Posture Analysis Job Offloading System (GPU-Based)

🚀Project Overview

This system processes posture images sent from Raspberry Pis using a posture analyzer container. If the master node (e.g., Intel NUC) is tainted or overloaded, the job is dynamically offloaded to a GPU-equipped worker node (e.g., Jetson Orin) based on real-time GPU usage scraped by Prometheus. The job runs via a Kubernetes Job using a custom job YAML that is patched and launched programmatically.

Directory & File Structure

```
Keda Testing/
├── build_and_push.py           # Builds and pushes Docker image, taints master node, launches monitoring
├── stop.py                     # Stops jobs, untains master, stops monitor & docker
├── cpu_monitor_and_offload.py # Queries Prometheus, launches job on underloaded GPU node
├── mqtt_posture_analyzer_with_db.py # Container entrypoint for posture analysis
├── gpu_scaler.py               # Helper script to test/query GPU stats from Prometheus manually
├── Dockerfile                  # Docker image definition for the posture analyzer
├── docker-compose.yml          # Runs posture analyzer locally with Supabase DB connection
├── requirements.txt            # Python dependencies for posture analysis
├── posture-job.yaml            # Kubernetes Job template (not patched)
├── patched-job.yaml            # Final Job file generated dynamically per deployment
```

How to Build and Run the Project

1. Pre-Requisites

- Kubernetes cluster (K3s or standard)
- Worker nodes labeled with `role=worker`
- Prometheus scraping GPU usage from each worker node
- Supabase PostgreSQL setup (credentials must match environment variables)

- Docker and Docker Hub access

---

2. Step-by-Step Instructions

A. Build & Launch System

```
python3 build_and_push.py
```

This will:

- Stop any existing container or image
- Build and push `shahroz90/posture-analyzer` image to Docker Hub
- Taint the master node (`dedicated=master:NoSchedule`)
- Start `cpu_monitor_and_offload.py` and `stop.py` in parallel

B. How Offloading Works

- `cpu_monitor_and_offload.py` queries GPU usage from Prometheus for each node
- If a node has GPU usage < 60%, it patches `posture-job.yaml` with that `nodeName`
- It writes `patched-job.yaml` and runs `kubectl apply -f patched-job.yaml`
- The Kubernetes job will run the posture analyzer on that specific worker node

C. Stopping the System

Press `ENTER` in the terminal window or run:

```
python3 stop.py
```

This will:

- Stop any docker containers
- Delete all active posture-analyzer jobs
- Kill the background monitoring script
- Untaint the master node so it can receive jobs again

Key Files Explained

1. `build_and_push.py`

- Cleans up old containers/images
- Builds multi-arch Docker image for arm64/amd64
- Pushes to Docker Hub

- Taints the master node
- Launches:
- `cpu_monitor_and_offload.py` (job offloading)
- `stop.py` (ENTER-based shutdown listener)

2. `stop.py`

- Kills `cpu_monitor_and_offload.py` process
- Deletes posture-analyzer jobs via `kubectl`
- Brings down Docker containers
- Untaints master node using:

```
kubectl taint nodes nuc node-role.kubernetes.io/master- --overwrite
```

3. `cpu_monitor_and_offload.py`

- Queries Prometheus endpoint (`localhost:9090`) for:
- `jetson_gpu_usage_percent` (AGX)
- `jetson_orin_gpu_load_percent` (Orin)
- Compares usage to a 60% threshold
- Picks lowest-usage node
- Patches `posture-job.yaml` with:

```
nodeName: <selected-node>
```

- Launches job via `kubectl apply -f patched-job.yaml`

4. `posture-job.yaml`

Template used for Kubernetes Job. Fields:

- `image: shahroz90/posture-analyzer`
- Volume mount: `/home/agx/analyzed_images`
- Environment variables: Supabase credentials
- `nodeName:` is dynamically injected

5. `mqtt_posture_analyzer_with_db.py`

- Connects to MQTT broker to receive images
- Runs MediaPipe-based posture analysis
- Saves annotated image
- Inserts metadata into Supabase

6. `gpu_scaler.py`

- Manual testing script to query Prometheus GPU usage

- Helps validate metric availability per node

7. `docker-compose.yml`

Used for running posture analyzer container locally during development

8. `requirements.txt`

Python libraries:

- opencv-python
- mediapipe
- paho-mqtt
- numpy
- matplotlib
- psycopg2-binary
- psutil
- keyboard (for ESC shutdown)

Important Notes

- GPU metrics must be exposed at Prometheus using exporters
- Image must be rebuilt and pushed every time `mqtt_posture_analyzer_with_db.py` is updated
- Job auto-cleans after `ttlSecondsAfterFinished: 60`
- Multiple jobs will not launch on same node unless previous job is complete or usage rises

---

🚜Future Improvements

- Integrate KEDA external scaler for GPU metrics
- Visualize Prometheus stats in Grafana
- Extend to RAM/CPU-aware scaling
- Use Supabase real-time feedback to display analysis stats on dashboard

---

🌐Authors

Muhammad Shahroz Abbas\ Contact: shahroz.abbas@oulu.fi

Ready for production testing and review.