Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2022, 9(10):91-95

Research Article ISSN: 2394 - 658X

Troubleshooting Resource Exhaustion during Pod Scaling in Kubernetes

Mohit Thodupunuri

MS in Computer Science
Sr Software Developer - Charter Communications Inc
mohit.thodupunuri@gmail.com

ABSTRACT
In Kubernetes, scaling applications horizontally by adding new pods is a fundamental practice for handling varying workloads. However, when resource exhaustion occurs, Kubernetes may fail to scale pods effectively, resulting in unscheduled or evicted pods and system instability. Resource exhaustion typically arises when nodes in the cluster do not have enough CPU, memory, or storage to accommodate additional pods. This article provides a comprehensive guide for troubleshooting this common issue. It covers key diagnostic steps, including checking node resource utilization, inspecting pod resource requests and limits, investigating pending pod status, and ensuring proper configuration of auto scalers and node affinity rules. Following the outlined troubleshooting approach, Kubernetes administrators can quickly identify the root causes of resource exhaustion and apply targeted solutions, ensuring more reliable and efficient pod scaling in production environments. The article also offers best practices for avoiding resource-related scaling issues in the future.

Keywords: resource exhaustion, troubleshooting methods, challenges, troubleshooting scenarios, pod scaling, Kubernetes, Troubleshooting significance

INTRODUCTION

Kubernetes is an open-source platform that provides container management and helps in software development with better scaling and easy deployment methods. Applications' workloads are distributed and maintained easily, reducing the development team's exclusive efforts. Therefore, load fluctuations become easy to tackle. However, there remains a possibility of failure when a serious resource fluctuation carries the potential of failure of pod scaling. The pods' disruption can be due to staying in pending states or complete downtime [1].

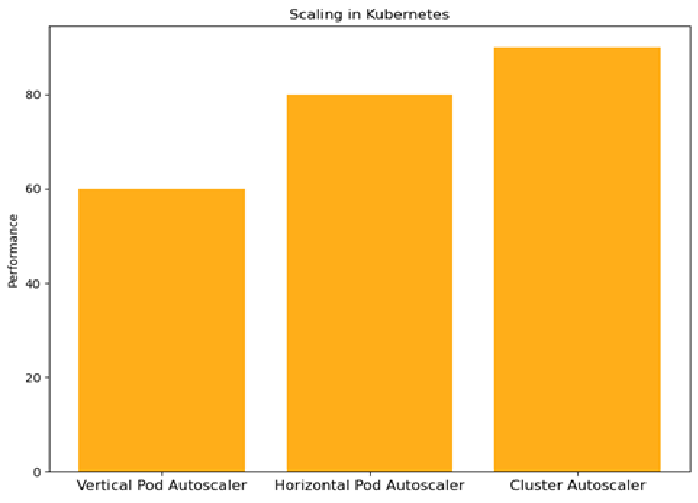

Figure 1: Effect of Scalers in Kubernetes Scaling

The figure 1 above shows the tentative or approximate percentages of the effect of each autoscalers on the performance of applications using Kubernetes. The cluster autoscaler is the most efficient and provides better performance compared to VPA and HPA as it utilizes more advance auto scaling methods with each task distributed in proper constraints.
The resource exhaustion may result in poor pod scheduling in Kubernetes which affects the overall user experience due to poor performances in critical scenarios. The key indicators of resource exhaustion can be accumulated in the following points,

- Pods in a delayed state where tasks remain pending for an unexpected period.
- The nodes may end up in the memory or disk shortage during high-demand task execution.
- The poor scaling or no scaling can result in high memory consumption with high processing demands which puts more pressure on scheduling.

The presented research studies explore the possible scenarios where troubleshooting can be targeted. The possible methods to resolve these conflicts are suggested which are easy to follow and do implementations. However, the focus remains on pod scaling and its significance is discussed in detail beforehand. The problem statement is targeted considering the troubleshooting during the exhaustion of resources specifically during pod scaling in Kubernetes. Future developments are suggested at the end to present the more advanced troubleshooting methods that can be designed shortly.

LITERATURE REVIEW

The declarative model is used in the resource allocation in Kubernetes to handle requests for resources and apply limitations for pods. This scheduling can be done with multiple methods. The research study finds the optimal timing of selecting nodes for pod and takes into account the problem of selecting the current optimal node and not the costs of resource usage. The resource scheduling methods are discussed in detail where the optimized way is selected to deploy the pods [2].
Resource exhaustion is the main challenge in Kubernetes as it directly affects efficiency. The poor configuration of resource requests can lead to an inefficient utilization of resources and overall reduced performance of the applications. The load balancing can be done for example with the use of horizontal pod autoscaling which ensures scalability with the increased number of replicas [3]. However, it’s important to distribute client requests equally otherwise it can result in long delays.
The performance metrics are important to analyze with monitoring tools like Grafana and Prometheus for complete resource management. These tools provide real-time visualization of useful insights during the allocation of resources and highlight any conflicts if appear. The different components to monitory are CPI, memory, disk and network where a baseline can be set as a reference point to measure the performance of manually deployed Kubernetes clusters [4]. The federated cloud infrastructure can be analyzed in comparison with the tactical networks and disadvantages associated with them [5]

PROBLEM STATEMENT

Better resource management is a critical factor in overall software development, and it becomes more prominent when it comes to Kubernetes as its main job is to handle resources efficiently. The clusters must have sufficient memory, and storage all the time even during high resource demands. The bad resource management can result in misconfigured infrastructure with a lot of limitations. Primarily the shortage of resources can affect pod scheduling which in turn reduces the performance of associated applications. So, there is a need to properly troubleshoot resource exhaustion during pod scaling in Kubernetes.

IMPORTANCE OF POD SCALING IN KUBERNETES

One of the well-known features in Kubernetes is horizontal pod scaling which helps to suggest possible solutions for high-load scenarios. There are automatic methods to handle the high workload situations that are offered by Kubernetes,

- **Availability:** The scaling trait from pod scaling offers responsive results and reduced downtime, especially during high traffic.
- **Optimization:** The resources are dedicated in an optimized manner where balance is maintained between the uses of resources against costs.
- **Event Handling:** During high traffic like seasonal sales or other promotional periods, pod scaling offers a broad perspective and provides a smooth experience.
- **Diverse Architectural Support:** The individual micro services that are integrated are associated with appropriate resources.
- **Continuous Integration:** The continuous updates, integrations, and deployments are handled with automatic scaling facilities. Automatic bug fixing plays an important role in providing a better user experience by retaining performance.

CHALLENGES

Pod scaling is extremely important in Kubernetes, but it is associated with serious challenges when it comes to practical deployment. Let’s highlight some of the prominent challenges [6],

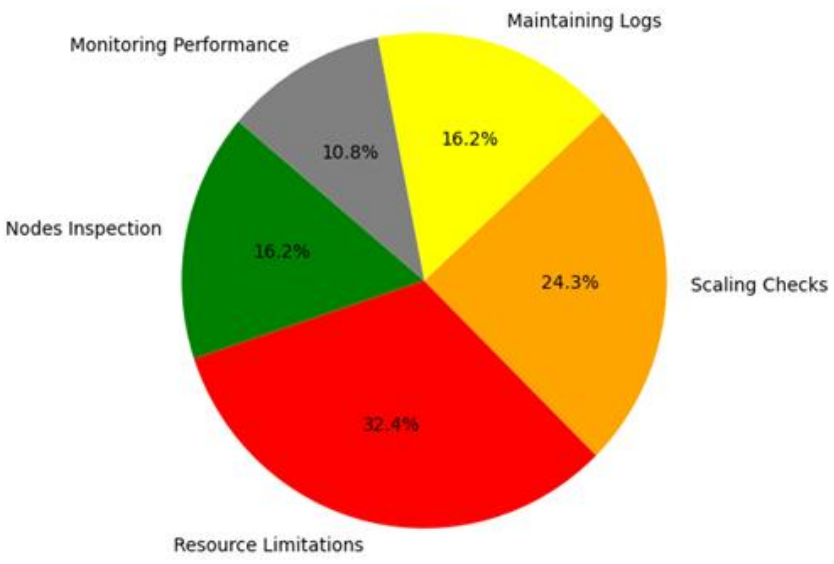

Figure 2: Intensity of Challenges for Pod Scaling

The above pie chart shows that resource limitations are the most difficult challenge to handle. This is because historical data is required which must be verifiable and taken on legitimate grounds. The other challenges are discussed in detail below.

Memory Shortage

The limited disk space is one of the critical factors and a potential challenge to pod scaling which impacts the efficiency of the application which is associated with Kubernetes. Therefore, the complex and data-dependent applications are mostly affected during high demand because of memory shortage.

Improper Scaling

Horizontal Pod Autoscaler (HPA) is responsible for automatic cluster scaling and its job is to be triggered immediately. However, the improper configuration of this auto-scaling is possible due to inaccurate configuration values and constraints and therefore affects pod scaling.

Troubleshooting and Cluster Latency

Limited visibility in the performance measures of clusters can result in delayed troubleshooting and where scaling problems can occur. The additional nodes may consume memory and take extra time during the initial introduction. The performance can therefore be compromised for a limited time during high traffic which also directly affects the user experience [7].

Bad Constraints

The upcoming pods may be restricted in Kubernetes due to limited resources in nodes. This scenario mostly occurs during the high demand for resources in critical executions. The memory and execution rules need to be followed strictly for better execution.

Lack of Management

The bad management of resources and inappropriate allocation can result in underutilization of resources which increases the resource exhaustion factor. This results in less cluster efficiency during pod scaling and therefore better management is required during resource handling.

RESOURCE EXHAUSTION IDENTIFICATION

Following is the stepwise procedure for the identification of resource exhaustion,

Inspect Nodes

Check if the memory and storage are dedicated appropriately through the clusters. The continuous observation can be done with a single command of kubectl top nodes. The kubectl describe nodes can be used to further get information regarding disk pressure and memory pressure.

Exploring Limitations

The kubectl describe pod <pod-name> command can be used to examine pods and check if the requests and limits meet the requirement of actual node capacity to reduce negative experience impacts on users.

Scheduling Checks

The scheduling errors can be monitored by investigating pending pods with kubectl get events command. Also, there is a need to identify negative rules, taints, and tolerations that can affect scheduling negatively.

Maintain Logs

The behavior of Horizontal Pod Autoscaler (HPA) can be verified using kubectl describe hpa <deployment-name> command. The up-and-down scaling failure can be detected in this way with continuous monitoring of errors.

Performance Metrics

The last but the most significant step in the identification of resource exhaustion where visualized trends can often be studied to get insights from the continuous performance metrics. The real-time insights can be displayed in the Kuberneted dashboard.

TROUBLESHOOTING METHODS

Before considering the actual methods of troubleshooting resource exhaustion during pod scaling, here are some of the common scenarios where troubleshooting may be required,

- The nodes may run out of memory where more nodes are being utilized, and pods get into the pending state. This can be resolved by enabling the auto scalers and optimization of pod resource requests.
- The node shortage can be faced sometimes when disk pressure is increased. This can be reduced by clearing rarely used or unused volumes or extending storage where possible.
- The scheduling failures may arise due to over or under-requesting which leads to inaccurate utilization. The solution to this is to use historical data to limit the requests with consideration of resources. The tools like Metrics Server are already offered by Kubernetes.
- There are some pending pods left even when the resources are available which mostly happens due to incorrect configurations. This however can be solved easily by balancing workload distribution with suitable constraints.

Continuous Monitoring

The monitoring tools like Grafana can be used to carry out extensive trend studies. The active alert methods can be used to notify in advance for future threshold breaks. These trends are especially helpful when identifying peak scenarios in advance. If identified early, the extra nodes can be introduced in advance to minimize conflicts.

Resource Limitation

The resources should be limited according to the potential needs in the respective timings, and some should be spared for the high-demand periods. The quota system can be introduced, and limits can be set using kubectl create quota <quota-name>.

Efficient Autoscaling

The most suitable thresholds can be provided, and pods can be scaled with better HPA configuration values. The cluster's auto scalers should be installed correctly to add nodes respectively during high-demand periods. This in return helps to reduce the manual costs of conflict resolution as well.

Limit Constraints

The accurate values can be identified to set limits on the number of requests. The performance metrics can be helpful here to identify the correct values. The underutilization of resources can be avoided in this way. The resources can be then freely available for critical pods.

FUTURE DEVELOPMENT

Considering the current advancements in modern scaling techniques even within Kubernetes declares the potential future with better-equipped solutions. The prominent forecasted solution is the introduction of better resource-scheduling algorithms. These active feedback-oriented algorithms may analyze the runtime performance metrics and make more insightful decisions in critical scenarios. The other future developments may include proactive scaling where historical data can be used to forecast resource utilization, reduced wastage of resources with dynamic container allocation, distributed workload methods where regional clusters are utilized in a better way, and improved scaling capabilities for smooth experiences during a critical time of application execution for the end users.

CONCLUSION

Troubleshooting resource exhaustion during pod scaling is much more significant specifically during high-demand tasks and therefore requires a step-by-step approach to doing necessary configurations. The resource shortage can be minimized with better utilization of Kubernetes' methods and tools. The appropriate strategies are optimized configuration constraints and auto-scaling to maintain the cluster's ability to provide resources at all times even during high demand. The challenges are mentioned in the presented writing and these can be dealt with through efficient troubleshooting methods as discussed.

With the continuous evolution of Kubernetes techniques, predictive scaling, and strong architecture can be introduced to provide better application scaling. Thus, efficient resource utilization helps to reduce costs and make the applications more resilient in dealing with high-demand situations. These ongoing innovations can equip the organizations with optimized Kubernetes clusters and the high-demand scenarios will then be dealt with without any problem.

REFERENCES

[1]. B. Burns, J. Beda, K. Hightower and L. Evenson, Kubernetes Up and Running: Dive into the Future of Infrastructure, O'Reilly, 02 Aug 2022.

[2]. Z. Wei-guo, M. Xi-lin and Z. Jin-zhong, "Research on Kubernetes' Resource Scheduling Scheme," in ICCNS '18: Proceedings of the 8th International Conference on Communication and Network Security, 02 Nov 2018.

[3]. Q.-M. Nguyen, L.-A. Phan and T. Kim, "Load-Balancing of Kubernetes-Based Edge Computing Infrastructure Using Resource Adaptive Proxy," vol. 22, no. 8, 22 Feb 2022.

[4]. A. P. Ferreira and R. Sinnott, "A Performance Evaluation of Containers Running on Managed Kubernetes Services," in 2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Sydney, NSW, Australia, 28 Jan 2020.

[5]. M. Fogli, T. Kudla, B. Musters, G. Pingen, C. V. d. Broek and H. Bastiaansen, "Performance Evaluation of Kubernetes Distributions (K8s, K3s, KubeEdge) in an Adaptive and Federated Cloud Infrastructure for Disadvantaged Tactical Networks," in 2021 International Conference on Military Communication and Information Systems (ICMCIS), The Hague, Netherlands, 20 July 2021.

[6]. K. Lehtinen, "Scaling a Kubernetes Cluster," 20 Apr 2022.