EvaDB in Amazon SageMaker

GitHub: https://github.gatech.edu/shrn-sthsh/EvaDB-SageMaker-Container

DockerHub: https://hub.docker.com/r/shrnsthsh/evadb-sagemaker-container (CPU-only)

https://hub.docker.com/r/shrnsthsh/evadb-sagemaker-container-cuda (CUDA GPU)

Introduction

Amazon SageMaker serves as a cloud-based platform for machine learning, equipping developers with the necessary tools for creating, training, and deploying ML models. In contrast, EvaDB shares similar objectives with SageMaker but adopts a distinct approach, simplifying the complexities involved in creating ML models by abstracting various details. This abstraction makes EvaDB an asset for SageMaker users, assisting in the seamless execution of steps within the ML model pipeline.

Project Overview

To integrate EvaDB as a tool with SageMaker, it is imperative that EvaDB exists within a containerized environment and is incorporated into a SageMaker instance hosted on Amazon Elastic Container Service (ECS). Subsequently, for the effective utilization of EvaDB within SageMaker's tools, the implementation of train and serve execution modes becomes crucial.

Given the project's two assignments and the inherent logical split in the implementation, this second part of the assignment focuses on completing the implementation of the train and serve execution modes, accompanied by an illustrative example.

Implementation Details

The directory tree below serves as a high-level overview of the project. Dockerfiles are available for creating an image on ECS, which will later be integrated into the execution modes of SageMaker. The standard Dockerfile is designed for use when a CUDA-supported GPU is unavailable, but the primary intention is to utilize the CUDA-based Dockerfile.

```
.
├── Dockerfile              – Dockerfile for usage with a CPU and integrated graphics
├── Dockerfile.CUDA         – Dockerfile for usage with a CUDA GPU
├── build_and_push.sh       – Shell script to push to ECS and build
├── test/
│     ├── serve_local.sh    – Shell script for testing serve execution
│     └── train_local.sh    – Shell script for testing train execution
└── evadb_instance/
      ├── algorithm.py      – EvaDB application for usage (to be edited by user)
      ├── nginx.py          – Application configuration
      ├── serve             – Serve execution mode
      ├── train             – Train execution mode
      └── wsgi.py           – Wrapper for Gunicorn
```

The EvaDB instance itself is constructed as a Flask application, incorporating Gunicorn and NGINX for server functionality. Users have the flexibility to customize the application's behavior through the algorithm.py file, specifically in terms of training and serving. In the context of the second project, both serving and training are supported. Detailed instructions on utilizing the container can be found on the GitHub repository link.

Metrics

In assessing basic operations, several noteworthy points emerged:

1. The Docker image intended for hosting on ECS is currently approximately 2.23 GB before the implementation of training and serving handlers.
2. Conversely, the Docker image primarily employed for local testing and development measures around 314 MB in the same state with the handlers.
3. Pricing for hosting options exhibits significant variation:
    - For the more common data science workflow, training costs approximately $0.96 per hour, utilizing an ml.m4.4xlarge instance for 30 minutes per training run. This cost includes the activation of Amazon SageMaker Debugger with 2 built-in rules and 1 custom rule.

  - Larger, more customized workloads still maintain a cost well below a dollar per hour. However, total costs increase proportionally with extended time requirements.
  - Notably, the addition of EvaDB within the container resulted in minimal cost escalation. This observation may be attributed to the simplicity of the trained and tested models.

Challenges & Lessons

Similar to the previous instance, the MindsDB SageMaker container repository serves as an effective model for creating the EvaDB SageMaker container, featuring well-executed documentation. A noticeable distinction, however, lies in the usage of EvaDB compared to MindsDB; it's not a direct translation between the two.

I have not done much with Python outside of ML and DL courses and algorithms problems, so building a Flask application with different server libraries was new and challenging. And it's worth noting that, to the best of my knowledge, EvaDB does not provide support for the creation of predictor objects in the same manner as MindsDB, which utilizes specific syntax for this purpose.

Create and train predictor object on data	Call on predictor object to predict
`predictor = db.Predictor(data, ...)`	`predictor.predict(column, ...)`

Indeed, the sole method for making predictions in EvaDB involves employing the PREDICT SQL keyword, along with specifying parameters like HORIZON and FREQUENCY. This approach necessitates users to adjust the algorithm they intend to use for training more directly compared to the MindsDB version of the project.

In contrast to the MindsDB approach, where a predictor object abstracts away intricacies and allows external feeding of data and queries, the EvaDB implementation in the algorithm.py file requires users to make calls similar to those made in a Jupyter notebook. This design choice emphasizes a more direct and explicit involvement in specifying prediction-related details within the code.

Outputs

The container uses EvaDB with all the libraries required. The environment is fitted that any EvaDB application should work as expected. As part of the implementation of serve and train, I filled in algorithm.py with a test program 16, home sale forecasting.

Here is the training command:

```
cursor.query("""
    CREATE OR REPLACE FUNCTION HomeSaleForecast FROM (
        SELECT propertytype, datesold, price
        FROM postgres_data.home_sales
        WHERE bedrooms = 3 AND postcode = 2607
    )
    TYPE Forecasting
    PREDICT 'price'
    HORIZON 3
    TIME 'datesold'
    ID 'propertytype'
    FREQUENCY 'W'
""")
```

And the query:

```
print(cursor.query("SELECT HomeSaleForecast()"))
```

With the following results

index	homesaleforecast.propertytype	homesaleforecast.datesold	homesaleforecast.price
0	house	2019-07-21 00:00:00	766572.9375
1	house	2019-07-28 00:00:00	766572.9375
2	house	2019-08-04 00:00:00	766572.9375
3	unit	2018-12-23 00:00:00	417229.78125
4	unit	2018-12-30 00:00:00	409601.65625
5	unit	2019-01-06 00:00:00	402112.96875

References

The only sources I used for this project was the MindsDB implementation provided, Docker docs, and Amazon SageMaker docs.

- https://github.com/mindsdb/mindsdb-sagemaker-container
- https://docs.docker.com/
- https://docs.aws.amazon.com/sagemaker/
- https://sagemaker.readthedocs.io/en/stable/