CMPT 756 Project Progress Report

Git: link | **Members:** Kexin Liu, Hanyi Zhang, Guang Yang, Yingqi Zhao, Shiyu Zhao

Solution Design

We deploy the same YOLO-based inference service in two ways to compare containerized versus serverless approaches. On the **Kubernetes side**, we run a Dockerized YOLO model on **GKE (Google Kubernetes Engine)**, with an **Ingress** or **Load Balancer** receiving client requests and routing them through a Kubernetes Service to multiple YOLO Pods. These pods scale via a Horizontal Pod Autoscaler (HPA) based on CPU/GPU usage, and we collect performance metrics using **Prometheus** and **Grafana**. On the **serverless side**, we host the same Docker image on **Google Cloud Run**, which automatically scales container instances based on request volume. Cloud Run provides an HTTPS endpoint for inference requests, and **GCP Monitoring/Logging** captures metrics on latency, resource utilization, and scaling events. Both deployments expose a REST API for uniform testing, returning detection results (bounding boxes and classes) to the client. Ultimately, we compare them on key factors such as latency, throughput, scalability, cost, and operational overhead to guide best practices for real-time ML workloads. Architecture image for solution design is uploaded to this link.

Implementation

Current Progress

- Successfully fine-tuned a pre-trained YOLO model with COCO128 datasets (link) in the local environment (link), using 80% of the dataset for training and the remaining 20% for inference
- Created and tested a Dockerfile which encapsulates all dependencies of the YOLO model and pushed it to Google Container Registry
- Use `gcloud run deploy` to deploy the docker image on Cloud Run for a serverless environment

Future Plan

- Develop a Kubernetes deployment script to dynamically scale replicas based on system load, enabling efficient parallel processing
- Finalize the Kubernetes deployment configuration and deploy it on Google Kubernetes Engine (GKE) for a fully containerized environment
- Measure, compare, and analyze response time, cost, latency, and scalability in both environments (Kubernetes and Cloud Run)

Evaluation Plan

The evaluation plan assesses deployment complexity, scalability, response time, cost, resource efficiency, and reliability across different environments. Metrics include setup time, auto-scaling efficiency, response time, CPU, memory, and GPU utilization efficiency, cost analysis, and fault tolerance. Methods involve load testing (k6, Locust, JMeter) to measure max throughput and auto-scaling behavior, benchmarking (curl, wrk, Google Cloud Profiler) to analyze response time and cold starts, and monitoring (Google Cloud Monitoring, Prometheus/Grafana) to track resource usage. Reliability is assessed by simulating failures and measuring downtime and recovery time. Cost analysis uses Google Cloud Billing and Profiler to compare Cloud Run's per-request pricing with Kubernetes' sustained VM costs under different load conditions.

We will compare the collected data in tables and analyze the performance differences across environments and their underlying causes.