Discussion on Phase 1 - Data Preparation and Phase 2 - PPO Implementation

Task

Here is a structured summary of the Google Colab notebook's contents and execution:

Notebook Summary: Hierarchical Deep Reinforcement Learning for Multi-Cloud Task Scheduling - Phase 1 and Phase 2

Overall Project Goal:
The primary objective of this project is to implement a Hierarchical Deep Reinforcement Learning (HDRL) framework for efficient and privacy-preserving task scheduling across multi-cloud environments. Phase 1 focuses on establishing the foundation and developing core components, particularly the local PPO (Proximal Policy Optimization) agents.

1. Data Acquisition and Preprocessing (Phase 1):

- **Dataset Source:** The project utilizes the Google Cloud Trace 2019 dataset, specifically "borg_traces_data.csv" from a Kaggle source (`derrickmwiti/google-2019-cluster-sample`).
- **Data Loading:** The `kagglehub` library was used to download the dataset. The notebook then guided the user to manually upload specific files (instance_events, instance_usage) to Google Drive, though the `borg_traces_data.csv` file was directly loaded from the specified path `/content/drive/MyDrive/mythesis/srilakshmi/HDRL_Research/data/raw/borg_traces_data.csv` for analysis and transformation.
- **Initial Data Analysis:** The `borg_traces_data.csv` (405,894 records, 34 columns) was analyzed for shape, data types, missing values (primarily `resource_request` , `vertical_scaling` , `scheduler` , `cycles_per_instruction` , `memory_accesses_per_instruction`), and suitability for HDRL research. It was deemed suitable due to sufficient volume, temporal information, resource data, and task characteristics.
- **Data Transformation (`KaggleGoogleTraceAdapter`):** The raw dataset was adapted into a standardized HDRL format. Key transformations included:
  - Creating `task_id` (from `collection_id`).
  - Standardizing `timestamp` , `cpu_request` , `memory_request` , `duration` , `priority` , `task_type` , `data_size` , `has_dependency` , and `parent_task_id` .

    - Generating synthetic `resource_usage_data` from the processed task events (for a sample of 1000 tasks over a simulated duration) since the Kaggle dataset is not actual instance usage data.
- **Feature Engineering and Normalization (`DataPreprocessor`):**
    - Engineered features: `hour_of_day` , `day_of_week` , `is_weekend` , `is_peak_hour` , `cpu_memory_ratio` , `resource_intensity` , `estimated_complexity` , aggregated `avg` / `max` resource usage, `priority_duration` .
    - Numerical features (excluding IDs and specific categorical/discrete ones) were normalized using `StandardScaler` .
- **Train/Validation/Test Split:** The preprocessed dataset was split into training (284,123 tasks), validation (40,589 tasks), and test (81,179 tasks) sets.
- **Saved Artifacts:** Transformed task events and resource usage data, as well as the train/val/test splits and `StandardScaler` objects, were saved to `/content/drive/MyDrive/mythesis/srilakshmi/HDRL_Research/data/raw/` and `/content/drive/MyDrive/mythesis/srilakshmi/HDRL_Research/data/processed/` , respectively. Visualizations of the preprocessed data were also generated and saved.

2. Core Components of the HDRL Local Agent (Phase 2):

- **Multi-Cloud Environment (`MultiCloudEnvironment`):**
    - **Cloud Providers:** Simulated environments for AWS, Azure, and GCP, each with distinct CPU, memory, storage capacities, cost models, energy efficiency, and base latency.
    - **State Representation:** The environment's state includes normalized available CPU/memory, utilization, task queue length, average task resource requests, active tasks, provider cost/energy parameters, latency, current step, and completion/failure rates.
    - **Reward Function:** A multi-objective reward function combines utilization, cost penalty, energy penalty, latency penalty, and a completion reward.
- **Task Segmentation Module (`TaskSegmentationModule`):**
    - **Methodology:** Utilizes K-means clustering (with 5 clusters) on key task features (`cpu_request` , `memory_request` , `data_size` , `priority` , `duration` , `resource_intensity`) to categorize tasks.
    - **Functionality:** Segments tasks into clusters and calculates a "complexity score" to determine if a task can be split.
    - **Training:** Fitted on the training dataset, demonstrating a clear distribution across the 5 segments.
- **Differential Privacy (`DifferentialPrivacyLayer`):**
    - **Methodology:** A simplified implementation using Gaussian noise with parameters `epsilon=1.0` and `delta=1e-5` to add privacy to sensitive state information.

    - **Functionality:** Adds calibrated noise to numerical data, specifically applied to aspects like `cpu_utilization`, `memory_utilization`, `resource_availability`, and `workload_characteristics` to ensure privacy.
- **PPO Actor-Critic Architecture (`PPOActorCritic`):**
    - **Model Type:** A TensorFlow Keras `Model` implementing the Actor-Critic architecture for PPO.
    - **Network Structure:** Consists of shared dense layers (256, 128 units), a dropout layer, and separate actor (action logits) and critic (value estimation) heads.
    - **Action Selection:** Uses `tfp.distributions.Categorical` for action sampling and probability calculation.
    - **Dimensions:** Defined with `STATE_DIM=20` and `ACTION_DIM=50`.
- **PPO Trainer (`PPOTrainer`):**
    - **Algorithm:** Implements the PPO algorithm with Generalized Advantage Estimation (GAE).
    - **Optimization:** Uses the Adam optimizer with a `learning_rate=3e-4`.
    - **Loss Components:** Policy loss (clipped PPO objective), value loss (MSE), and entropy loss.
    - **Update Mechanism:** Updates the policy using mini-batches from collected trajectories over multiple epochs.

3. Training Methodology:

- **Agents Initialization:** Three local PPO agents were initialized, one for each simulated cloud provider (AWS, Azure, GCP), sharing the `PPOActorCritic` architecture.
- **Workload Generation:** A synthetic workload of 5,000 tasks was generated, leveraging patterns from the preprocessed Google Cloud Trace data.
- **Training Loop:**
    - **Episodes:** Trained for 50 episodes, with each episode having a `MAX_STEPS_PER_EPISODE` of 200.
    - **Multi-Agent Training:** Each provider's agent was trained sequentially within an episode.
    - **Privacy Integration:** Differential privacy (`dp_layer.add_noise`) was applied to the state before action selection.
    - **Trajectory Collection:** Each agent collected state, action, log-probability, reward, value, and done flags for its trajectory.
    - **Policy Update:** The `PPOTrainer` updated the agent's policy using the collected trajectory if enough samples were available (batch size of 64, 10 epochs per update).
    - **Checkpointing:** Models were saved every 10 episodes.

4. Observed Results and Performance:

- **Training Completion:** The training successfully completed for all 50 episodes across the three agents.
- **Task Scheduling:** A total of 15,000 tasks were scheduled across all agents (100 completed tasks per episode for each of 3 providers over 50 episodes).
- **Performance Summary (Average over 50 Episodes):**
    - **AWS:** Avg Reward: 41.74, Avg Cost: $12.73, Avg Energy: 251.12, Avg Completed: 100.0, Avg Failed: 0.0
    - **Azure:** Avg Reward: 41.53, Avg Cost: $12.10, Avg Energy: 254.73, Avg Completed: 100.0, Avg Failed: 0.0
    - **GCP:** Avg Reward: 41.88, Avg Cost: $13.28, Avg Energy: 241.74, Avg Completed: 100.0, Avg Failed: 0.0
- **Stability:** The rewards, costs, energy, and task completion/failure rates appeared constant throughout the training, which suggests either a very stable policy was quickly found or the environment/reward function might be deterministic given the limited actions taken in each step. There were no failed tasks reported.
- **No Critical Technical Issues:** No `standard_error` or `traceback` indicated critical failures during the execution of any cell.

5. Saved Artifacts:

- **Models:** Final `PPOActorCritic` weights for AWS, Azure, and GCP (e.g., `AWS_final.weights.h5`) were saved to `/content/drive/MyDrive/mythesis/srilakshmi/HDRL_Research/models/ppo_agents/`. Checkpoints were saved every 10 episodes.
- **Task Segmenter:** The trained `TaskSegmentationModule` (K-means model and scaler) was saved as `task_segmenter.pkl` to `/content/drive/MyDrive/mythesis/srilakshmi/HDRL_Research/models/`.
- **DP Config:** The `DifferentialPrivacyLayer` configuration (epsilon, delta, noise type, noise scale) was saved as `dp_layer_config.json` to `/content/drive/MyDrive/mythesis/srilakshmi/HDRL_Research/models/`.
- **Training History and Stats:** Detailed training history (`training_history.pkl`) and a summary of statistics (`training_stats.json`) were saved to `/content/drive/MyDrive/mythesis/srilakshmi/HDRL_Research/results/phase1_part2/`.
- **Visualizations:** Training performance plots (`training_visualization.png`) were saved to `/content/drive/MyDrive/mythesis/srilakshmi/HDRL_Research/results/phase1_part2/`.

6. Future Work (Phase 3):

The notebook outlines several next steps:

- Implement a Global Coordinator (DNN-based).
- Develop Secure Multi-Party Computation (SMPC).
- Integrate Local Agents with the Global Coordinator.
- Implement an LSTM-based Resource Predictor.
- Deploy on AWS Infrastructure.

7. Research Contributions So far:

The project highlights the following contributions from Phase 1 and Phase 2:

- Privacy-preserving hierarchical DRL architecture.
- Local agents with integrated differential privacy.
- Adaptive task segmentation for multi-cloud.
- Multi-objective optimization (cost, energy, latency).
- Real-world data integration (405K Google traces).