Event Driven Autoscaling Strategies For Kubernetes Applications

M.B. Yelpale[1], Purva Mahajan[2], Priyanka Kulkarni[3], Amruta Nikam[4], Vaishnavi Taware[5]

[1]Assistant Professor , Department of Computer Engineering , NBNSTIC , Pune , India
[2,3,4,5] Student , Department of Computer Engineering , NBNSTIC , Pune , India

---***---

Abstract - This study emphasizes the critical role that Kubernetes autoscaling plays in improving the performance of containerized environments. It overcomes the drawbacks of conventional scaling techniques, particularly when it comes to adjusting to the real-time message queue traffic patterns of contemporary cloud-native apps. The research suggests a customized metric scaler that improves on Kubernetes' Horizontal Pod Autoscaler in order to address these issues. Through the use of customized metrics from message queue traffic, this innovation optimizes application performance and enables dynamic workload scaling. In order to facilitate smooth integration and agile workload management, the research offers a microservices architecture that incorporates essential elements such as Producer, RabbitMQ, Consumer, Decision, and Kubernetes Services. Additionally, it displays the system's applications in e-commerce platforms, real-time analytics, IoT data processing, and microservices orchestration. With its integration expansion of supported metrics, and outline of upcoming scalability and adaptability advancements, this research represents a major step towards efficient Kubernetes clusters..

Key Words **: Kubernetes autoscaling, Cloud-native applications, Real-time event-driven scaling, Custom metric scale, Message queue traffic patterns, Microservices architecture, Dynamic workload optimization, Resource-efficient Kubernetes.**

1.INTRODUCTION

When it comes to optimizing application performance in containerized settings, Kubernetes autoscaling is revolutionary. It ensures maximum efficiency during intervals of high traffic and resource conservation during low traffic times by dynamically adjusting the number of running instances based on variations in workload. Through the use of tools like Horizontal Pod Autoscaler (HPA), Kubernetes autonomously scales applications in real-time by keeping track on metrics like CPU and memory. In addition to increasing responsiveness and dependability, this optimizes resource utilization and reduces costs. In the dynamic world of cloud-native computing, adopting Kubernetes autoscaling is essential to building robust and effective systems.

2. Body of Paper

2.1 Problem Statement:

Considering real-time events frequently result in dynamic and unpredictable workloads for modern cloud-native apps, standard autoscaling solutions are unsuitable for effective resource management. Creating a unique auto-scaling solution for Kubernetes workloads that can adjust dynamically in response to patterns of real-time message queue traffic is a problem.

2.2 Proposed Solution :

In order to address the shortcomings of the conventional Kubernetes Horizontal Pod Autoscaler (HPA), a custom metric scaler is introduced in the suggested solution. This custom scaler allows developers to define and use custom metrics that are pulled from message queue traffic, in contrast to HPA's dependence on CPU and memory metrics.

Based on particular queue patterns, it dynamically modifies workload replicas to optimize application performance. The system's smooth integration with Kubernetes ensures that it can react instantly to changes in workload. Its flexibility improves responsiveness and resource usage by concentrating on the specific requirements of each application.

Its efficacy and dependability across a range of cloud-native environments will be ensured by rigorous testing and community feedback driving ongoing improvements.

2.3 System Design :

System Architecture:

Microservice Architecture: Implementing a microservices architecture to improve scalability and maintainability. Components include:

- Producer Service: Produce events according to application metrics.
- RabbitMQ Service: Holds events until used by the Consumer Service.
- Consumer Service: Handles events and intiates autoscaling decisions.
- Decision Service: Informs Kubernetes about scaling decisions after analyzing events.
- Kubernetes Service: Based on choices made by the Decision Service, scales consumer service pods.

Component Details:

1. **Producer Service:**
   - Gathers metrics or events unique to applications and forword them to the message queue.

2. **RabbitMQ Service:**
   - Stores events until they are used, serving as the message broker.
   - Uses a queue data structure for messages management.

3. **Consumer Service:**
   - Collects and handles messages from RabbitMQ in accordance with predetermined capacity or guidelines..

4. **Decision Service:**
   - Examin the data gathered from the message queue.
   - Produce scaling decisions based on patterns and preset policies.
   - Makes autoscaling decisions by evaluating workload demands using custom logic.

5. **Kubernetes Service:**
   - Adapts the number of Consumer Service pods based on choices made by the Decision Service by integrating with Kubernetes APIs.
   - Enables autoscaling based on user-defined metrics by utilizing a custom metrics API.

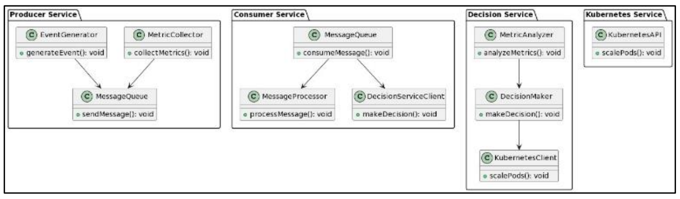

Fig 1. System Architecture

Communication:

1. Internal Communication:
   - Microservices can interact to others effortlessly through messaging protocols or RESTful APIs.
   - Within the cluster, utilize secure communication protocols.

2. External Integration:
   - For accurate metrics and analysis, integrate likely with external monitoring systems, logging services, or other data sources.

Data Flow and Scalability:

1. Data Flow:
   - For processing and decision-making, metrics move from the Producer Service to the RabbitMQ Service then to the Consumer Service.
   - The Kubernetes Service scales in response to decisions made by the Decision Service.

2. Scalability:
   - Put scalability into practice at different levels, such as handling message queues, the consumer service's processing capabilities, and the scaling of the Kubernetes infrastructure.

2.4 Applications :

1. **Microservices Orchestration for Cloud-Native Applications**: Microservices orchestration is essential for cloud-native application scaling. For optimum performance, every microservice dynamically scales to meet a variety of demands and traffic patterns.

2. **Real-Time Analytics Platforms:** To manage dynamic data streams, real-time analytics platforms require event-driven autoscaling. In order to guarantee prompt processing of rising requests for insights, it strategically distributes resources during peak periods.

3. **IoT Data Processing:** Autoscaling that is triggered by events is advantageous for IoT applications that handle different data volumes. Adaptive scaling effectively handles growing data flows by allocating more resources as needed.

4. **E-Commerce Platforms**: Event-driven autoscaling is vital in the dynamic e-commerce landscape. It dynamically adjusts resources to manage rapid changes in website traffic during events like sales or promotions, preventing crashes and facilitating smooth transactions.

3. Literature survey

1) Paper name: Auto-scaling of Containers: the Impact of Relative and Absolute Metrics
Author name: Emiliano Casalicchio, Vanessa Perciballi
Description: Recommends using both relative and absolute measurements for auto-scaling decisions. Focuses on absolute metrics for CPU-intensive workloads. Develops and evaluates an enhanced auto-scaling method to improve response time in Kubernetes' horizontal auto-scaling algorithm.

2)Paper name: Database Scaling on Kubernetes
Author name: H.C.S. Perera, T.S.D. De Silva, W.M.D.C. Wasala, R.M.P.R.L. Rajapakshe, N. Kodagoda, Udara Srimath S. Samaratunge Arachchilage, H.H.N.C. Jayanandana
Description: Explores Kubernetes' impact on managing databases. Proposes a solution for highly available PostgreSQL databases on Kubernetes, addressing data synchronization issues and introducing a novel auto-scaling mechanism..

3)Paper name: Intelligent Autoscaling of Microservices in the Cloud for Real-Time Applications.
Author name: ABEER ABDEL KHALEQ, ILKYEUN RA
Description: Addresses microservices auto-scaling challenges for cloud applications. Introduces an autonomous system using a generic algorithm and reinforcement learning on Google Kubernetes Engine. Achieves a 20% increase in response time while maintaining Quality of Service (QoS) constraints.

4)Paper name: Research on Resource Prediction Model Based on Kubernetes Container Auto-scaling Technology
Author name: Anqi Zhao , Qiang Huang , Yiting Huang , Lin Zou, Zhengxi Chen and Jianghang Song
Description: Explores cloud computing's influence on resource acquisition and software deployment, focusing on Docker and Kubernetes. Presents an auto-scaling optimization technique based on predictive modeling to minimize reaction times during capacity increase.

5)Paper name: Autoscaling Pods on an On-Premise Kubernetes Infrastructure QoS-Aware.
Author name: LLUÍS MAS RUÍZ , PERE PIÑOL PUEYO, JORDI MATEO-FORNÉS , JORDI VILAPLANA MAYORAL , and FRANCESC SOLSONA TEHÀS
Description: examines the benefits of cloud computing and microservices. emphasizes using Docker containers and Kubernetes to optimize SLOs (Service Level Objectives) and dynamically scale resources to enhance QoS (Quality of Service).

CONCLUSION AND FUTURE SCOPE

Kubernetes' real-time, event-driven autoscaling system is a major advancement in addressing the demands for modern applications. Several approaches are used in this project to improve system responsiveness, optimize resource allocation, and facilitate cost-effective management in Kubernetes clusters. Its scalability and adaptability could revolutionize the management of cloud-native applications. This progress is driven by extending supported metrics beyond message queues and improving cost optimization techniques. By addressing the growing need for responsive and scalable resource management in today's computing environments, it greatly expands Kubernetes' capabilities.

By adding more metrics (network traffic, unique application-specific metrics, external data sources) for dynamic autoscaling decisions, the project hopes to continue innovating in the future. Machine learning integration can improve prediction performance, and dynamic policy management evolves to accommodate changing application behaviors. Advanced event correlation, cost optimization, and expanding autoscaling across multiple Kubernetes clusters for hybrid cloud environments and complex distributed applications are priorities. Monitoring and management will also be improved by placing a strong emphasis on user-friendly interfaces and visualization capabilities. The goal of these developments is to strengthen Kubernetes' position as a resource management tool that is scalable and responsive.

ACKNOWLEDGEMENT

We are grateful to NBN Sinhgad Technical Institute Campus, Pune, for constant support and encouragement in working on our paper, and we are also thankful to Prof. S.B. Bendale (HoD Comp. dept) and all guides for their helpful guidance and support in improving our paper.

REFERENCES

[1] MetalLB. MetalLB, Bare Metal Load-Balancer for Kubernetes.
Accessed: Mar. 14, 2022.
[2]F. Rossi, ‘‘Auto-scaling policies to adapt the application deployment in Kubernetes,’’ in Proc. ZEUS, 2020, pp. 30–38.

[3]T.-T. Nguyen, Y.-J. Yeom, T. Kim, D.-H. Park, and S. Kim, ‘‘Horizontal pod autoscaling in kubernetes for elastic container orchestration,’’ Sensors, vol. 20, no. 16, p. 4621, Aug. 2020.

[4] Kubernetes. Horizontal Pod Autoscaler. Accessed: Mar. 14, 2022.
[Online]. Available: https://kubernetes.io/docs/tasks/run-application/
horizontal-pod-autoscale/

[5] S. Taherizadeh and M. Grobelnik, ‘‘Key influencing factors of the
Kubernetes auto-scaler for computing-intensive microservice-native
cloud-based applications,’’ Adv. Eng. Softw., vol. 140, Apr. 2020,
Art. no. 102734. [Online]. Available: https://www.sciencedirect.com/
science/article/pii/S0965997819304375