A Comparative Study of Flyte and Kubeflow for Orchestrating Model Retraining Pipelines

Magesh Nandikkara
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
x00218707@mytudublin.ie

Introduction

Machine Learning Operations (MLOps) has emerged as a critical discipline for managing the lifecycle of ML models. MLOps has to address model drift, the degradation of model performance over time due to changes in data. Continuous training is the primary strategy to address model drift. This research presents a comparative study of two prominent Kubernetes-native orchestration platforms, Flyte and Kubeflow. The research will primarily focus on their efficacy in implementing adaptive model retraining pipelines. The study's core contribution is a quantitative and qualitative evaluation. In the quantitative analysis, the platforms are benchmarked across key performance metrics, including pipeline execution time, CPU and memory utilization, and scalability under increased data load. In the qualitative analysis, crucial operational characteristics and the developer experience are assessed. This research addresses two primary questions: (1) How can Flyte's dynamic workflow be effectively utilized for adaptive retraining logic? and (2) How does Flyte compare to Kubeflow across performance and operational metrics for this specific use case? The findings provide empirical data and insights to guide MLOps practitioners in making informed decisions when selecting an ML workflow orchestration tool.

Answering Research Question 1

The `@dynamic` workflow proved to be highly effective for implementing the adaptive retraining logic. The ability to generate a Directed Acyclic Graph (DAG) at runtime allowed conditional logic to be expressed in a Pythonic statement. This was both intuitive to write and easy to understand. The desired adaptive behaviour was correctly implemented with minimal code complexity and it did not appear to introduce any significant performance overhead. It represents a powerful architectural pattern which is well-suited for complex, decision-driven workflows where the structure of the DAG cannot be fully known at compile time.

Answering Research Question 2

Flyte is superior when it comes to developing and executing an adaptive model retraining pipeline. In Flyte, a brand new DAG is created at runtime while in KPF, the DAG is largely static, and "dynamism" can be achieved through specific control flow operators such as `dsl.ParallelFor`. Flyte's `@dynamic` is superior to KFP if massive parallelism is needed because Flyte handles aggregation natively. In terms of overall performance, Flyte was faster and more resource efficient compared to KFP. In terms of operational characteristics and developer experience, Flyte provided a simpler, more familiar, and more productive development process. Its Python native authoring, strong typing, local testing, and high-level abstractions significantly lower the cognitive load on the developer.

Experimental Design

Flyte single cluster (using `flyte-binary`) and Kubeflow Pipelines (KFP) standalone deployment paths were selected for the comparison. The Kube Prometheus Stack was chosen for monitoring because it simplified the process of setting up monitoring for Kubernetes. The `flyte-binary` was deployed using a modified Helm chart on a single node Amazon Elastic Kubernetes Service (EKS) cluster along with the Kube Prometheus Stack. KFP was installed on a single node Google Kubernetes Engine (GKE) cluster via a one-click deployment route. The Kube Prometheus Stack was once again installed using Helm on this GKE cluster. Two synthetic, binary classification datasets were designed: `train_data` and `drifted_data`. A Logistic Regression model was first trained on the `train_data` dataset. The model was then exposed to `drifted_data` which was different from `train_data`. This simulated drift and the model performance degraded which triggered model retraining. To check scalability of the ML pipeline, `train_data` and `drifted_data` were augmented by three times to 15 rows each. Logistic Regression model was chosen for the retraining pipeline because it trains incredibly fast which is perfect for repeated pipeline executions. The retraining pipeline in Flyte was implemented using the `flytekit` SDK. In the base use case, the pipeline was designed to train a model, measure the model accuracy, introduce drift, detect the drift, retrain the model on the drifted data, tune the model with a default hyperparameter value of one and return model accuracy. The base case pipeline was executed with five hyperparameter values to understand how the pipelines work during hyperparameter tuning. Prometheus was configured to scrape utilisation metrics from all pods. The experimental design is shown in Figure 1.

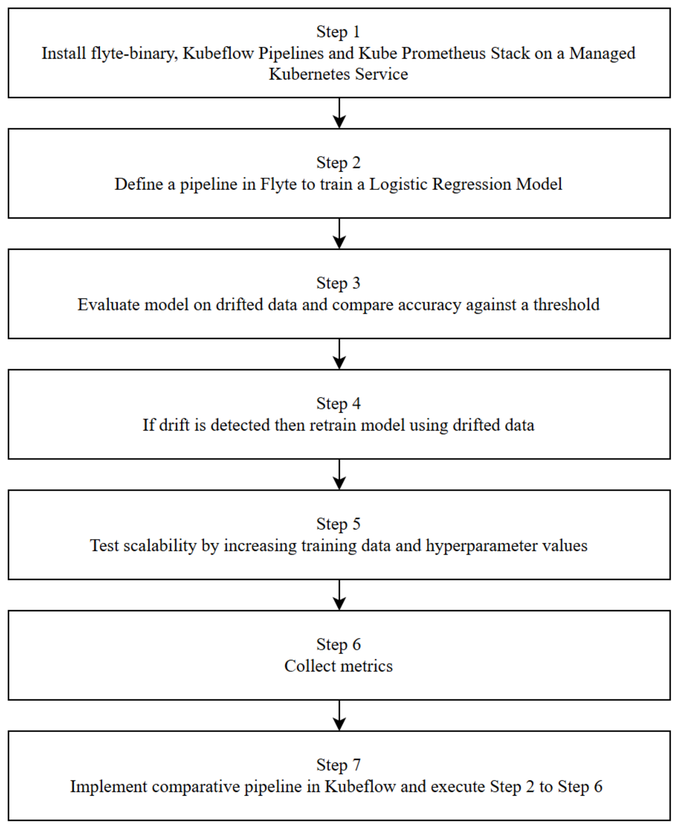

Figure 1: Overview of the experimental design.

Experiment Findings

Metric	Flyte (Mean ± Std Dev)	Kubeflow (Mean ± Std Dev)
Total Pipeline Time (s)	118.3 ± 1.79	196.4 ± 3.14
Retraining and Tuning Time (s)	52.7 ± 1.10	16.6 ± 1.62
Peak Total Memory (GiB)	0.370 ± 0.03	1.988 ± 0.05
Average Total Memory (GiB)	0.247 ± 0.02	1.849 ± 0.02
Peak Total CPU (mCPU)	10.79 ± 0.55	313.5 ± 49.85
Average Total CPU (mCPU)	7.07 ± 0.38	140.5 ± 22.33
Reliability (Success Rate %)	100%	100%

Table 1: Comparison of Flyte vs Kubeflow Base Case

Metric	Flyte (Mean ± Std Dev)	Kubeflow (Mean ± Std Dev)
Total Pipeline Time (s)	120.5 ± 1.43	190.6 ± 5.43
Retraining and Tuning Time (s)	54.2 ± 0.87	16.4 ± 0.49
Peak Total Memory (GiB)	0.38 ± 0.02	2.055 ± 0.04
Average Total Memory (GiB)	0.25 ± 0.01	1.897 ± 0.03
Peak Total CPU (mCPU)	11.05 ± 0.31	313.8 ± 27.81
Average Total CPU (mCPU)	7.277 ± 0.21	126.67 ± 19.51
Reliability (Success Rate %)	100%	100%

Table 2: Comparison of Flyte vs Kubeflow Scaling Case

Metric	Flyte (Mean ± Std Dev)	Kubeflow (Mean ± Std Dev)
Total Pipeline Time (s)	127.8 ± 0.87	192.5 ± 5.18
Retraining and Tuning Time (s)	62.6 ± 0.80	16.2 ± 0.40
Peak Total Memory (GiB)	0.51 ± 0.04	2.103 ± 0.03
Average Total Memory (GiB)	0.34 ± 0.02	1.937 ± 0.02
Peak Total CPU (mCPU)	14.94 ± 0.63	371.2 ± 39.02
Average Total CPU (mCPU)	9.832 ± 0.41	142.8 ± 12.88
Reliability (Success Rate %)	100%	100%

Table 3: Comparison of Flyte vs Kubeflow Hyperparameter Tuning

Conclusions and Future Work

Flyte consistently outperformed Kubeflow in quantitative benchmarks. The qualitative analysis revealed that Flyte has a substantial advantage over Kubeflow in terms of developer experience. However, for large organisations looking to implement a highly customisable and all-encompassing ML platform, Kubeflow remains a powerful option. The research highlighted that the "best" MLOps platform ultimately depends on the context.

Future work should focus on expanded comparison, multi-cluster analysis, larger datasets, cost-benefit analysis, integration with other MLOps tools and sophisticated retraining triggers.

QR Code for Recording