## Background & Motivation

Modern Kubernetes deployments often start with generously provisioned CPU and memory resources to ensure service reliability. However, this overprovisioning results in significant resource wastage, increasing operational costs without delivering proportional performance benefits. This inefficiency becomes particularly pronounced in microservice architectures, where each service may have distinct performance-resource relationships. Existing autoscaling strategies like HPA/VPA and static provisioning heuristics fail to capture the intricate latency behaviors under multi-axis resource reduction (CPU, memory, or both).

## Objective:

To design an intelligent, latency-constrained resource tuning strategy that dynamically reduces CPU and memory *requests* and *limits* for each service, starting from overprovisioned conditions. The reduction continues until the service latency increases by no more than a certain percentage from the baseline (measured under full resource availability).

## Experimental Scope

Your research involves four distinct Java and Go-based microservices under constant or semi-constant request load:

* Prime Verifier
* Echo
* Hash Generator
* Random Password Generator

Each service is evaluated under:

1. CPU-only reductions
2. Memory-only reductions
3. Combined CPU and memory reductions

Latency, CPU usage, and memory usage are tracked continuously during gradual reductions.