A Comparative Analysis of Kubernetes and OpenShift based on Workloads using Different Hardware Architecture

MSc Research Project
Cloud Computing

Anant Sakharam Pednekar
Student ID: 21188947

School of Computing
National College of Ireland

Supervisor: Rejwanul Haque

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Anant Sakharam Pednekar
Student ID:	21188947
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Rejwanul Haque
Submission Due Date:	15/12/2023
Project Title:	A Comparative Analysis of Kubernetes and OpenShift based on Workloads using Different Hardware Architecture
Word Count:	XXX
Page Count:	15

Signature:	
Date:	29th January 2024

A Comparative Analysis of Kubernetes and OpenShift based on Workloads using Different Hardware Architecture

Anant Sakharam Pednekar
21188947

Abstract

The application deployment and management landscape has undergone a transformative shift with containerization, spearheaded by prominent platforms like Kubernetes and OpenShift. Despite their widespread adoption, a crucial knowledge gap persists in understanding the performance nuances and optimization strategies when deployed on diverse hardware architectures, encompassing x86, IBM ppc64le, and ARM.

This research attempts to bridge this knowledge gap by conducting thorough benchmarking and optimization tests tailored for Kubernetes on x86 and ARM and OpenShift on ppc64le platforms. The evaluation encompasses critical performance metrics, including CPU and memory utilization and storage performance.

The outcomes of this research are poised to empower businesses and researchers in selecting containerization engines that align with their unique hardware and software requirements. The study aims to improve application performance and operational efficiency in containerized environments by enhancing our understanding of the performance across diverse architectures.

The evaluations reveal significant findings from the Sysbench and Kbench tests, showcasing notable differences in performance metrics between x86 and ARM platforms. This research highlights the positive aspects of ARM architecture in terms of CPU speed and efficiency, making it the preferred choice for high-CPU-usage tasks. It also emphasizes the importance of considering workload-specific factors when deciding between x86 and ARM. In addition, the study suggests using OpenShift on ppc64le for situations where performance requirements perfectly match workload needs. These findings offer valuable insights for making informed decisions when choosing the most appropriate hardware architecture for containerized applications.

1 Introduction

Over the past few years, the methodology used to develop and deploy cloud computing applications has experienced a notable shift with the growing use of containerization. Unlike traditional virtualization methods that use Virtual Machines (VMs), containers provide a separate runtime environment for running applications directly on the host operating system (OS). This virtualization method improves efficiency and reduces resource usage.(Felter et al.; 2015; Joy; 2015)

The inherent benefits of containerization, including improved scalability and portability, have significantly contributed to its extensive adoption in various industries. The increasing adoption of containers requires strong management tools to handle deployment, scalability, and overall operations effectively. This feature is especially noticeable in extensive implementations, as demonstrated by Google's weekly management of billions of containers using Kubernetes, an internally developed open-source tool for container orchestration. As a result, the container management industry is experiencing significant growth and intense competition. [1] [2]

Docker is widely recognized as the container runtime, while Kubernetes is widely adopted as the leading container orchestration solution, setting industry standards.

This study aims to analyze the performance characteristics of Kubernetes and OpenShift, which are widely used container orchestration platforms, on three different hardware architectures: x86, ARM, and IBM ppc64le. The investigation seeks to analyze the operational characteristics of these platforms on these architectures by closely examining key performance metrics such as throughput, latency, and resource utilization. This research is crucial for developers and system administrators who aim to enhance the performance of their containerized applications and make informed choices about the most appropriate platform for their specific requirements.[3]

Commencing with a comprehensive overview of Kubernetes and OpenShift and highlighting their unique characteristics, this report will explain the research design and methodology utilized to assess the effectiveness of these platforms on different hardware architectures. The performance assessments will be adjusted to consider various hardware configurations, including x86, ARM, and IBM ppc64le architectures. The provided analyses will thoroughly examine the outcomes, offering valuable insights into the performance characteristics of Kubernetes and OpenShift on these hardware platforms. The report will provide recommendations on the most suitable platform hardware combinations for specific use cases, ultimately concluding with these recommendations. The following discussion will focus on the practical implications and potential areas for future research within container orchestration tools and performance evaluation.

2 Literature Review

This literature study examines the current understanding of the performance of Kubernetes and OpenShift, two prominent container orchestration platforms, across different hardware architectures. The subsequent segment encompasses an evaluation of prior literature.

2.1 Historical Context

The increasing use of containers and container management frameworks has led to increased research and scholarly attention on evaluating these technologies. Container-based systems were initially set up as faster options compared to hypervisor-based sys-

[1]Containers at Google: `https://cloud.google.com/containers/`

[2]Kubernetes: `https://kubernetes.io/`

[3]Pure Storage 2021 Kubernetes Adoption Survey: `https://www.purestorage.com/content/dam/pdf/en/analyst-reports/ar-portworx-pure-storage-2021-kubernetes-adoption-survey.pdf`

tems. However, introducing Docker has increased curiosity about these solutions. Prior studies have demonstrated that containers offer a more efficient and streamlined alternative to conventional hypervisor-based virtualization.(Merkel et al.; 2014)

The study by Felter et al. (2015) aimed to evaluate the CPU, memory, storage, and network overheads of Docker containers compared to KVM hypervisors. The findings demonstrated the greater efficiency of containers compared to virtual machines (VMs) in many circumstances. Joy (2015) conducted an analogous investigation centered on container efficiency, assessing the operational capability of a front-end application server implemented on a container instead of a VM. The results revealed a notable superiority of Docker containers compared to VMs.

Although containers provide a lightweight and effective method for packaging and deploying software, handling them on a large scale presents difficulties. To tackle this issue, container orchestration tools streamline deploying, scaling, and managing containers, simplifying the management of large-scale containerized systems.

Kubernetes, a widely acclaimed container orchestration technology, is renowned for its exceptional stability, remarkable scalability, and impressive fault tolerance. Nevertheless, the functioning of it may necessitate significant resource allocations. Researchers have created more lightweight alternatives, such as MicroK8s and K3S, to address this issue. These technologies strive to accelerate the process of deploying and providing support. However, they may not achieve the same level of performance as Kubernetes. Telenyk et al. (2021) conducted a performance evaluation of Kubernetes and MicroK8s/K3S, comparing their resource utilization, cluster startup speed, and time required for orchestration activities. Although Kubernetes generally performed better than MicroK8s/K3S, K3S showed superior disc utilization. The testing revealed that MicroK8s had subpar performance, emphasizing the benefits of using lightweight platforms in situations with few resources.

Selecting a container orchestration tool requires careful evaluation of the application's requirements. The study conducted by Koziolek and Eskandani (2023) in 2023 examined four lightweight Kubernetes (K8s) distributions, focusing on resource utilization, control plane throughput, and data plane performance. The findings revealed a somewhat greater capacity for managing control plane traffic in k3s and k8s, yet MicroShift exhibited superior performance in handling data plane traffic. Nevertheless, the decision-making process should also consider considerations such as usability, security, and maintainability. The results of this study provide helpful guidance for professionals in choosing the most appropriate distribution for their unique requirements.

Evaluating the performance of various orchestration technologies is crucial for identifying the most suitable option for specific applications. Medel et al. (2016) researched to develop a Reference net-based model for Kubernetes resource management. This model, constructed using real Kubernetes deployment data, guides the creation of scalable applications based on Kubernetes. Grounded in the concept of a reference net representing system interactions, the model was constructed through a benchmark-driven analysis of Kubernetes' deployment behavior.

In a study by Reddy et al. (2022), the efficiency of Kubernetes clusters running on

OpenStack, VMs, and bare metal was examined. The research compared CPU and memory utilization of the clusters using a sample program with a microservices architecture. The study revealed that bare-metal deployment outperformed others for computational and memory-intensive applications. While VMs and OpenStack performed similarly for computationally intensive tasks, OpenStack excelled in memory-intensive workloads. The study emphasized the need for a tailored deployment strategy based on the specific requirements of the application and its operating context.

Pereira Ferreira and Sinnott (2019) evaluated the performance of containers on managed Kubernetes services from Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). The study assessed container performance regarding CPU, memory, disk, and network using benchmarking tools. Findings indicated performance variations based on the cloud provider and resources used. AWS and NeCTAR performed best for CPU-intensive workloads, while GCP excelled in network performance. The study emphasized the absence of a one-size-fits-all solution, with the optimal managed Kubernetes service dependent on the application's specific requirements and the chosen cloud provider.

Examining hardware considerations, Gupta and Sharma (2021) conducted a recent study comparing two prevalent CPU architectures: ARM and x86. ARM processors, optimized for low power consumption, are increasingly powerful and employed in devices like laptops, desktop computers, and servers. The authors predict ARM's dominance due to rising demand for mobile devices, cloud computing popularity, and advancements in ARM technology. This transition will significantly impact the software development industry, potentially leading to new programming languages and frameworks optimized for ARM processors.

Recent research also compared x86 and IBM POWER processor architectures. While x86 is common in PCs and servers, IBM POWER is used in high-performance computing (HPC) systems and is known for scalability and performance with demanding workloads. With a power-efficient design, IBM POWER systems outperform x86 in HPC benchmarks due to their complexity-optimized architecture. Despite being costlier, IBM POWER systems support different operating systems and applications on the same hardware, offering savings. These systems also outperform x86 in terms of energy efficiency. Nebula's findings indicate that IBM POWER8 CPUs surpass x86 processors in HPCC and SPECint benchmarks, emphasizing better scalability, performance, and energy efficiency with IBM POWER. [4] [5] [6]

2.2 Related Work

Moreover, conducting a comprehensive comparative analysis of orchestration tools is of most significance, considering their unique capabilities and limitations. Determining the optimal tool for a given application is based on specific requirements.

[4]IBM POWER vs. x86 (Precisely Blog): `https://www.precisely.com/blog/data-availability/ibm-power-vs-x86-the-key-differences`

[5]Data Center Efficiency (IBM EX5 BladeCenter): `https://www.intel.ie/content/dam/doc/white-paper/data-center-efficiency-ibm-ex5-bladecenter-paper.pdf`

[6]IBM Power8 Outperforms X86 on STAC Benchmarks: `https://www.hpcwire.com/2015/06/09/ibm-power8-outperforms-x86-on-stac-benchmarks`

In their work, Pan et al. (2019) conducted a thorough comparative analysis of the current container orchestration platforms, Kubernetes and Docker in Swarm mode. This scholarly attempt aimed to identify each platform's distinct advantages and potential drawbacks, shedding light on the complex functioning details. By employing popular benchmarking methodologies, the researchers comprehensively assessed the performance overheads associated with container orchestration tools, thereby identifying their inherent advantages and disadvantages. The empirical findings suggest that Kubernetes demonstrated marginally diminished efficiency compared to Docker in Swarm mode. Although Docker in Swarm mode may exhibit reduced flexibility and strength compared to Kubernetes in detailed scenarios, both platforms can administer container clusters and similar services proficiently, employing distinct technical methodologies. The authors have also demonstrated the potential variability in the effectiveness of cloud services based on Kubernetes offered by various providers, emphasizing the importance of additional research in this domain.

In a comprehensive investigation conducted by Malviya and Dwivedi (2022), a comparative analysis was performed on four prominent container orchestration tools: Kubernetes, Docker Swarm, Mesos, and Redhat OpenShift. The evaluation was based on crucial parameters encompassing security, deployment efficiency, stability, scalability, cluster installation proficiency, and the learning curve associated with each tool. Kubernetes has demonstrated exceptional proficiency in its scheduling capabilities, showcasing its ability to allocate and manage resources for containerized applications efficiently. On the other hand, Docker Swarm has garnered recognition for its user-friendly interface and intuitive features, making it accessible and easy for individuals with varying technical expertise. The Mesos framework has exhibited commendable scalability, showcasing its ability to handle increasing workloads and resource demands efficiently. In contrast, OpenShift has demonstrated remarkable prowess in security, boasting advanced capabilities that surpass conventional standards. Notwithstanding these variances, Kubernetes has emerged as the most popular platform. The research underscored the significance of harmonizing the selection of a container orchestration tool with the organization's unique requirements. In the pursuit of expanding the scope of evaluation criteria, it becomes imperative to meticulously deliberate upon the efficacy of these tools in diverse hardware architectures. This study aims to expand upon existing knowledge by conducting a comparative analysis of Kubernetes and OpenShift on various hardware architectures and workloads. Through this investigation, we intend to provide a comprehensive understanding of the two platforms, enabling us to make informed recommendations for their deployment in diverse environments.

The scholarly works of Aly et al. (2018) and Jakkula (2019) encompassed meticulous examinations that sought to compare the operational efficiency of Kubernetes and OpenShift. Their findings unveiled noteworthy disparities contingent upon the nature of the workload and the underlying hardware architecture. In their study,Aly et al. (2018) conducted an empirical investigation to compare the performance of OpenShift and Kubernetes in distinct workloads on bare metal infrastructure. The findings revealed that OpenShift performed superior in certain workloads, surpassing Kubernetes. However, in scale-out scenarios, Kubernetes showcased enhanced efficiency. These results shed light on the nuanced performance characteristics of these two prominent container orchestration platforms, providing valuable insights for practitioners and researchers in the field.

In contrast, Jakkula conducted an empirical analysis wherein Kubernetes demonstrated superior performance in handling CPU-intensive workloads, whereas OpenShift exhibited exceptional proficiency in managing memory-intensive tasks. Both investigations highlighted the inherent variability in operational effectiveness, contingent upon distinct workloads and hardware architectures, thereby emphasizing the imperative for additional scholarly inquiry to grasp the determinants that impact platform performance comprehensively.

Given the extensive adoption of these tools and the apparent differences in their performance on diverse hardware architectures, a noticeable gap exists in the scholarly discussions regarding an organized evaluation of their performance attributes. Previous evaluations have utilized various benchmarking techniques, while my approach incorporates the evaluation of performance overheads and a comprehensive cross-comparison across all hardware architectures. The current investigation aims to provide a comprehensive understanding of the methodologies, which will be discussed in the following section. This study outlines an elaborate process for evaluating and comparing the performance of Kubernetes and OpenShift on various hardware architectures, including x86, ARM, and IBM Power.

3 Research Methods and Specifications

Running benchmarking for Kubernetes and OpenShift on various hardware architectures requires suitable tools. Employing the correct approach, the setup procedure can be efficiently controlled, enhancing the accuracy of the generated outcomes. This section offers a comprehensive summary of the Kubernetes and OpenShift benchmarking process, covering everything from the initial input to the final results.

3.1 Performance Metrics

To comprehensively analyze and assess container orchestration tools like Kubernetes and OpenShift, tracking performance metrics that directly influence their efficiency and scalability is imperative. In this benchmarking process, I will employ two distinct metrics: control plane/master metrics and data plane/worker metrics. These metrics encompass four crucial dimensions: CPU, Memory, Disk, and Network.

The CPU metric gauges the central processing unit's utilization, providing insights into the container orchestration system's efficiency in executing computational tasks. Simultaneously, the Memory metric assesses memory consumption, which is pivotal in evaluating the system's adeptness in managing memory resources, mitigating memory leaks, and optimizing overall performance. The Disk metric focuses on storage performance, ensuring the container orchestration utility adeptly executes read and write operations to the storage medium. This evaluation guarantees the system's seamless interaction with storage resources. Lastly, the Network metric delves into networking capabilities, encompassing data transmission rates and latency. These parameters directly influence container communication and coordination, thereby impacting the overall effectiveness of the container orchestration system.

The performance evaluation of container orchestration tools will be conducted using a carefully selected set of open-source benchmarking tools. Different tools exert different stress levels on the system's resources, allowing for observing container performance in various conditions. Incorporating control plane/master metrics and data-plane/worker metrics guarantees a thorough evaluation from various viewpoints, bolstering the strength and credibility of the assessment outcomes. Table 1 presents a concise summary of the chosen software, including their versions and the metrics they track. This robust benchmarking methodology enables an unbiased and comprehensive assessment of container orchestration systems.

Resource	Tools	Version
CPU	Sysbench	1.0.2
CPU	K-bench	Master
Memory	Sysbench	1.0.2
Memory	K-bench	Master

Table 1: Benchmarking tools and Metrics

benchmark-operator: The benchmark operator is a powerful tool specifically developed to evaluate the performance of Kubernetes clusters. This tool is capable of running a diverse range of micro-benchmarks, including fio and uperf, as well as application benchmarks like YCSB and pgbench. Moreover, the benchmark operator enables users to do benchmarks using tools such as iperf3, sysbench, and byowl, offering useful insights into the performance characteristics of the Kubernetes cluster. It simplifies obtaining benchmark data for cluster classification, assisting in making data-driven decisions regarding distribution, platform deployment, and storage provisioning. The software provides a wide range of benchmarking features, allowing users to assess the performance of networks and storage systems. This data can then be used to make well-informed decisions.[7]

Sysbench: Sysbench, a crucial tool in this study, is a flexible software for CPU testing. Sysbench, which consists of many modules, allows for a comprehensive assessment of different operating system resources. The study utilizes Sysbench to evaluate the CPU and disc performance in the systems under consideration. The tool's main purpose is to assess and compare solutions by emulating the demanding functions of a database management system, eliminating the requirement for separate installations and configurations. Sysbench operates in a manner that is not dependent on any specific platform and can handle several threads simultaneously. It is particularly effective in doing calculations for prime numbers, which serves as demanding work for measuring the performance of a CPU. The Sysbench CPU test precisely measures the performance of a CPU by calculating prime numbers within a given range. It provides a scalable and accurate assessment of the CPU's capabilities, considering the severity of the benchmark. [8]

K-Bench: K-Bench is a complete framework specifically developed to evaluate and compare the performance of the control and data plane components in a Kubernetes environment. K-Bench offers a customizable method for users to construct and control Kubernetes resources on a large scale systematically. It also provides performance data

[7]Benchmark-Operator: `https://github.com/cloud-bulldozer/benchmark-operator`

[8]Scriptable database and system performance benchmark (sysbench): `https://github.com/akopytov/sysbench`

that are relevant to the target architecture. The framework facilitates various activities on Kubernetes resources, including Pod, Deployment, Service, and ReplicationController. This empowers users to manage client-side concurrency, establish workflows, and set parameters for different operations. K-Bench demonstrates exceptional proficiency in evaluating control-plane performance by combining server-side timing with a client-side method, utilising Kubernetes event callback system to achieve greater accuracy. K-Bench also provides integrated benchmarks for analysing data plane performance, optimising infrastructure resource utilisation, and facilitating the conversion of Docker Compose files into Kubernetes specification files. K-Bench is a versatile framework for conducting thorough performance evaluations of Kubernetes. It can be seamlessly integrated with monitoring tools such as Prometheus and Wavefront. [9]

4 Design Specification

This section outlines the design specs, providing detailed information about software and hardware components. The topic covers essential aspects such as software versions, components, and hardware infrastructure, including VMs and architecture. The goal is to provide a concise yet comprehensive explanation of the system utilized in these benchmarks.

4.1 Hardware Specification

The benchmarks were conducted on three distinct systems, each with specific configurations:

x86 System:

- *Instance Type: t2.2xlarge,*
- vCPUs: 8,
- *Memory: 32 GiB,*
- Architecture: 64-bit (x86)

ARM System:

- *Instance Type: t4g.2xlarge,*
- vCPUs: 8,
- *Memory: 32 GiB,*
- Architecture: 64-bit (Arm)

PPC64LE System:

- *Architecture: ppc64le,*
- CPU(s): 160,
- *Cores per Socket: 20,*

[9]vmware-tanzu-k-bench: `https://github.com/vmware-tanzu/k-bench`

- Threads per Core: 4,
- *Sockets: 2,*
- Model: POWER9

4.2 Software Specification

This section describes the details of the software used

x86 System:

- *Operating System: Canonical, Ubuntu Server 22.04 LTS (HVM),*
- Kubernetes Version: 1.26.10,
- *Master Nodes: 1,*
- Worker Nodes: 1

ARM System:

- *Operating System: Canonical, Ubuntu Server 22.04 LTS (HVM),*
- Kubernetes Version: 1.26.10,
- *Master Nodes: 1,*
- Worker Nodes: 1

ppc64le System:

- *Operating System: RHEL 9,*
- Openshift Version: 4.11,
- *Master Nodes: 3,*
- Worker Nodes: 3

benchmarking tools

- *K-bench: Master,*
- Benchmark Operator (sysbench): 1.0.2

5 Implementation

Over the implementation phase, AWS Cloud EC2 instances served as Kubernetes nodes for x86 and ARM architectures. Based on the instructions provided in the article by Kawonise and Olorunfemi [10], a Kubernetes cluster consisting of two EC2 instances was created. Specifically, the t2.2xlarge instance for x86 architecture and the t4g.2xlarge

[10]Simplifying Kubernetes Installation on Ubuntu using a Bash Shell Script: `https://medium.com/@olorunfemikawonise_56441/simplifying-kubernetes-installation-on-ubuntu-using-a-bash-shell-script-d75fed68a31`

instance for ARM architecture were used. Every Kubernetes cluster comprises a single instance designated as the master node and another as the worker node. In contrast, OpenShift, a proprietary software solution, employed a dedicated cluster hosted by IBM to conduct the tests. The cluster consisted of three master nodes and three worker nodes.

The K-bench tool evaluated the performance of the container platforms by executing three different test profiles, namely dp_redis, dp_redis_service, and cp_heavy_12client. The purpose of these test profiles is to assess the CPU and memory efficiency by examining the delays caused by scheduling pods in Kubernetes and OpenShift systems. The benchmark operator utilized Sysbench to conduct different test profiles on x86 and ARM architectures. It is important to mention that Sysbench is currently unavailable for use on the IBM Power architecture. The K-bench tool enabled an in-depth review of performance indicators across all three hardware platforms, providing insights into the effects of scheduling and resource utilization in these containerized systems.

6 Evaluation and discussion

This section provides an in-depth examination of the experimental results, illustrating the importance and impact of the Sysbench and Kbench tests. The main objective is to clarify key findings based on statistical analysis and visual representations. Refer to Tables 6.1 ,6.1, 4.

6.1 Benchmark-Operator: Sysbench Analysis

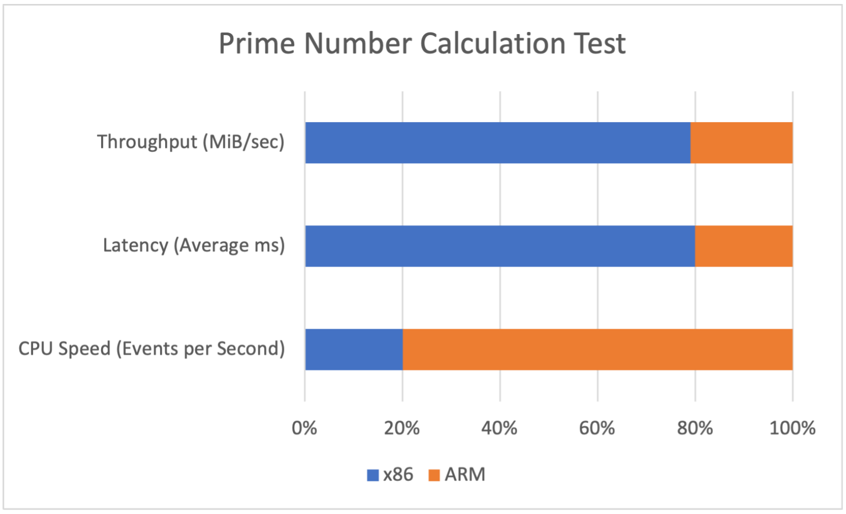

Figure 1: Prime Number Calculation Test

ARM exhibited substantial performance superiority over x86, achieving a rate of 31507.94 events per second, while x86 only managed 7937.13. The outcome, in conjunction with CPU speeds and latency measurements, highlights the exceptional processing capability of ARM. Refer to Table 6.1 and Figure 1.

ARM showcased its efficiency in latency measurements, boasting an average latency of 0.03 milliseconds, which stands in sharp contrast to x86's 0.12 milliseconds. Con-

Metric	x86	ARM
CPU Speed (Events per Second)	7937.13	31507.94
Latency (Average ms)	0.12	0.03
Throughput (MiB/sec)	116.86	31.08

Table 2: Performance Metrics from Prime Number Calculation Test

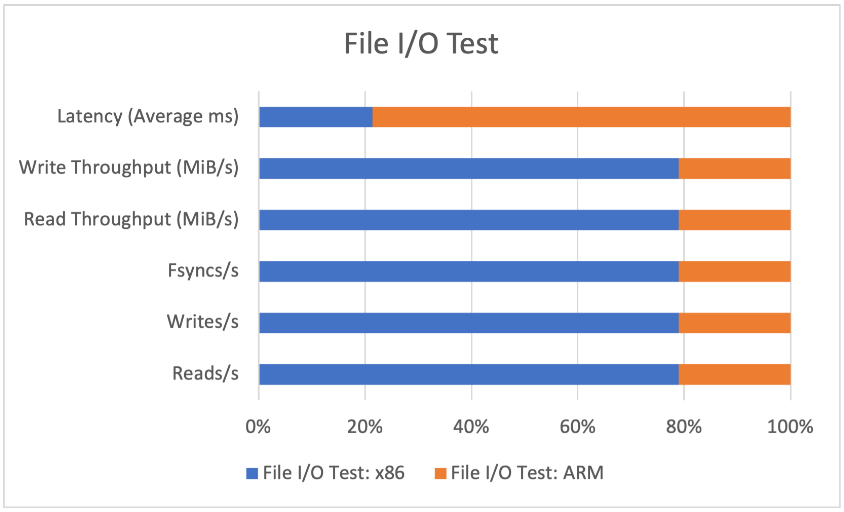

Figure 2: File I/O Test

sequently, ARM demonstrates outstanding effectiveness in executing tasks quickly and responsively, rendering it a desirable alternative for applications that demand low latency in performance.

Metric	x86	ARM
Reads/s	946.79	252.23
Writes/s	631.19	168.15
Fsyncs/s	2029.11	541.6
Read Throughput (MiB/s)	14.79	3.94
Write Throughput (MiB/s)	9.86	2.63
Latency (Average ms)	0.28	1.03

Table 3: Performance Metrics from File I/O Test

Analyzing the File I/O test in depth exposes small differences between x86 and ARM. The throughput and latency displayed by ARM were lower, although these differences could change depending on the situation, affecting the system's overall performance. Refer to Table 6.1 and Figure 2.

6.2 K-bench Analysis

In the context of cp_heavy_12client, the performance of ARM exhibited a striking resemblance to that of x86 and ppc64le in terms of pod creation throughput and average latency. This observation implies a similar level of effectiveness in managing demanding computational tasks among these different architectures. Refer to Table 4 and Figure 3.

Kbench Tests			
Metric	**x86**	**ARM**	**ppc64le**
cp_heavy_12client_Metric			
Pod Creation Throughput	113.23	110.88	108.42
Pod Creation Average Latency	2.93	2.96	2.95
Pod Startup Latency (Median)	3356.59	3356.59	3356.59
dp_redis_kbench_Metric			
Pod Creation Throughput	8.2	7.66	1.53
Pod Creation Average Latency	7.32	7.83	39.12
Pod Startup Latency (Median)	N/A	N/A	N/A
dp_redis_service_kbench_Metric			
Pod Creation Throughput	4.37	5.02	2.77
Pod Creation Average Latency	13.71	11.96	21.68
Pod Startup Latency (Median)	N/A	N/A	N/A

Table 4: Performance Metrics from Kbench Tests

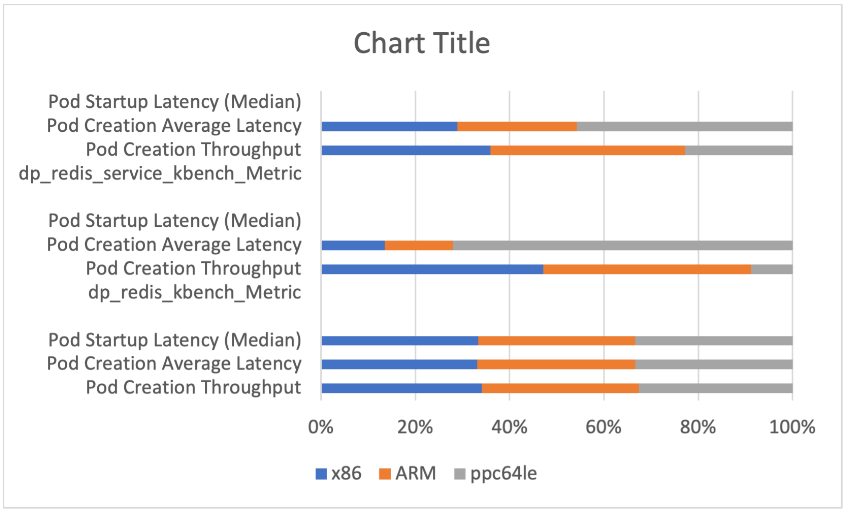

Figure 3: Kbench Results

In the case of dp_redis_kbench, ARM impressively exhibited its prowess in generating pods with remarkable throughput, exemplifying its ability to manage tasks that demand substantial data processing capabilities efficiently. Nevertheless, this development had its drawbacks, as it resulted in an increase in average latency compared to x86. This suggests that there may be certain trade-offs in latency inherent to the ARM architecture.

Within the context of dp_redis_service_kbench, ARM demonstrated a notable increase in the rate at which pods are created, indicating a significant performance improvement, specifically for service-oriented workloads. However, this enhancement was offset by an increase in average latency compared to x86 architecture. Regrettably, the unavailability of median pod startup latency data impairs an accurate assessment of the startup efficiency of this particular benchmark on ARM architecture.

6.3 Discussion

In the case of system selection for specific jobs, various considerations arise. The ARM architecture is the more suitable selection for computationally demanding tasks owing to its inherent ability to execute high-speed and efficient processing operations. In situations where achieving optimal low-latency performance is of utmost importance, the distinct advantage of ARM architecture in substantially mitigating average latency positions it as the preferred choice. Implementing file input/output (I/O) operations presents a complex decision-making framework that demands a cautious assessment of the particular needs. The selection between x86 and ARM architectures requires a delicate balance between throughput and latency, requiring thoughtful consideration of trade-offs.

The selection of system architecture within the Kubernetes platform, deployed on both x86 and ARM, as well as OpenShift, which utilizes ppc64le, is predicated on the workload's distinct requirements. Kubernetes deployments on both x86 and ARM architectures present an optimal solution for applications that demand enhanced CPU performance and heightened responsiveness. In addition, the ARM architecture presents a promising catalyst for optimizing performance in specific workloads within this framework. On the other hand, using OpenShift on ppc64le is particularly suitable for situations in which the inherent performance attributes of ppc64le harmoniously correspond to the demands of the workload. Nevertheless, it is imperative to use cautious discussion when confronted with instances of increased pod starting latency.

7 Conclusion and Future Work

7.1 Conclusion

This evaluation of the ARM, x86, and ppc64le architectures using Sysbench and Kbench tests provides valuable insights into their performance characteristics. The findings demonstrate that ARM outperforms CPU speed and efficiency, positioning it as the preferred option for tasks requiring high CPU usage. Moreover, its superior average latency makes it highly suitable for low-latency performance applications. Workload requirements should determine the selection of x86 or ARM due to subtle variations in File I/O operations.

The choice of system architecture in Kubernetes (on x86 and ARM) and OpenShift (on ppc64le) should be based on the specific requirements of the workload. Kubernetes is suitable for applications that demand strong CPU performance and responsiveness. It works well on both x86 and ARM architectures, with ARM potentially providing better efficiency for certain workloads. OpenShift on ppc64le is ideal for situations where performance requirements match seamlessly with workload needs, but caution is needed for cases with higher pod startup latency.

7.2 Future Work

For future research, it is imperative to ensure a more standardized environment across all architectures by using similar hardware nodes and configurations. This would involve employing homogeneous hardware specifications to eliminate potential biases introduced by diverse hardware setups. Furthermore, utilizing a common benchmarking tool supported uniformly across all platforms would enhance the comparability and reliability of results. This would facilitate a more robust evaluation of each architecture's capabilities and provide a clearer basis for decision-making in system selection.

Additionally, to further enhance the generalizability of findings, future work should involve deploying the same workload in identical environments across x86, ARM, and ppc64le architectures. This approach ensures a fair and direct comparison, enabling a more accurate assessment of each architecture's performance under consistent conditions.

By addressing these considerations in future research efforts, we can refine our understanding of the strengths and limitations of each architecture, providing more subtle insights for informed decision-making in diverse computing environments.

References

Aly, M., Khomh, F. and Yacout, S. (2018). Kubernetes or openshift? which technology best suits eclipse hono iot deployments, *2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA)*, pp. 113–120.

Felter, W., Ferreira, A., Rajamony, R. and Rubio, J. (2015). An updated performance comparison of virtual machines and linux containers, *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 171–172.

Gupta, K. and Sharma, T. (2021). Changing trends in computer architecture: A comprehensive analysis of arm and x86 processors, *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT)* **7**(3): 619–631. Published in Volume 7 — Issue 3 — May-June 2021. Date of Publication: 2021-06-30. Manuscript Number: CSEIT2173188. Page(s): 619-631.

Jakkula, P. (2019). Container runtime performance evaluation of kubernetes and openshift, *Researchgate*.

Joy, A. M. (2015). Performance comparison between linux containers and virtual machines, *2015 International Conference on Advances in Computer Engineering and Applications*, pp. 342–346.

Koziolek, H. and Eskandani, N. (2023). Lightweight kubernetes distributions: A performance comparison of microk8s, k3s, k0s, and microshift, *ICPE '23: Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering*, ICPE '23, Association for Computing Machinery, New York, NY, USA, p. 17–29.

Malviya, A. and Dwivedi, R. K. (2022). A comparative analysis of container orchestration tools in cloud computing, *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 698–703.

Medel, V., Rana, O., Banares, J. A. and Arronategui, U. (2016). Modelling performance & resource management in kubernetes, *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, pp. 257–262.

Merkel, D. et al. (2014). Docker: lightweight linux containers for consistent development and deployment, *Linux Journal* **239**(2): 2.

Pan, Y., Chen, I., Brasileiro, F., Jayaputera, G. and Sinnott, R. (2019). A performance comparison of cloud-based container orchestration tools, *2019 IEEE International Conference on Big Knowledge (ICBK)*, pp. 191–198.

Pereira Ferreira, A. and Sinnott, R. (2019). A performance evaluation of containers running on managed kubernetes services, *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 199–208.

Reddy, Y. S. D., Reddy, P. S., Ganesan, N. and Thangaraju, B. (2022). Performance study of kubernetes cluster deployed on openstack,vms and baremetal, *2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pp. 1–5.

Telenyk, S., Sopov, O., Zharikov, E. and Nowakowski, G. (2021). A comparison of kubernetes and kubernetes-compatible platforms, *2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Vol. 1, pp. 313–317.