COMPARATIVE ANALYSIS OF CPU BASED AND REQUEST PER-SECOND (RPS) METRICS IN KUBERNETES HORIZONTAL POD AUTOSCALER (HPA)

By

I Gusti Putu Rangga Adithia Putra, 2215101001

Department of Informatics Engineering

ABSTRAK

Kubernetes Horizontal Pod Autoscaler (HPA) secara bawaan umumnya menggunakan utilisasi CPU sebagai dasar penskalaan. Namun, pada beban kerja *I/O-bound* seperti Nginx *reverse proxy*, utilisasi CPU tidak selalu merepresentasikan tekanan koneksi dan penurunan kinerja layanan secara akurat. Penelitian ini bertujuan membandingkan efektivitas HPA berbasis CPU dan *Request per Second* (RPS) dalam menjaga ketersediaan dan responsivitas layanan, serta mengevaluasi dampak optimasi konfigurasi HPA berbasis RPS. Eksperimen dilakukan pada klaster Kubernetes *single-node* menggunakan Minikube di Google Cloud Platform dengan arsitektur dua tingkat yang terdiri atas Nginx sebagai *reverse proxy* dan layanan *backend* dengan penundaan pemrosesan tetap 500 ms. Pengujian menggunakan k6 mencakup empat pola trafik, yaitu *ramp-up*, *sustained high-load*, *cyclic load*, dan *gradual decreasing load*. Setiap kombinasi konfigurasi dan skenario diuji sebanyak lima kali. Metrik yang dianalisis meliputi jumlah replika pod, utilisasi CPU, latensi p50 dan p99, serta tingkat kesalahan HTTP 5xx. Hasil penelitian menunjukkan bahwa HPA berbasis RPS secara umum lebih responsif dan mampu mempertahankan ketersediaan layanan lebih baik dibandingkan HPA berbasis CPU. Uji-t berpasangan menunjukkan perbedaan signifikan pada tingkat kesalahan dan latensi p99 di seluruh skenario beban puncak dengan nilai $p < 0{,}001$. Optimasi melalui penggunaan imagePullPolicy: IfNotPresent, pemendekan interval *scrape* Prometheus dari 15 menjadi 5 detik, dan kebijakan *scale-up* yang lebih agresif menurunkan rata-rata tingkat kesalahan pada *sustained high-load* dari 24,97% menjadi 0,001% dan pada *cyclic load* dari 67,99% menjadi 0,05%. Meskipun demikian, latensi p99 masih relatif tinggi pada beberapa interval. Penelitian selanjutnya disarankan melakukan validasi pada klaster *multi-node*, menggunakan trafik dunia nyata berdurasi panjang, menerapkan pendekatan *autoscaling* hibrida, serta menambahkan pengukuran langsung terhadap koneksi Nginx, *file descriptor*, dan TCP *listen queue*.

Kata-kata kunci: Kubernetes, Horizontal Pod Autoscaler, Request per Second, autoscaling, Nginx

COMPARATIVE ANALYSIS OF CPU BASED AND REQUEST PER-SECOND (RPS) METRICS IN KUBERNETES HORIZONTAL POD AUTOSCALER (HPA)

By

I Gusti Putu Rangga Adithia Putra, 2215101001

Department of Informatics Engineering

ABSTRAK

Kubernetes Horizontal Pod Autoscaler (HPA) commonly uses CPU utilization as its default scaling signal. However, for I/O-bound workloads such as an Nginx reverse proxy, CPU utilization may not accurately represent connection pressure or service degradation. This study aims to compare the effectiveness of CPU-based and Request per Second (RPS)-based HPA configurations in maintaining service availability and responsiveness and to evaluate the impact of optimizing the RPS-based configuration. The experiments were conducted on a single-node Kubernetes cluster using Minikube on Google Cloud Platform. A two-tier architecture was deployed, consisting of Nginx as the reverse proxy and a backend service with a fixed 500 ms processing delay. Load testing was performed using k6 under four traffic patterns: ramp-up, sustained high load, cyclic load, and gradual decreasing load. Each combination of HPA configuration and traffic scenario was independently executed five times. The evaluated metrics included pod replica count, CPU utilization, p50 and p99 latency, and HTTP 5xx error rate. The results show that the RPS-based HPA was generally more responsive and maintained service availability more effectively than the CPU-based HPA. Paired t-tests identified statistically significant differences in error rate and p99 latency across all peak-load scenarios, with p-values below 0.001. Further optimization using imagePullPolicy: IfNotPresent, a reduction of the Prometheus scrape interval from 15 to 5 seconds, and a more aggressive scale-up policy reduced the average error rate during sustained high load from 24.97% to 0.001% and during cyclic load from 67.99% to 0.05%. Nevertheless, p99 latency remained relatively high during several intervals. Future studies should validate these findings in production-grade multi-node clusters, use longer and more realistic traffic patterns, investigate hybrid autoscaling approaches, and include direct measurements of Nginx connections, file descriptor usage, and TCP listen queues.

Keywords: Kubernetes, Horizontal Pod Autoscaler, Request per Second, autoscaling, Nginx