THE UNIVERSITY OF HONG KONG

FINAL YEAR PROJECT

HOGAR: Deployment and Management Framework for Modern Machine Learning

PROJECT PLAN

Guoyang Cui

Department of Computer Science

Supervised by

Dr. Heming Cui

Department of Computer Science

September 27, 2019

1. Introduction

In recent years, practice has shown that more and more training data and larger models tend to produce better precision in a variety of applications. However, for ordinary machine learning researchers and practitioners, learning large models from a large amount of data is still a challenge because it usually requires a lot of computing resources. In the era of big data, distributed machine learning can provide more computing power [4], so that it becomes more important than ever. That being so, in mission-critical online service, multiple machine learning models become more pervasive and are often deployed in a distributed graph manner [2], where each operator can be deployed on different physical hosts.

However, most of machine learning models focus mainly on executing and training models [1] and there are not enough tools for users to easily define such distributed graph, visualize pipelines running in production and monitor the process, for example, handling the real-time process and adjust batching for each training epoch.

In this paper, I present HOGAR, an efficient deployment and management framework for distributed ML serving graphs. In general, HOGAR consists of 2 parts, the deployment and management. First, the deployment framework is a library, which is written in python and supports defining and parsing graph information into configuration file to be feed in EARG system. On the other hand, management framework is based on and a web-based monitoring interface for visualizing pipelining machine learning tasks.

In this project plan, I will introduce the backgrounds of the project in section 2, focusing on EARG and Airflow. In section 3, I try to show the initial objective of this project while in section 4 the proposed methodology is shown in greater details. Finally, section 5 will discuss about the tentative

schedule of the project.

2. Background

This section will provide detailed information of two related works mentioned in the introduction

2.1. EARG

EARG is a robust and efficient deployment and recovery system for distributed ML serving graphs [2]. It presents a new recovery abstraction called Autonomous Recovery Graph (ARG) and its distributed runtime protocol. Below, figure 1 shows the dependency graph of one distributed machine learning models. However, from user/developer point of view, the interrelation between each worker nodes would be as clear as presented without this architecture graph. In contrary, the process of defining worker nodes and dependency graph is quite complicated.

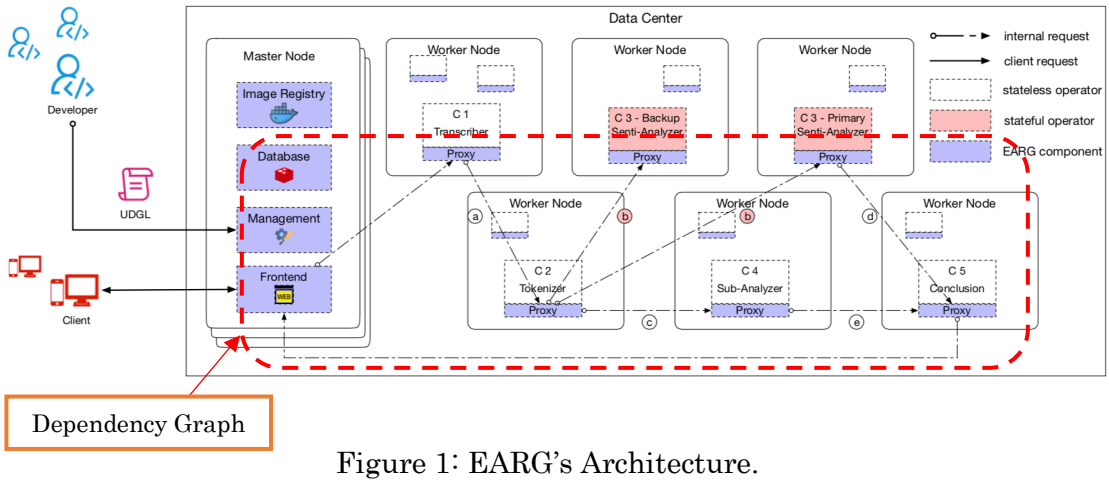

Figure 1: EARG's Architecture.

2.2. Airflow.

The airflow executes the task on a set of worker threads while following the

specified dependencies [4]. DAG's dependencies and their current status for a specific run can be showed quite comprehensively. However, Airflow is not specified for machine learning tasks, which cannot provide views for batching or provide detailed machine learning related metrics for each node during training.

3. Project Objective

HOGAR aims to provide a simple User Defined Graph Language (UDGL), specify its service graph, and monitor web Interface for Machine learning models.

3.1. Development of UDGL

A library will be devised to translate a directed graph of operators into configuration files that can be used in EARG system. There are mainly two steps: How to design the graph and how to parse it into files.

In fact, HOGAR is developed and compatible for of EARG system, which presents recovery abstraction called Autonomous Recovery Graph (ARG) [2] and its distributed runtime protocol. So, the first goal for this development of UDGL library will be translation for ARG.

In figure 2, a simple sentiment analysis model is designed by current UDGL (which will be discussed in section 4). In this senatio case, a library to translate this Autonomous Recovery Graph into configuration file, recurring the library to support basic functionality, such as editing operator and dependency.

Then, more varieties of graph should be compatible with UDGL by adding new features of flow representation and operator factors. Moreover, a deeper research will be conducted about how state of the art machine

learning serving platform defines their configuration files so that HOGAR will be suited for a broader range of systems.

Algorithm 1: Sentiment Analysis UDGL

```
meta
    name SentiAnalyzer
    version 3

operator Trancriber
    version 2, image senti/trancriber:v2
    batch 32, stateful No
operator Tokenizer
    version 1, image senti/tokenizer:v1
    batch 32, stateful No
operator Senti-Analyzer
    version 3, image senti/sentiment:v3
    batch 1, stateful Yes
operator Sub-Analyzer
    version 3, image senti/subject:v3
    batch 32, stateful No
operator Conclusion
    version 1, image senti/conclude:v1
    batch 32, stateful No

logical flow
    distribute
        Trancriber→Tokenizer
        Tokenizer→Senti-Analyzer, Sub-Analyzer
    reduce
        Senti-Analyzer, Sub-Analyzer→Conclusion
```

Figure 2: EARG's Architecture.

3.2. Development of Monitoring web Interface

As for model management, HOGAR will develop a web-based monitoring interface for visualizing pipelining. The first goal is providing real-time graphic views including revealing running request and dead host. This task will mainly focus on the application on top on EARG system since tasks information are hidden inside the data center in figure 1 and each system has a different architecture of design.

4. Methodology

4.1. Development of UDGL

Based on Autonomous Recovery Graph, this project will further research on User Defined Graph Language and its interrelation with the architecture of distributed machine learning model. For UDGL design, more research will be done on distributed machine learning scheduling and directed acyclic graph analysis. After literature review, new improvement will be suggested to the generalization of UDGL. For the UDGL implementation, EARG system's config file will be taken as a model for the output of UDGL.

4.2. Development of Monitoring web Interface

Web interface will be developed in Django, a high-level python framework and link to parameters in EARG systems. Meanwhile, more literature review about evaluating machine learning performance will be done and more metric for monitor will be suggested.

5. Tentative Schedule

Time Periods	Tasks
September	• Meeting and discussing with supervisor • Project plan formulating • Project website designing
October - Mid November	• Producing the first version of UDGL library which can • Producing prototype of monitoring website
Mid November - December	• Producing more metrics of monitoring website

	• Maintaining and adding features to the UDGL library • Interim report writing
January - February	• Producing the first version of UDGL library • Producing the first version of monitoring website
March - April	• Evaluation of monitoring website • Final report finishing • Final presentation • Poster design and exhibition

6. Conclusion

Undeniably, deploying machine learning task on a distributed system is never an easy task. While many tasks have been done to accelerate training process but not for the deployment and monitoring procedure, I will attempt to make possible improvement on model deployment process and provide developers a clearer and real-time access to model performance.

7. Reference

[1] D. Crankshaw, X. Wang, G. Zhou, M. Franklin, J. Gonzalez and I. Stoica. Clipper: A low-latency online prediction serving system. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, PAGES 613-627, 2017.

[2] H. Cui, "EARG: Robust and Efficient Recovery for Distributed Machine

Learning Serving Graphs," Research submission in progress, Dept. Computer Science, Hong Kong Univ., 2019.

[3] C. Olston, N. Fiedel, K. Gorovoy, J. Harmsen, L. Lao, F. Li, V. Rajashekhar, S. Ramesh, and J. Soyke. Tensorflow-serving: Flexible, high-performance ml serving. *arXiv preprint arXiv: 1712.06139*, 2017.

[4] Airflow. https://airflow.apache.org/