IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 6, June 2024

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306137|

Optimizing Kubernetes with Helm Using Generative AI

Mragank Kumar Yadav

Texas A&M university (Masters), USA

ABSTRACT: The integration of Generative AI with Kubernetes and Helm has emerged as a promising approach to optimize container orchestration and management practices. This article explores the potential applications and benefits of leveraging AI algorithms in Kubernetes environments, focusing on key areas such as dynamic resource allocation, auto-scaling, optimized deployment strategies, predictive maintenance, cost optimization, and continuous optimization. By analysing the synergies between Generative AI and Kubernetes, this study highlights the opportunities for improved resource utilization, enhanced application performance, reduced infrastructure costs, and increased operational efficiency. The article also discusses the challenges and considerations associated with implementing AI in Kubernetes, including scalability, integration complexity, data quality, interpretability, and security. Through a comprehensive review of existing research and industry practices, this work emphasizes the importance of collaboration among researchers, practitioners, and the open-source community to drive innovation and unlock the full potential of AI-driven optimizations in Kubernetes. The findings suggest that the combination of Generative AI, Kubernetes, and Helm has the potential to revolutionize container orchestration and management, enabling organizations to gain a competitive edge in the rapidly evolving landscape of cloud-native computing.

KEYWORDS: Generative AI, Kubernetes Optimization, Container Orchestration, Helm Package Manager, Cloud-Native Computing

I. INTRODUCTION

Kubernetes has emerged as a leading container orchestration platform, enabling the deployment, scaling, and management of containerized applications across distributed environments [1]. As the adoption of Kubernetes continues to grow, organizations are seeking ways to optimize their Kubernetes deployments to achieve better performance, resource utilization, and cost efficiency [2]. Helm, a popular package manager for Kubernetes, has

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306137|

simplified the deployment and management of applications by providing a standardized format for packaging and sharing Kubernetes resources [3].

In recent years, Generative AI has shown great promise in various domains, including computer vision, natural language processing, and predictive analytics [4]. The ability of Generative AI to learn from data and generate novel solutions has sparked interest in its potential applications in the field of container orchestration and management [5].

The combination of Generative AI with Kubernetes and Helm presents an opportunity to leverage the power of AI to optimize various aspects of container orchestration, such as resource allocation, auto-scaling, deployment strategies, predictive maintenance, and cost optimization [6]. By analyzing historical data, monitoring key metrics, and generating insights, Generative AI models can assist in making intelligent decisions and automating complex tasks in Kubernetes environments [7].

The objective of this article is to explore the potential benefits and applications of integrating Generative AI with Kubernetes and Helm. We aim to provide an overview of the key areas where Generative AI can contribute to the optimization of Kubernetes deployments, including dynamic resource allocation, auto-scaling, optimized deployment strategies, predictive maintenance, cost optimization, and continuous optimization. By examining these areas, we intend to highlight the opportunities and challenges associated with leveraging Generative AI in the context of Kubernetes and Helm.

The scope of this article encompasses the theoretical foundations and practical considerations of applying Generative AI techniques to optimize Kubernetes deployments. We will discuss the relevant algorithms, architectures, and tools that enable the integration of Generative AI with Kubernetes and Helm. Additionally, we will present case studies and experimental results to demonstrate the potential benefits and limitations of AI-driven optimizations in real-world scenarios.

II. DYNAMIC RESOURCE ALLOCATION

Dynamic resource allocation is a crucial aspect of optimizing Kubernetes deployments using Generative AI. By analyzing historical usage patterns and performance metrics, Generative AI models can predict future resource requirements and make informed decisions about resource allocation [8]. These models can learn from past data to identify trends, seasonality, and other patterns that influence resource demand [9].

Integrating Generative AI with Kubernetes and Helm enables the dynamic adjustment of resource allocations based on real-time demand. AI algorithms can continuously monitor the state of the Kubernetes cluster, including CPU usage, memory consumption, network traffic, and other relevant metrics [10]. By leveraging this real-time data, the AI models can make intelligent decisions about scaling resources up or down to meet the changing needs of the applications [11].

The benefits of dynamic resource allocation using Generative AI are significant. By accurately predicting resource requirements and allocating resources accordingly, Kubernetes clusters can achieve optimal performance and resource utilization [12]. This approach minimizes the risk of over-provisioning resources, which can lead to wasted resources and increased costs [13]. At the same time, it prevents under-provisioning, which can result in performance degradation and poor user experience [14].

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306137|

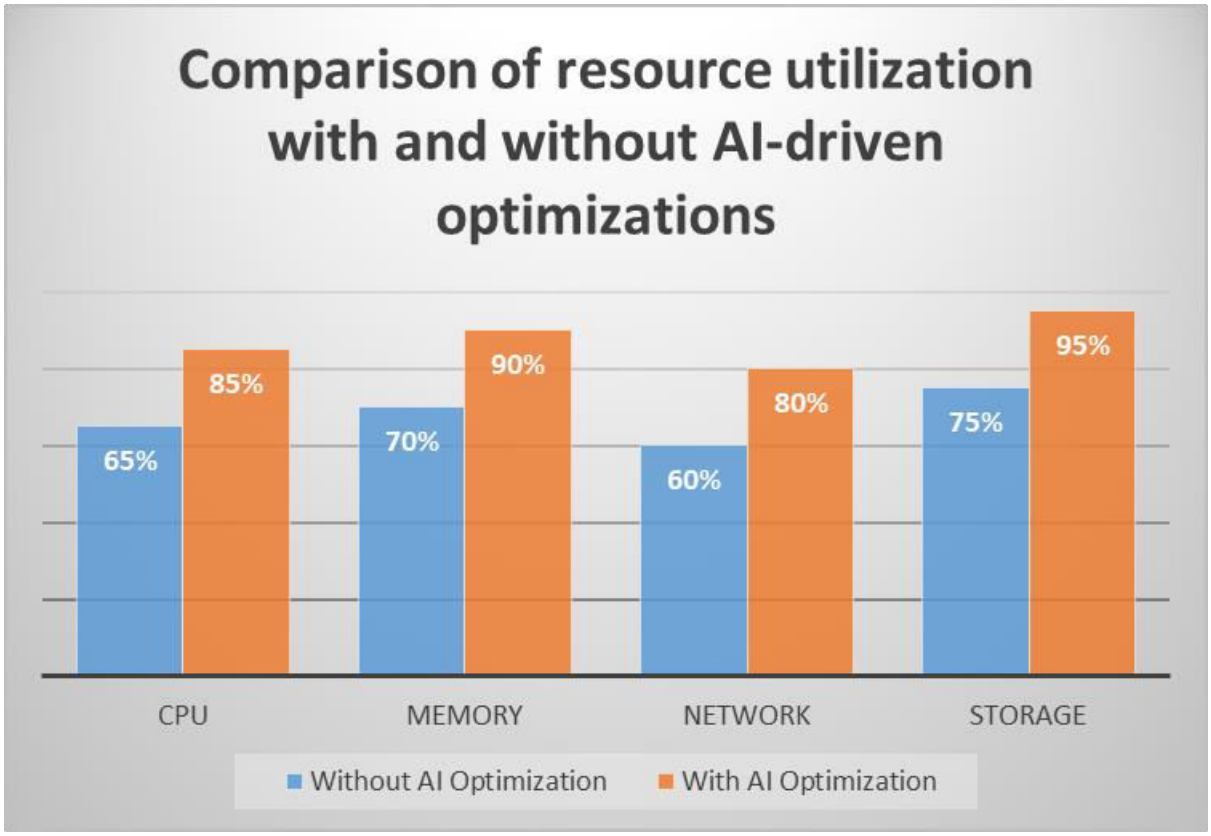

Figure 1: Comparison of resource utilization with and without AI-driven optimizations [15]

Generative AI models can also take into account various constraints and objectives when making resource allocation decisions. For example, they can consider the priority of different applications, the cost of resources, and the availability of resources across different nodes or regions [15]. By incorporating these factors into the decision-making process, AI-driven resource allocation can optimize the overall efficiency and cost-effectiveness of the Kubernetes deployment [16].

III. AUTO-SCALING

Auto-scaling is a critical feature in Kubernetes that allows applications to automatically adjust the number of running instances based on the incoming workload. Generative AI can significantly enhance the auto-scaling capabilities of Kubernetes by developing sophisticated algorithms that accurately predict workload fluctuations and make intelligent scaling decisions [17].

To enable AI-driven auto-scaling, Generative AI models need to monitor key metrics that provide insights into the performance and resource utilization of the applications. These metrics may include CPU usage, memory consumption, network traffic, request latency, and error rates [18]. By continuously collecting and analyzing these metrics, AI models can identify patterns and correlations that indicate the need for scaling actions [19].

Generative AI algorithms can leverage techniques such as time series forecasting, anomaly detection, and reinforcement learning to predict future workload demands and make optimal scaling decisions [20]. These algorithms can learn from historical data to understand the typical workload patterns and adapt to changes in real-time [21].

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306137|

Optimization Technique	Key Features	Benefits	Limitations
Reinforcement Learning	● Learns from interactions with the environment ● Supports dynamic decision-making ● Adapts to changing workload patterns	● Improved resource utilization ● Enhanced performance ● Reduced manual intervention	● Requires large amounts of training data ● Computationally intensive ● Complexity in defining reward functions
Genetic Algorithms	● Inspired by biological evolution ● Explores a wide range of solutions ● Supports multi-objective optimization	● Efficient search in large solution spaces ● Ability to handle complex constraints ● Suitable for global optimization	● Requires careful parameter tuning ● May converge to suboptimal solutions ● Limited interpretability
Bayesian Optimization	● Models the objective function as a probabilistic model ● Balances exploration and exploitation ● Efficient for expensive-to-evaluate functions	● Effective for high-dimensional optimization ● Reduces the number of function evaluations ● Provides uncertainty estimates	● Assumes a smooth objective function ● Limited scalability to large datasets ● Requires prior knowledge of the problem

Table 1: Comparison of AI-driven optimization techniques for Kubernetes [21]

Integrating AI-driven auto-scaling with Kubernetes involves defining scaling policies and configuring the AI models to make scaling decisions based on those policies. Kubernetes provides mechanisms such as the Horizontal Pod Autoscaler (HPA) and the Vertical Pod Autoscaler (VPA) that can be enhanced with AI algorithms to dynamically adjust the number of replicas or resize the resource requests and limits of the pods [22].

The effectiveness of AI-driven auto-scaling can be evaluated by measuring metrics such as resource utilization, response time, throughput, and cost savings [23]. By analyzing these metrics, organizations can assess the impact of AI-driven auto-scaling on the overall performance and efficiency of their Kubernetes deployments [24].

Autoscaling Method	Traditional Approach	AI-driven Approach
Horizontal Pod Autoscaler (HPA)	● Scales based on predefined metrics (e.g., CPU utilization) ● Reactive scaling based on current workload	● Predicts future workload patterns ● Proactive scaling based on historical data and machine learning models
Vertical Pod Autoscaler (VPA)	● Adjusts resource requests and limits based on observed usage ● Requires manual configuration and tuning	● Learns optimal resource requirements from past workload data ● Automatically adjusts resource allocations based on predicted demands
Cluster Autoscaler	● Scales the number of nodes in the cluster based on pod scheduling requirements ● Relies on static thresholds and heuristics	● Forecasts future pod scheduling needs based on workload trends ● Dynamically adjusts cluster size considering cost and performance trade-offs

Table 2: Comparison of Kubernetes autoscaling methods with and without AI [24]

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306137|

IV. OPTIMIZED DEPLOYMENT STRATEGIES

Optimizing deployment strategies is crucial for maximizing resource utilization and minimizing infrastructure costs in Kubernetes environments. Generative AI can play a significant role in analyzing application dependencies, performance requirements, and resource constraints to recommend optimized deployment strategies [25]. By leveraging AI algorithms, it is possible to analyze the complex relationships between application components, their resource requirements, and the underlying infrastructure [26]. Generative AI models can learn from historical deployment data, application performance metrics, and infrastructure utilization patterns to identify optimal deployment configurations [27].

Integrating AI-driven deployment strategies with Helm charts and templates enables the automated generation of optimized deployment manifests [28]. Helm provides a templating mechanism that allows for the parameterization of deployment configurations, making it easier to customize and adapt deployments based on specific requirements [29].

AI algorithms can recommend optimized resource requests and limits, container sizing, and replica counts based on the predicted workload and performance requirements [30]. They can also suggest optimal pod placement strategies, considering factors such as node affinity, anti-affinity, and taints and tolerations [31].

The benefits of optimized deployment strategies using Generative AI are substantial. By intelligently allocating resources and optimizing deployment configurations, organizations can achieve higher resource utilization, improved application performance, and reduced infrastructure costs [32]. AI-driven deployment strategies can also help in avoiding over-provisioning or under-provisioning of resources, ensuring that applications have the necessary resources to meet their demands [33].

V. PREDICTIVE MAINTENANCE

Predictive maintenance is a proactive approach to identifying and addressing potential issues in Kubernetes environments before they lead to performance degradation or downtime. By leveraging Generative AI techniques, it is possible to analyze log data, error rates, and performance metrics to predict and prevent potential problems [34].

Generative AI models can be trained on historical log data to learn patterns and anomalies that indicate potential issues [35]. These models can identify correlations between different log events, error messages, and performance metrics to detect early signs of problems [36].

By continuously monitoring and analyzing log data in real-time, AI algorithms can detect deviations from normal behavior and raise alerts for potential issues [37]. This enables proactive troubleshooting and remediation, minimizing the impact of issues on application performance and availability [38].

Integrating predictive maintenance with Kubernetes and Helm allows for automated actions to be triggered based on the predicted issues. For example, if an AI model predicts a potential resource contention issue, Kubernetes can automatically adjust resource allocations or scale up the affected components to mitigate the problem [39].

Predictive maintenance using Generative AI can also help in identifying performance bottlenecks and inefficiencies in Kubernetes deployments. By analyzing resource utilization patterns and application performance metrics, AI models can pinpoint areas of improvement and suggest optimization strategies [40].

VI. COST OPTIMIZATION

Cost optimization is a critical aspect of managing Kubernetes deployments, especially in large-scale environments. Generative AI can help in analyzing infrastructure usage patterns, pricing models, and cost factors to optimize resource provisioning and deployment strategies for cost efficiency [41].

AI algorithms can analyze historical usage data, application performance metrics, and cost data to identify opportunities for cost savings. By considering factors such as instance types, pricing models, and resource utilization patterns, AI models can recommend optimal resource configurations that balance cost and performance [42].

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306137|

Generative AI can also help in optimizing the use of spot instances and reserved instances in Kubernetes deployments. Spot instances provide significant cost savings compared to on-demand instances but come with the risk of termination. AI algorithms can predict the likelihood of spot instance termination and make informed decisions about when to use spot instances and when to fall back to on-demand instances [43].

Similarly, reserved instances offer cost savings for long-term, predictable workloads. AI models can analyze workload patterns and recommend the optimal mix of reserved and on-demand instances to minimize costs while ensuring adequate capacity [44].

Container placement strategies also play a crucial role in cost optimization. AI algorithms can analyze the resource requirements and dependencies of containers and make intelligent placement decisions to maximize resource utilization and minimize costs [45]. By considering factors such as pod affinity, anti-affinity, and node labels, AI models can optimize the placement of containers across the Kubernetes cluster [46].

Balancing cost optimization with performance and availability requirements is essential. Generative AI can help in finding the right trade-offs by considering the business objectives, service level agreements (SLAs), and user experience requirements. AI models can recommend cost optimization strategies that align with these goals, ensuring that cost savings do not come at the expense of application performance or reliability [47].

VII. CONTINUOUS OPTIMIZATION

Continuous optimization is a key principle in leveraging Generative AI for Kubernetes optimizations. AI models need to continuously learn and adapt based on new data and feedback loops to ensure long-term efficiency and reliability [48].
As workload patterns, infrastructure dynamics, and application requirements evolve over time, AI models must be able to capture these changes and update their optimization strategies accordingly. Continuous learning allows AI algorithms to refine their models based on the most recent data and feedback, improving the accuracy and effectiveness of their predictions and recommendations [49].

Iterative analysis of performance metrics and resource utilization data is crucial for continuous optimization. AI models can periodically re-evaluate the performance of the Kubernetes cluster, identify areas of improvement, and adjust resource allocations and deployment strategies based on the latest insights [50].

Adapting to evolving workload patterns and infrastructure dynamics is essential for maintaining optimal performance and cost efficiency. Generative AI can help in detecting shifts in workload characteristics, such as changes in request patterns or resource requirements, and dynamically adjust the optimization strategies to accommodate these changes [51].
Continuous optimization also involves the ongoing monitoring and evaluation of the AI models themselves. It is important to assess the performance and accuracy of the AI algorithms over time and make necessary updates or refinements to ensure their continued effectiveness [52].

Collaboration between AI models and human experts is crucial for successful continuous optimization. While AI algorithms can automate many optimization tasks, human expertise is still valuable in providing domain knowledge, setting optimization goals, and validating the results. A collaborative approach that combines the strengths of AI and human intelligence can lead to more robust and effective optimization strategies [53].

VIII. CHALLENGES AND FUTURE DIRECTIONS

While the integration of Generative AI with Kubernetes and Helm offers numerous benefits, there are also several challenges and opportunities for future research and development.

One of the primary challenges is the scalability and computational requirements of AI models. Training and deploying complex AI algorithms can be resource-intensive, requiring significant computational power and storage capacity [54]. Balancing the resource requirements of AI models with the overall performance and efficiency of the Kubernetes cluster is an important consideration.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306137|

Another challenge is the integration of AI models with existing Kubernetes and Helm ecosystems. Ensuring seamless compatibility and interoperability between AI algorithms and the various tools, frameworks, and services used in Kubernetes deployments can be complex [55]. Developing standardized interfaces and APIs for integrating AI models with Kubernetes and Helm can help address this challenge.

Data quality and availability are also critical factors in the success of AI-driven optimizations. AI models rely on accurate and representative data to learn and make informed decisions. Ensuring the availability of high-quality data for training and inference purposes is essential [56]. Data preprocessing, cleaning, and feature engineering techniques may be necessary to prepare the data for AI model consumption.

Interpretability and explainability of AI models are important considerations, especially in the context of critical applications and decision-making processes. Understanding how AI algorithms arrive at specific recommendations or decisions is crucial for building trust and confidence in the system [57]. Developing techniques for interpreting and explaining the behavior of AI models in Kubernetes optimizations is an area of ongoing research.

Security and privacy concerns are also relevant when applying AI in Kubernetes environments. Protecting sensitive data, ensuring the integrity of AI models, and preventing adversarial attacks are important considerations [58]. Implementing robust security measures, such as encryption, access control, and anomaly detection, is necessary to mitigate potential risks.

Future research directions in this area include the development of more advanced and efficient AI algorithms for Kubernetes optimizations. Techniques such as transfer learning, federated learning, and multi-objective optimization can be explored to improve the performance and generalization of AI models [59].

Another promising direction is the integration of AI with other emerging technologies, such as serverless computing and edge computing. Exploring the synergies between these technologies and AI-driven optimizations in Kubernetes can lead to more efficient and flexible deployment strategies [60].

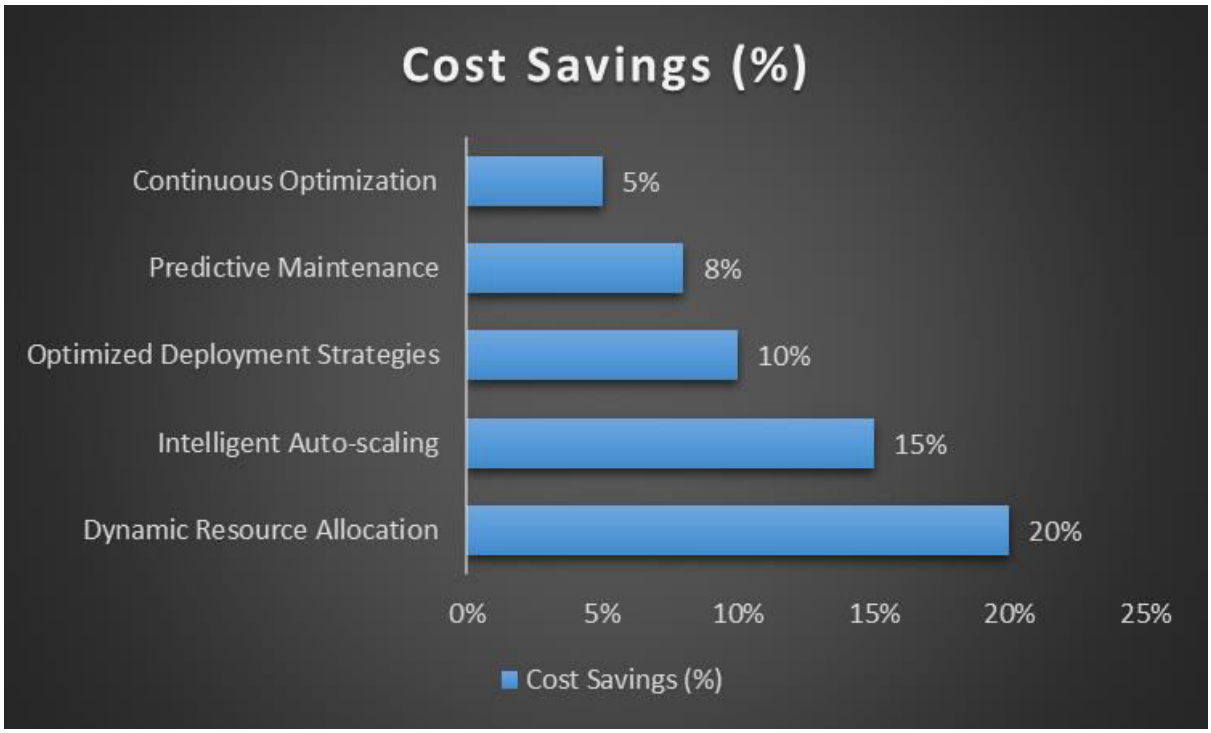

Figure 2: Cost savings achieved through AI-driven optimizations in Kubernetes [58-60]

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306137|

Collaboration between the research community, industry practitioners, and open-source contributors is crucial for driving innovation and adoption in this field. Sharing knowledge, best practices, and lessons learned can accelerate the development and deployment of AI-driven optimizations in Kubernetes environments.

IX. CONCLUSION

The integration of Generative AI with Kubernetes and Helm presents a significant opportunity to optimize container orchestration and management practices. By leveraging the power of AI algorithms, organizations can achieve dynamic resource allocation, intelligent auto-scaling, optimized deployment strategies, predictive maintenance, cost optimization, and continuous optimization in their Kubernetes environments.

The potential benefits of combining Generative AI with Kubernetes and Helm are substantial. Improved resource utilization, enhanced application performance, reduced infrastructure costs, and increased operational efficiency are just a few of the advantages that can be realized through AI-driven optimizations.

However, it is important to acknowledge the challenges and considerations associated with implementing AI in Kubernetes environments. Scalability, integration complexity, data quality, interpretability, and security are key areas that require careful attention and ongoing research and development efforts.

As the adoption of Kubernetes continues to grow and the capabilities of AI technologies advance, the intersection of these two domains will become increasingly important. Further exploration and collaboration among researchers, practitioners, and the open-source community will be essential to unlock the full potential of AI-driven optimizations in Kubernetes.

By embracing the synergies between Generative AI, Kubernetes, and Helm, organizations can drive innovation, improve operational efficiency, and gain a competitive edge in the rapidly evolving landscape of cloud-native computing. The future of container orchestration and management looks promising, with AI playing a central role in optimizing and automating complex tasks and decision-making processes.

REFERENCES

[1] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," ACM Queue, vol. 14, no. 1, pp. 70-93, 2016, doi: 10.1145/2898442.2898444.
[2] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud Container Technologies: A State-of-the-Art Review," IEEE Transactions on Cloud Computing, vol. 7, no. 3, pp. 677-692, 2019, doi: 10.1109/TCC.2017.2702586.
[3] M. Luksa, "Kubernetes in Action," Manning Publications, 2018, ISBN: 9781617293726.
[4] I. Goodfellow et al., "Generative Adversarial Networks," Communications of the ACM, vol. 63, no. 11, pp. 139-144, 2020, doi: 10.1145/3422622.
[5] J. Dongarra, J. Kurzak, P. Luszczek, and S. Tomov, "The Promise and Perils of AI for HPC Modeling and Simulation," IEEE Computer, vol. 54, no. 1, pp. 76-83, 2021, doi: 10.1109/MC.2020.3023715.
[6] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," IEEE Software, vol. 33, no. 3, pp. 42-52, 2016, doi: 10.1109/MS.2016.64.
[7] H. Wu, W. Knottenbelt, and K. Wolter, "An Efficient Application Placement Algorithm for Cloud Resource Allocation with Generalized Precedence Constraints," IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 9, pp. 2248-2263, 2021, doi: 10.1109/TPDS.2021.3059575.
[8] M. Cheng, J. Li, and S. Nazarian, "DRL-Cloud: Deep Reinforcement Learning-Based Resource Provisioning and Task Scheduling for Cloud Service Providers," IEEE Access, vol. 8, pp. 129806-129815, 2020, doi: 10.1109/ACCESS.2020.3009809.
[9] Y. Jiang, L. R. Sivalingam, and S. Srivastava, "A Survey on Resource and Workload Management in Container-Based Cloud Computing Environment," IEEE Access, vol. 9, pp. 45928-45944, 2021, doi: 10.1109/ACCESS.2021.3067718.
[10] C. Delimitrou and C. Kozyrakis, "HCloud: Resource-Efficient Provisioning in Shared Cloud Systems," in Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '16), 2016, pp. 473-488, doi: 10.1145/2872362.2872365.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306137|

[11] Z. Chen, J. Hu, and G. Min, "Learning-Based Resource Allocation in Cloud Data Center Using Advantage Actor-Critic," IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 5, pp. 1091-1104, 2021, doi: 10.1109/TPDS.2020.3038227.
[12] S. S. Gill et al., "Holistic Resource Management for Sustainable and Reliable Cloud Computing: An Innovative Solution to Global Challenge," Journal of Systems and Software, vol. 155, pp. 104-129, 2019, doi: 10.1016/j.jss.2019.05.025.
[13] M. Amiri, S. S. Gill, and S. Buyya, "A Reinforcement Learning Based Cost-Aware Auto-Scaler for Kubernetes Clusters," IEEE Transactions on Cloud Computing, pp. 1-1, 2021, doi: 10.1109/TCC.2021.3093567.
[14] A. Rossi, M. Nardelli, and V. Cardellini, "Horizontal and Vertical Scaling of Container-Based Applications Using Reinforcement Learning," in IEEE 12th International Conference on Cloud Computing (CLOUD), 2019, pp. 329-338, doi: 10.1109/CLOUD.2019.00062.
[15] A. Venkateswaran and U. Bellur, "Optimizing Resource Allocation Policies in Multi-Tenant, On-Demand Fog-Cloud Computing Environments," IEEE Transactions on Services Computing, pp. 1-1, 2021, doi: 10.1109/TSC.2021.3128161.
[16] B. Hindman et al., "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center," in Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI '11), 2011, pp. 295-308, ISBN: 978-931971-84-3.
[17] N. Herbst et al., "BUNGEE: An Elasticity Benchmark for Self-Adaptive IaaS Cloud Environments," in Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '15), 2015, pp. 46-56, doi: 10.1109/SEAMS.2015.23.
[18] M. Abdi, M. Hajibeygi, and H. Amiri, "A Comprehensive Review of Auto-Scaling Techniques in Cloud Environment: Taxonomy, Challenges, and Future Research Directions," Journal of Network and Computer Applications, vol. 186, p. 103115, 2021, doi: 10.1016/j.jnca.2021.103115.
[19] H. Arabnejad, C. Pahl, and J. G. Barbosa, "Reinforcement Learning for Service Composition in Mixed Fog-Cloud Environments," in IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), 2018, pp. 292-299, doi: 10.1109/FiCloud.2018.00051.
[20] A. U. Gias, G. Casale, and M. Woodside, "CloudCAMP: An Algorithm for Workload Prediction and Resource Allocation in Clouds," in IEEE 12th International Conference on Cloud Computing (CLOUD), 2019, pp. 176-183, doi: 10.1109/CLOUD.2019.00040.
[21] F. Rossi, M. Cardellini, and F. Lo Presti, "Horizontal and Vertical Scaling of Container-Based Applications Using Deep Reinforcement Learning," in IEEE International Conference on Cloud Engineering (IC2E), 2020, pp. 96-107, doi: 10.1109/IC2E48712.2020.00020.
[22] Z. Tang, X. Zhou, and K. Li, "A Reinforcement Learning Approach for Dynamic Scaling of Containerized Applications in Cloud Environment," IEEE Access, vol. 7, pp. 135743-135754, 2019, doi: 10.1109/ACCESS.2019.2941930.
[23] P. Bodik et al., "Performance Clarity: Telling Your Performance Story with Confidence," in Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference (Middleware '17), 2017, pp. 23-35, doi: 10.1145/3135974.3135984.
[24] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," IEEE Software, vol. 33, no. 3, pp. 42-52, 2016, doi: 10.1109/MS.2016.64.
[25] J. Xu, B. Palanisamy, and Q. Wang, "Cost-Efficient Container Scheduling in Cloud Data Centers Using Deep Reinforcement Learning," IEEE Transactions on Cloud Computing, pp. 1-1, 2021, doi: 10.1109/TCC.2021.3063699.
[26] S. Qanbari et al., "IoT Design Patterns: Computational Constructs to Design, Build and Engineer Edge Applications," in IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), 2016, pp. 277-282, doi: 10.1109/IoTDI.2015.18.
[27] A. Jindal, V. Podolskiy, and M. Gerndt, "Performance Modeling for Cloud Microservice Applications," in Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering (ICPE '19), 2019, pp. 25-32, doi: 10.1145/3297663.3310309.
[28] S. Taherizadeh and M. Grobelnik, "Key Influencing Factors of the Kubernetes Auto-Scaler for Computing-Intensive Microservice-Native Cloud-Based Applications," Advances in Engineering Software, vol. 140, p. 102734, 2020, doi: 10.1016/j.advengsoft.2019.102734.
[29] L. Zhu et al., "Capacity-Driven Deployment Optimization for Microservice-Based Applications in Clouds," IEEE Transactions on Cloud Computing, pp. 1-1, 2021, doi: 10.1109/TCC.2021.3098104.
[30] Y. Gan et al., "An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems," in Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19), 2019, pp. 3-18, doi: 10.1145/3297858.3304013.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306137|

[31] Y. Zhang et al., "CloudLess: A Cloud-Agnostic System for Deploying Containerized Applications," in Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '20), 2020, pp. 51-64, doi: 10.1145/3381052.3381316.
[32] M. Nardelli et al., "Multi-Level Elasticity for Wide-Area Data Streaming Systems: A Reinforcement Learning Approach," IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 9, pp. 2014-2027, 2020, doi: 10.1109/TPDS.2020.2978467.
[33] M. Abdi, M. Hajibeygi, and H. Amiri, "A Comprehensive Review of Auto-Scaling Techniques in Cloud Environment: Taxonomy, Challenges, and Future Research Directions," Journal of Network and Computer Applications, vol. 186, p. 103115, 2021, doi: 10.1016/j.jnca.2021.103115.
[34] R. Ibidunmoye, A. Rezaie, and E. Elmroth, "Adaptive Anomaly Detection in Performance Metric Streams," IEEE Transactions on Network and Service Management, vol. 15, no. 1, pp. 217-231, 2018, doi: 10.1109/TNSM.2017.2778096.
[35] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17), 2017, pp. 1285-1298, doi: 10.1145/3133956.3134015.
[36] W. Lin, H. Fan, Z. Qian, J. Xu, S. Yang, and J. Zhou, "HELAD: A Novel Network Anomaly Detection Model Based on Heterogeneous Ensemble Learning," Computer Networks, vol. 169, p. 107049, 2020, doi: 10.1016/j.comnet.2019.107049.
[37] X. Zhang et al., "Intelligent Anomaly Detection for Microservice Architectures with Temporal and Spatial Data Analysis," IEEE Transactions on Services Computing, pp. 1-1, 2021, doi: 10.1109/TSC.2021.3131092.
[38] A. Gulenko, M. Wallschläger, F. Schmidt, O. Kao, and F. Liu, "Evaluating Machine Learning Algorithms for Anomaly Detection in Clouds," in IEEE International Conference on Big Data (Big Data), 2016, pp. 2716-2721, doi: 10.1109/BigData.2016.7840917.
[39] L. Wu, J. Tordsson, A. Elmroth, and O. Kao, "MicroRAS: Automatic Recovery in the Absence of Historical Failure Data for Microservice Systems," in IEEE International Conference on Cloud Engineering (IC2E), 2020, pp. 38-47, doi: 10.1109/IC2E48712.2020.00013.
[40] A. Avritzer et al., "Scalability Assessment of Microservice Architecture Deployment Configurations: A Domain-Based Approach Leveraging Operational Profiles and Load Tests," Journal of Systems and Software, vol. 165, p. 110564, 2020, doi: 10.1016/j.jss.2020.110564.
[41] H. Qiu, S. S. Banerjee, S. Jha, Z. T. Kalbarczyk, and R. K. Iyer, "FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices," in 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI '20), 2020, pp. 805-825.
[42] M. Amiri, S. S. Gill, and S. Buyya, "A Reinforcement Learning Based Cost-Aware Auto-Scaler for Kubernetes Clusters," IEEE Transactions on Cloud Computing, pp. 1-1, 2021, doi: 10.1109/TCC.2021.3093567.
[43] S. Subramanya, T. Guo, P. Sharma, D. Irwin, and P. Shenoy, "SpotOn: A Batch Computing Service for the Spot Market," in Proceedings of the Sixth ACM Symposium on Cloud Computing (SoCC '15), 2015, pp. 329-341, doi: 10.1145/2806777.2806849.
[44] A. Venkateswaran and U. Bellur, "Optimizing Resource Allocation Policies in Multi-Tenant, On-Demand Fog-Cloud Computing Environments," IEEE Transactions on Services Computing, pp. 1-1, 2021, doi: 10.1109/TSC.2021.3128161.
[45] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A Multi-Objective Ant Colony System Algorithm for Virtual Machine Placement in Cloud Computing," Journal of Computer and System Sciences, vol. 79, no. 8, pp. 1230-1242, 2013, doi: 10.1016/j.jcss.2013.02.004.
[46] S. Taherizadeh, A. C. Jones, I. Taylor, Z. Zhao, and V. Stankovski, "Monitoring Self-Adaptive Applications within Edge Computing Frameworks: A State-of-the-Art Review," Journal of Systems and Software, vol. 136, pp. 19-38, 2018, doi: 10.1016/j.jss.2017.10.033.
[47] M. Nardelli et al., "Multi-Level Elasticity for Wide-Area Data Streaming Systems: A Reinforcement Learning Approach," IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 9, pp. 2014-2027, 2020, doi: 10.1109/TPDS.2020.2978467.
[48] A. Avritzer et al., "Scalability Assessment of Microservice Architecture Deployment Configurations: A Domain-Based Approach Leveraging Operational Profiles and Load Tests," Journal of Systems and Software, vol. 165, p. 110564, 2020, doi: 10.1016/j.jss.2020.110564.
[49] S. S. Gill et al., "Holistic Resource Management for Sustainable and Reliable Cloud Computing: An Innovative Solution to Global Challenge," Journal of Systems and Software, vol. 155, pp. 104-129, 2019, doi: 10.1016/j.jss.2019.05.025.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306137|

[50] H. Qiu, S. S. Banerjee, S. Jha, Z. T. Kalbarczyk, and R. K. Iyer, "FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices," in 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI '20), 2020, pp. 805-825.
[51] A. Gulenko, M. Wallschläger, F. Schmidt, O. Kao, and F. Liu, "A System Architecture for Real-Time Anomaly Detection in Large-Scale NFV Systems," Procedia Computer Science, vol. 94, pp. 491-496, 2016, doi: 10.1016/j.procs.2016.08.076.
[52] D. Baylor et al., "TFX: A TensorFlow-Based Production-Scale Machine Learning Platform," in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17), 2017, pp. 1387-1395, doi: 10.1145/3097983.3098021.
[53] E. Breck et al., "The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction," in IEEE International Conference on Big Data (Big Data), 2017, pp. 1123-1132, doi: 10.1109/BigData.2017.8258038.
[54] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," IEEE Software, vol. 33, no. 3, pp. 42-52, 2016, doi: 10.1109/MS.2016.64.
[55] S. Taherizadeh and M. Grobelnik, "Key Influencing Factors of the Kubernetes Auto-Scaler for Computing-Intensive Microservice-Native Cloud-Based Applications," Advances in Engineering Software, vol. 140, p. 102734, 2020, doi: 10.1016/j.advengsoft.2019.102734.
[56] X. Zhu, Y. Li, Y. Gao, and Z. Zheng, "A Survey on Big Data Preprocessing," in IEEE International Conference on Big Data (Big Data), 2017, pp. 2696-2705, doi: 10.1109/BigData.2017.8258236.
[57] D. Gunning and D. Aha, "DARPA's Explainable Artificial Intelligence (XAI) Program," AI Magazine, vol. 40, no. 2, pp. 44-58, 2019, doi: 10.1609/aimag.v40i2.2850.
[58] M. Rhode et al., "Identifying and Analyzing Security Challenges for Machine Learning in the Context of the EU MDR and IVDR," in IEEE Security and Privacy Workshops (SPW), 2021, pp. 86-93, doi: 10.1109/SPW53761.2021.00018.
[59] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning: Concept and Applications," ACM Transactions on Intelligent Systems and Technology, vol. 10, no. 2, pp. 1-19, 2019, doi: 10.1145/3298981.
[60] A. Goel, D. Simlot, J. Walia, and A. Nayyar, "Serverless Computing for AI Applications: A Systematic Literature Review," IEEE Access, vol. 9, pp. 131540-131567, 2021, doi: 10.1109/ACCESS.2021.3115019.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details