# Objective

To intelligently and autonomously optimize CPU and memory resource limits for multiple microservices deployed in a Kubernetes environment, such that:

Application latency remains within acceptable bounds, defined as no more than certain percentage of degradation from the service-specific baseline latency under normal load.

Resource configurations adapt dynamically and continuously, without relying on statically defined Critical Reduction Points (CRPs) or manual tuning thresholds.

# Expanded Sub-Objectives

1. Resource Efficiency

* Reduce overprovisioning of CPU and memory requests/limits.
* Automatically discover and apply the minimum safe resources required to meet Service-Level Objectives (SLOs), thereby minimizing infrastructure costs.

1. SLA Adherence

* Continuously ensure that latency does not exceed a certain amount of the baseline latency observed at initial overprovisioned levels.
* Latency-aware adjustments must be consider individual behavior and load profiles.

1. Dynamic & Continuous Adaptation

* Adjust CPU and memory resource limits at runtime based on real-time telemetry (e.g., latency, request rates, usage).
* Avoid one-time profiling or offline stress testing; instead, leverage on-the-fly observations for decision-making.

1. Online Learning Without CRPs

* Replace CRP-based static analysis (which identifies “safe-to-reduce” breakpoints) with adaptive feedback control that responds continuously to observed performance impacts.
* Learn the performance/resource tradeoff implicitly over time through interaction with the system and feedback loops.

1. Service and Multi-Stage Optimization

* Support microservices with potentially diverse performance/resource characteristics.
* Allow the system to generalize and specialize across services without manual tuning heuristics.