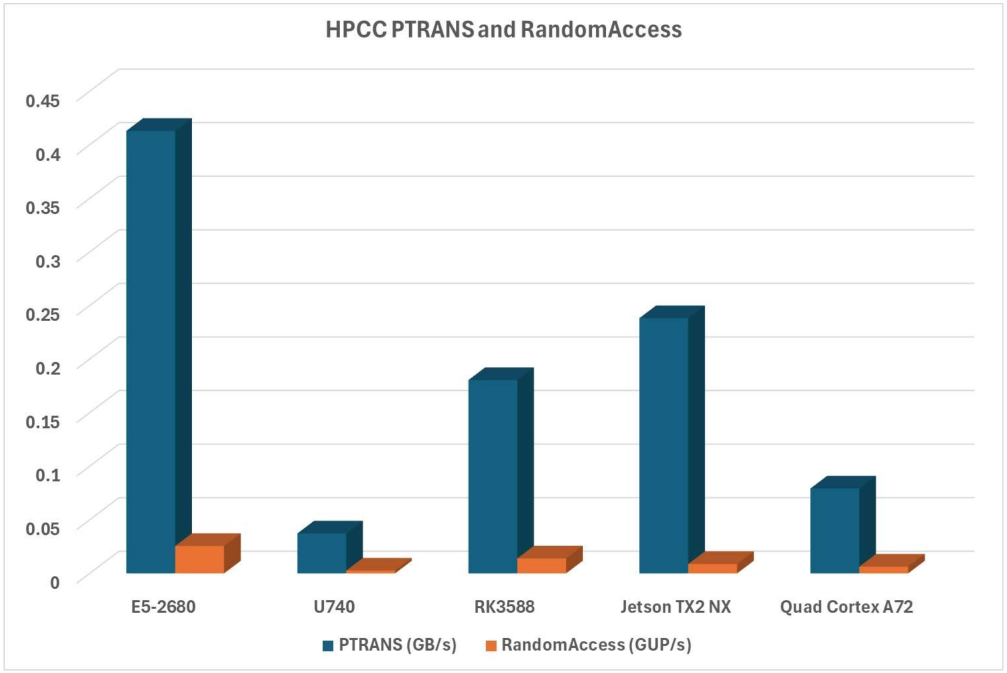


Figure 12. PTRANS and RandomAccess evaluations for large arrays.

HPCC STREAM evaluates the sustainable memory bandwidth transfer in GB/s. RISC-V platform falls to the bottom here, but ARM platforms are surprisingly close to the x86 platform (especially when we consider that they're consuming much less power), as can be seen in Figure 13:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

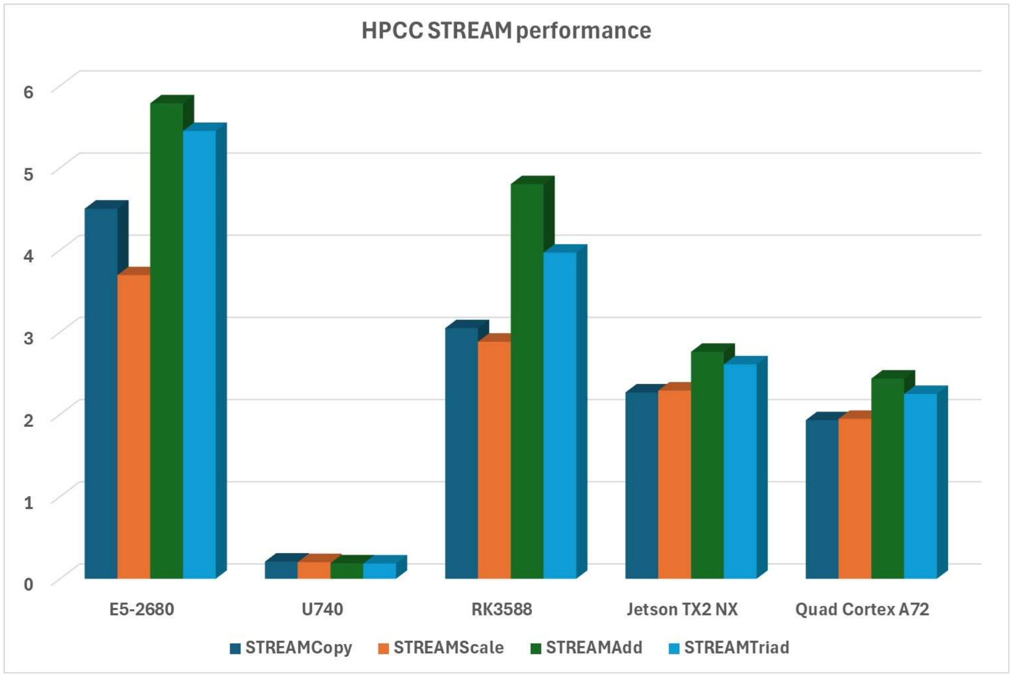


Figure 13. HPCC STREAM performance evaluation.

The last set of performance evaluations is related to FFT, which measures the floating-point DFT (Discrete Fourier Transform) execution rate. Again, Turing RK1 is very close to the x86 system here, and the other ARM platforms are far ahead of the RISC-V platform. as can be seen in Figure 14:

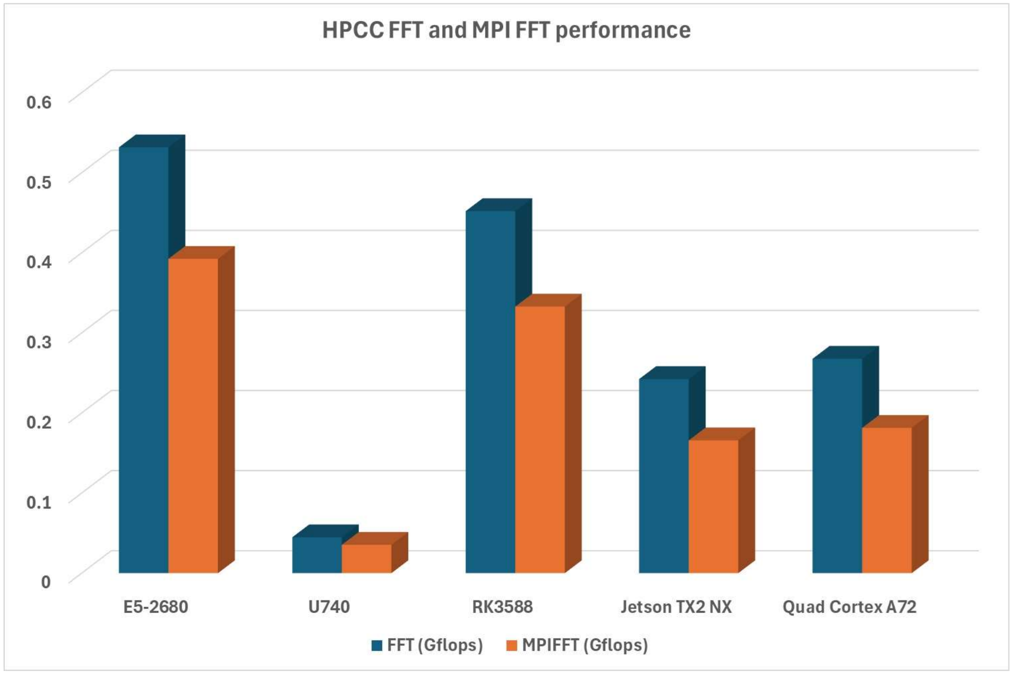


Figure 14. HPCC FFT performance analysis.

In all the performance metrics we could show in this paper (and quite a few more), ARM platforms are much faster than anything the RISC-V platform can offer for the same price. Let's discuss this in a bit more detail in the next section.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

6. Discussion

We can conclude that the ARM platform is much more robust and production-ready than the RISC-V platform. Of course, this isn't surprising, as it's been on the market for decades. ARM has experience designing CPU architectures from billions of processors being used in various devices, so this was to be expected.

What we did not expect, however, was the incredible difference in performance between our RISC-V platform and all the ARM platforms. The RISC-V platform is much slower in terms of performance and latency. If we were to discuss these results based on the timeline, a direct comparison could be made between the A72 ARM CPU and the RISC-V platform, as they were launched almost simultaneously. The difference in memory performance (four times plus faster), for the same basic CPU performance in all-core, and better single-core performance is notable when comparing Quad Cortex A72 ARM core to U740 RISC-V core.

Then, there's the comparison to Turing RK1 and NVIDIA Jetson TX2 NX. Yes, both platforms are newer than the RISC-V platform, although TX2 NX was introduced only a few months after U740, while the RK1 was introduced a year and a half later. However, the performance difference, even accounting for the 256 CUDA cores in Jetson TX2 NX, is staggering. We compare them for the same amount of money and a much more favorable power envelope. Jetson's memory performance is roughly 5x the U740, while RK3588 is 15x plus times faster in memory performance. The CPU performance gap is also quite big - RK3588 is approximately 5x faster, and Jetson TX2 NX is approximately 2x faster than U740. Suppose we count the CUDA cores on Jetson; that makes the comparison even worse. That's why, if we were to deploy micro-clusters for Docker/Kubernetes for either cloud services or super-efficient HPC environments, TuringPi platforms based on NVIDIA compute modules, and Turing RK1 are a much more efficient and faster solution. The only fact that works in the RISC-V platform's favor is its PCI-Express slot on the motherboard. But that advantage is null and void when we look at the following facts:

- The only officially supported PCIe graphics cards are AMD RX 500-Series and Radeon HD 6000-Series VGA cards, which are both old and don't run CUDA formally, so they cannot be used to accelerate anything;
- There's no support for CUDA on the RISC-V platform, even if the platform supports NVIDIA GPUs;
- There are no known FPGAs that can be used on RISC-V;
- There are no known ASICs that can be used on RISC-V.

The big plus of RISC-V—the fact that it's an open-source platform—will only start paying dividends when critical players on the market support the platform for familiar use cases. There are currently EU-sponsored projects, such as the European Processor Initiative, for developing an HPC ecosystem based on the RISC-V core. This is where concepts like FAUST [64] will shine - these sorts of specialized acceleration units that can be integrated with RISC-V architecture are where RISC-V's forte will come to the fore. However, it will also take time, as RISC-V is currently not well supported on the software side, while the basic hardware side still needs quite a bit of additional development.

Looking at the performance analytics charts, we can see why ARM, specifically the TuringPI2 platform, is used so often, especially in the education sector, to teach the different ways to do distributed programming and HPC-related topics. These platforms are very price competitive, highly capable, and offer incredible consolidation ratios. When the platform has all four nodes running at full speed, we can have four independent nodes in one mini-ITX system that consumes less than 70W of power, which is incredibly power-efficient compared to anything x86 offers. ARM Ampere-based systems would probably be an even better example to illustrate that point, which is what we will try to acquire in the future to do further evaluations. However, TuringPi2 systems can handle Docker and Kubernetes, have full upstream support for those platforms, and can be procured quickly and used for educational and production tasks.

7. Future Works

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

We see multiple exciting research areas for the future of heterogeneous computing based on different ISAs, especially in HPC. These research areas depend on Intel, AMD, NVIDIA, and others to further develop their ARM, especially RISC-V-based software stack, and offer readily available software support to continue the research path. Given better workload scheduling, heterogeneous HPC clusters could provide a massive bump in energy efficiency with more development.

Further research is needed to optimize RISC-V performance. Given its open-source nature, RISC-V could use more microarchitectural enhancements, but even more so, with the integration of various hardware accelerators (AI, cryptography, …) or different Domain-Specific Architectures for specific industries and tasks, to give RISC-V a bit more foothold in specific niche technology areas.

Research into performance optimizations has to go hand in hand with additional research into compiler and library optimization to boost performance in specific, targeted applications.

Further research needs to be done on energy efficiency and power consumption, especially for various workloads. x86 platforms will be the best overall choice, but there will also be areas where ARM and, potentially, RISC-V might be the correct choice. But, the last five years of development of ARM products for the data center are a great example of that—ARM many-core architectures (for example, Ampere) have taken a strong foothold in the data center space, as they're very efficient for many different tasks. Given the choice, cloud providers will gladly sell us the capability to run Kubernetes/Docker environments on ARM-based architectures, and rightfully so, as they're much more efficient than x86 platforms. This research can lead to workload partitioning across different ISA architectures for bottom-up environments for heterogeneous computing.

More research is needed into standardization in heterogeneous computing to make it easier for researchers and regular or business users to integrate and switch seamlessly between various architectures. For example, we mentioned that Kubernetes is supported on x86 and ARM but not RISC-V. That means that, even if we wanted to, we can't use the same toolset that Kubernetes offers on RISC-V, no matter how hard we try. The latest available binary distribution of Kubernetes on RISC-V is five years old and needs to be brought into the present. Research into the management and implementation of such heterogeneous Kubernetes clusters is underway. Given the popularity of the ARM platform and the rising interest in the RISC-V platform, it seems to be the right way to go [65]. We can see a potential future in which different ISAs will be used for various applications in large-scale heterogeneous environments.

8. Conclusions

The research described in this paper thoroughly examines ARM and RISC-V architectures performance in general and HPC workloads, specifically emphasizing their incorporation with Docker and Kubernetes. It provides thorough empirical assessments, practical insights, and performance comparisons between these two architectures, which are increasingly important in the changing landscape of heterogeneous computing systems.

In most instances, the performance study demonstrates the superiority of ARM over RISC-V. The architecture developed by ARM has consistently shown exceptional performance, particularly in tasks that require high memory usage and extensive processing by the CPU. The Turing RK1, based on the ARM architecture, demonstrated superior performance compared to RISC-V, exhibiting notably better data processing speed and reduced latencies for the same price range. The advantage is particularly highlighted in Docker and Kubernetes environments, where ARM's well-developed ecosystem guarantees smooth integration. The ARM platform's capacity to efficiently manage intricate containerized workloads makes it a more practical choice for HPC applications, especially in cloud and edge computing scenarios where scalability and power economy are essential.

RISC-V, despite its open-source appeal and ability for customization, encounters substantial obstacles. The study indicates that RISC-V's performance is inferior to ARM, particularly in single and multi-core processing. Furthermore, the absence of well-developed software assistance impeded incorporating Docker and Kubernetes on RISC-V platforms. The lack of a comprehensive and officially endorsed Kubernetes version for RISC-V is a significant obstacle to its implementation in production-level environments. Despite the potential for increased flexibility and creativity, these

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

drawbacks of the RISC-V architecture and the added intricacies of deploying and operating Kubernetes clusters on this platform outweigh its current benefits.

RISC-V will exhibit its potential in the mid-term, especially in specialized fields where its open-source characteristics and ability to be tailored to specific needs could be advantageous. Optimizing RISC-V's performance and improving its software ecosystem to increase its competitiveness in heterogeneous computing settings, especially in scenarios where Docker and Kubernetes are becoming more widespread, is an absolute must. Otherwise, RISC-V might end up being an excellent idea that never was—on its own or in heterogeneous environments.

Author Contributions: Conceptualization, V.D., and L.M.; methodology, V.D., and Z.K.; software, Z.K.; validation, V.D., and L.M.; formal analysis, V.D., and G.D.; investigation, Z.K., and G.D.; resources, L.M.; data curation, V.D., and G.D.; writing—original draft preparation, Z.K., and G.D.; writing—review and editing, L.M., and V.D.; supervision, L.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data supporting this study's findings are available from the corresponding author, [L.M.], upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bruhn, F., Brunberg, K., Hines, J., Asplund, L., & Norgren, M. (2015). Introducing radiation tolerant heterogeneous computers for small satellites. 2015 IEEE Aerospace Conference, 1-10. https://doi.org/10.1109/AERO.2015.7119158.
2. Reichenbach, M., Holzinger, P., Häublein, K., Lieske, T., Blinzer, P., & Fey, D. (2018). Heterogeneous Computing Utilizing FPGAs. Journal of Signal Processing Systems, 91, 745-757. https://doi.org/10.1007/S11265-018-1382-7.
3. Feng, L., Liang, H., Sinha, S., & Zhang, W. (2016). HeteroSim: A Heterogeneous CPU-FPGA Simulator. IEEE Computer Architecture Letters, 16, 38-41. https://doi.org/10.1109/FPL.2016.7577386.
4. Chang, L., Gómez-Luna, J., Hajj, I. E., Huang, S., Chen, D., & Hwu, W. (2017). Collaborative Computing for Heterogeneous Integrated Systems. Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering. https://doi.org/10.1145/3030207.3030244.
5. Mittal, S., & Vetter, J. (2015). A Survey of CPU-GPU Heterogeneous Computing Techniques. ACM Computing Surveys (CSUR), 47, 1-35. https://doi.org/10.1145/2788396.
6. Prongnuch, S., & Wiangtong, T. (2016). Heterogeneous Computing Platform for data processing. 2016 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), 1-4. https://doi.org/10.1109/ISPACS.2016.7824762.
7. Rethinagiri, S., Palomar, O., Moreno, J., Unsal, O., & Cristal, A. (2015). Trigeneous Platforms for Energy Efficient Computing of HPC Applications. 2015 IEEE 22nd International Conference on High Performance Computing (HiPC), 264-274. https://doi.org/10.1109/HiPC.2015.19.
8. Kurth, A., Vogel, P., Capotondi, A., Marongiu, A., & Benini, L. (2017). HERO: Heterogeneous Embedded Research Platform for Exploring RISC-V Manycore Accelerators on FPGA. ArXiv, abs/1712.06497. https://doi.org/10.3929/ETHZ-B-000219249.
9. Parnassos, I., Bellas, N., Katsaros, N., Patsiatzis, N., Gkaras, A., Kanellis, K., Antonopoulos, C., Spyrou, M., & Maroudas, M. (2017). A programming model and runtime system for approximation-aware heterogeneous computing. 2017 27th International Conference on Field Programmable Logic and Applications (FPL), 1-4. https://doi.org/10.23919/FPL.2017.8056774.
10. Lopez-Novoa, U., Mendiburu, A., & Miguel-Alonso, J. (2015). A Survey of Performance Modeling and Simulation Techniques for Accelerator-Based Computing. IEEE Transactions on Parallel and Distributed Systems, 26, 272-281. https://doi.org/10.1109/TPDS.2014.2308216.
11. Liao, S.-W., Kuang, S.-Y., Kao, C.-L., & Tu, C.-H. (2019). A Halide-based Synergistic Computing Framework for Heterogeneous Systems. Journal of Signal Processing Systems, 91, 219-233. https://doi.org/10.1007/S11265-017-1283-1
12. Wu, Z., Hammad, K., Beyene, A., Dawji, Y., Ghafar-Zadeh, E., & Magierowski, S. (2022). An FPGA Implementation of A Portable DNA Sequencing Device Based on RISC-V. 2022 20th IEEE Interregional NEWCAS Conference (NEWCAS), 417-420. https://doi.org/10.1109/NEWCAS52662.2022.9842014
13. Young, J., Jezghani, A., Valdez, J., Jijina, S., Liu, X., Weiner, M. D., & Powell, W., Sarajlic, S. (2022). Enhancing HPC Education and Workflows with Novel Computing Architectures. The Journal of Computational Science Education. https://doi.org/10.22369/issn.2153-4136/13/2/6

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

14. Kurth, A., Forsberg, B., & Benini, L. (2022). HEROv2: Full-Stack Open-Source Research Platform for Heterogeneous Computing. IEEE Transactions on Parallel and Distributed Systems, 33, 4368-4382. https://doi.org/10.1109/TPDS.2022.3189390
15. Ojika, D., Gordon-Ross, A., Lam, H., Yoo, S., Cui, Y., Dong, Z., Dam, K. V., Lee, S., & Kurth, T. (2019). PCS: A Productive Computational Science Platform. 2019 International Conference on High Performance Computing & Simulation (HPCS), 636-641. https://doi.org/10.1109/HPCS48598.2019.9188108
16. Fiolhais, L., Gonçalves, F. F., Duarte, R., Véstias, M., & Sousa, J. (2019). Low Energy Heterogeneous Computing with Multiple RISC-V and CGRA Cores. 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 1-5. https://doi.org/10.1109/ISCAS.2019.8702538
17. Reichenbach, M., Holzinger, P., Häublein, K., Lieske, T., Blinzer, P., & Fey, D. (2018). Heterogeneous Computing Utilizing FPGAs. Journal of Signal Processing Systems, 91, 745-757. https://doi.org/10.1007/S11265-018-1382-7
18. Dong, J., Zheng, F., Lin, J., Liu, Z., Xiao, F., & Fan, G. (2022). EC-ECC: Accelerating Elliptic Curve Cryptography for Edge Computing on Embedded GPU TX2. ACM Transactions on Embedded Computing Systems (TECS), 21, 1-25. https://doi.org/10.1145/3492734
19. D'Agostino, D., & Cesini, D. (2021). Editorial: Heterogeneous Computing for AI and Big Data in High Energy Physics. Frontiers in Big Data, 4. https://doi.org/10.3389/fdata.2021.652881
20. Hu, N., Wang, C., & Zhou, X. (2022). FLIA: Architecture of Collaborated Mobile GPU and FPGA Heterogeneous Computing. Electronics. https://doi.org/10.3390/electronics11223756
21. Datta, D., & Gordon, M. S. (2023). Accelerating Coupled-Cluster Calculations with GPUs: An Implementation of the Density-Fitted CCSD(T) Approach for Heterogeneous Computing Architectures Using OpenMP Directives. Journal of Chemical Theory and Computation. https://doi.org/10.1021/acs.jctc.3c00876
22. Lastovetsky, A. L., & Manumachu, R. R. (2020). The 27th International Heterogeneity in Computing Workshop and the 16th International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms. Concurrency and Computation: Practice and Experience, 32. https://doi.org/10.1002/cpe.5736
23. Horta, E., Chuang, H.-R., VSathish, N. R., Philippidis, C. J., Barbalace, A., Olivier, P., & Ravindran, B. (2021). Xar-trek: run-time execution migration among FPGAs and heterogeneous-ISA CPUs. Proceedings of the 22nd International Middleware Conference. https://doi.org/10.1145/3464298.3493388
24. Cerf, V. (2021). On heterogeneous computing. Communications of the ACM, 64, 9-9. https://doi.org/10.1145/3492896
25. Du, D., Liu, Q., Jiang, X., Xia, Y., Zang, B., & Chen, H. (2022). Serverless computing on heterogeneous computers. Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. https://doi.org/10.1145/3503222.3507732
26. Núñez-Yáñez, J. (2019). Energy Proportional Heterogenous Computing with Reconfigurable MPSoC. 2019 International Conference on High Performance Computing & Simulation (HPCS), 642-642. https://doi.org/10.1109/HPCS48598.2019.9188229
27. Carballo-Hernández, W., Pelcat, M., & Berry, F. (2021). Why is FPGA-GPU Heterogeneity the Best Option for Embedded Deep Neural Networks?. ArXiv, abs/2102.01343. https://doi.org/10.48550/arXiv.2102.01343
28. Mahmoud, D. G., Lenders, V., & Stojilović, M. (2022). Electrical-Level Attacks on CPUs, FPGAs, and GPUs: Survey and Implications in the Heterogeneous Era. ACM Computing Surveys (CSUR), 55, 1-40. https://doi.org/10.1145/3498337
29. Freytag, G., Serpa, M., Lima, J. F., Rech, P., & Navaux, P. (2021). Collaborative execution of fluid flow simulation using non-uniform decomposition on heterogeneous architectures. J. Parallel Distributed Comput., 152, 11-20. https://doi.org/10.1016/J.JPDC.2021.02.006
30. Rodríguez, A., Navarro, A., Asenjo, R., Corbera, F., Gran, R., Suárez, D., & Núñez-Yáñez, J. (2019). Parallel multiprocessing and scheduling on the heterogeneous Xeon+FPGA platform. The Journal of Supercomputing, 76, 4645-4665. https://doi.org/10.1007/s11227-019-02935-1
31. Wyrzykowski, R., & Ciorba, F. (2022). Algorithmic and software development advances for next-generation heterogeneous platforms. Concurrency and Computation: Practice and Experience, 34. https://doi.org/10.1002/cpe.7013.
32. Hagleitner, C., Diamantopoulos, D., Ringlein, B., Evangelinos, C., Johns, C., Chang, R. N., D'Amora, B. D., Kahle, J., Sexton, J., Johnston, M., Pyzer-Knapp, E. O., & Ward, C. (2021). Heterogeneous Computing Systems for Complex Scientific Discovery Workflows. 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), 13-18. https://doi.org/10.23919/DATE51398.2021.9474061.
33. Mavrogeorgis, N. (2021). Simplifying heterogeneous migration between x86 and ARM machines. ArXiv, abs/2112.01189. DOI: Link.
34. Thomadakis, P., & Chrisochoides, N. (2022). Towards Performance Portable Programming for Distributed Heterogeneous Systems. ArXiv, abs/2210.01238. https://doi.org/10.48550/arXiv.2210.01238.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

35. Nikov, K., Hosseinabady, M., Asenjo, R., Rodríguez, A., Navarro, A., & Núñez-Yáñez, J. (2020). High-Performance Simultaneous Multiprocessing for Heterogeneous System-on-Chip. ArXiv, abs/2008.08883. DOI: Link.
36. Fuentes, J., López, D., & González, S. (2022). Teaching Heterogeneous Computing Using DPC++. 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 354-360. https://doi.org/10.1109/IPDPSW55747.2022.00069.
37. Thomadakis, P., & Chrisochoides, N. (2023). Runtime Support for Performance Portability on Heterogeneous Distributed Platforms. ArXiv, abs/2303.02543. https://doi.org/10.48550/arXiv.2303.02543.
38. Kavanagh, R., Djemame, K., Ejarque, J., Badia, R. M., & García-Pérez, D. (2020). Energy-Aware Self-Adaptation for Application Execution on Heterogeneous Parallel Architectures. IEEE Transactions on Sustainable Computing, 5, 81-94. https://doi.org/10.1109/TSUSC.2019.2912000.
39. Yu, Y., Zhang, S., Fu, H., Wu, L., Chen, D., Gao, Y., Wei, Z., Jia, D., & Lin, X. (2022). Characterizing uncertainties of Earth system modeling with heterogeneous many-core architecture computing. Geoscientific Model Development. https://doi.org/10.5194/gmd-15-6695-2022.
40. Cheng, Y., Sun, W.-T., Bi, Y., Cheng, Y., Shi, J., Wang, L., Yao, Q., Hu, Q., & Zhang, M. (2022). Large Scale ARM Computing Cluster and its Application in HEP. Proceedings of International Symposium on Grids & Clouds 2022 — PoS(ISGC2022). https://doi.org/10.22323/1.415.0015.
41. Kamaleldin, A., & Göhringer, D. (2022). AGILER: An Adaptive Heterogeneous Tile-Based Many-Core Architecture for RISC-V Processors. IEEE Access, 10, 43895-43913. https://doi.org/10.1109/ACCESS.2022.3168686.
42. Nicholas, G. S., Gui, Y., & Saqib, F. (2020). A Survey and Analysis on SoC Platform Security in ARM, Intel and RISC-V Architecture. 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS), 718-721. https://doi.org/10.1109/MWSCAS48704.2020.9184573.
43. Wang, X., Leidel, J. D., Williams, B., Ehret, A., Mark, M., Kinsy, M., & Chen, Y. (2021). xBGAS: A Global Address Space Extension on RISC-V for High Performance Computing. 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 454-463. https://doi.org/10.1109/IPDPS49936.2021.00054.
44. Tornero, R., Rodríguez, D., Martínez, J. M., & Flich, J. (2023). An Open-Source FPGA Platform for Shared-Memory Heterogeneous Many-Core Architecture Exploration. 2023 38th Conference on Design of Circuits and Integrated Systems (DCIS), 1-6. https://doi.org/10.1109/DCIS58620.2023.10335973.
45. Gómez-Sánchez, G., Call, A., Teruel, X., Alonso, L., Morán, I., Perez, M. A., Torrents, D., & Berral, J. L. (2023). Challenges and Opportunities for RISC-V Architectures towards Genomics-based Workloads. https://doi.org/10.48550/arXiv.2306.15562.
46. Stoyanov, S., Kakanakov, N., & Marinova, M. (2023). Secure Heterogeneous Architecture based on RISC-V and root-of-trust. Proceedings of the 24th International Conference on Computer Systems and Technologies. https://doi.org/10.1145/3606305.3606312.
47. Gonzalez, A., Zhao, J., Korpan, B., Genç, H., Schmidt, C., Wright, J., Biswas, A., Amid, A., Sheikh, F., Sorokin, A., Kale, S., Yalamanchi, M., Yarlagadda, R., Flannigan, M., Abramowitz, L., Alon, E., Shao, Y., Asanović, K., & Nikolić, B. (2021). A 16mm2 106.1 GOPS/W Heterogeneous RISC-V Multi-Core Multi-Accelerator SoC in Low-Power 22nm FinFET. ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC), 259-262. https://doi.org/10.1109/ESSCIRC53450.2021.9567768.
48. Kamaleldin, A., Hesham, S., & Göhringer, D. (2020). Towards a Modular RISC-V Based Many-Core Architecture for FPGA Accelerators. IEEE Access, 8, 148812-148826. https://doi.org/10.1109/ACCESS.2020.3015706.
49. Jia, H., Valavi, H., Tang, Y., Zhang, J., & Verma, N. (2020). A Programmable Heterogeneous Microprocessor Based on Bit-Scalable In-Memory Computing. IEEE Journal of Solid-State Circuits, 55, 2609-2621. https://doi.org/10.1109/JSSC.2020.2987714.
50. Vohra, D. Installing Kubernetes Using Docker. Kubernetes Microservices with Docker 2016, 3–38. https://doi.org/10.1007/978-1-4842-1907-2_1.
51. Chen, C.; Hung, M.; Lai, K.; Lin, Y. Docker and Kubernetes. Industry 4.1 2021, 169–213. https://doi.org/10.1002/9781119739920.CH5.
52. Menegidio, F.B.; Jabes, D.L.; Costa de Oliveira, R.; Nunes, L.R. Dugong: A Docker Image, Based on Ubuntu Linux, Focused on Reproducibility and Replicability for Bioinformatics Analyses. Bioinformatics 2017, 34, 514–515. https://doi.org/10.1093/bioinformatics/btx554.
53. El Haj Ahmed, G.; Gil-Castiñeira, F.; Costa-Montenegro, E. KubCG: A Dynamic Kubernetes Scheduler for Heterogeneous Clusters. Softw Pract Exp 2020, 51, 213–234. https://doi.org/10.1002/spe.2898.
54. Eiermann, A.; Renner, M.; Großmann, M.; Krieger, U.R. On a Fog Computing Platform Built on ARM Architectures by Docker Container Technology. Communications in Computer and Information Science 2017, 71–86. https://doi.org/10.1007/978-3-319-60447-3_6.
55. Fornari, F.; Cavalli, A.; CESINI, D.; Falabella, A.; FATTIBENE, E.; Morganti, L.; Prosperini, A.; Sapunenko, V. Distributed Filesystems (GPFS, CephFS and Lustre-ZFS) Deployment on Kubernetes/Docker Clusters.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 August 2024 doi:10.20944/preprints202408.0867.v1

Proceedings of International Symposium on Grids &amp; Clouds 2021 — PoS(ISGC2021) 2021. https://doi.org/10.22323/1.378.0020.
56. Lumpp, F.; Barchi, F.; Acquaviva, A.; Bombieri, N. On the Containerization and Orchestration of RISC-V Architectures for Edge-Cloud Computing. Proceedings of the 3rd Eclipse Security, AI, Architecture and Modelling Conference on Cloud to Edge Continuum 2023. https://doi.org/10.1145/3624486.3624490.
57. Medium.com. Available online: https://medium.com/@bsatnam98/setup-of-a-kubernetes-cluster-v1-29-on-raspberry-pis-a95b705c04c1 (accessed on 09 Aug 2024.).
58. Butler, S.; Gamalielsson, J.; Lundell, B.; Brax, C.; Persson, T.; Mattsson, A.; Gustavsson, T.; Feist, J.; Öberg, J. An Exploration of Openness in Hardware and Software Through Implementation of a RISC-V Based Desktop Computer. Proceedings of the 18th International Symposium on Open Collaboration 2022. https://doi.org/10.1145/3555051.3555065.
59. Zaruba, F.; Benini, L. The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-Nm FDSOI Technology. IEEE Trans. VLSI Syst. 2019, 27, 2629–2640. https://doi.org/10.1109/TVLSI.2019.2926114.
60. Miura, J.; Miyazaki, H.; Kise, K. A Portable and Linux Capable RISC-V Computer System in Verilog HDL 2020. https://doi.org/10.48550/arXiv.2002.03576.
61. GitHub.com. Available online: https://github.com/carlosedp/riscv-bringup/releases/download/v1.0/kubernetes_1.16.0_riscv64.deb (accessed on 09 Aug 2024.).
62. GitHub.com. Available online: https://github.com/carlosedp/riscv-bringup/blob/master/kubernetes/Readme.md (accessed on 09 Aug 2024.).
63. GitHub.com. Available online: https://github.com/vEddYcro/HPCC-HetComp (accessed on 10 Aug 2024.).
64. Kovač, M.; Dragić, L.; Malnar, B.; Minervini, F.; Palomar, O.; Rojas, C.; Olivieri, M.; Knezović, J.; Kovač, M. FAUST: Design and Implementation of a Pipelined RISC-V Vector Floating-Point Unit. Microprocessors and Microsystems 2023, 97, 104762. https://doi.org/10.1016/j.micpro.2023.104762.
65. Dakić, V.; Kovač, M.; Slovinac, J. Evolving High-Performance Computing Data Centers with Kubernetes, Performance Analysis, and Dynamic Workload Placement Based on Machine Learning Scheduling. Electronics 2024, 13, 2651. https://doi.org/10.3390/electronics13132651.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.