HORIZONTALLY SCALABLE ML PIPELINES WITH A FEATURE STORE

Alexandru A. Ormenișan [1] **Mahmoud Ismail** [1] **Kim Hammar** [2] **Robin Andersson** [2] **Ermias Gebremeskel** [2] **Theofilos Kakantousis** [2] **Antonios Kouzoupis** [2] **Fabio Buso** [2] **Gautier Berthou** [2 3] **Jim Dowling** [1 2] **Seif Haridi** [1 3]

ABSTRACT

Machine Learning (ML) pipelines are the fundamental building block for productionizing ML models. However, much introductory material for machine learning and deep learning emphasizes ad-hoc feature engineering and training pipelines to experiment with ML models. Such pipelines have a tendency to become complex over time and do not allow features to be easily re-used across different pipelines. Duplicating features can even lead to correctness problems when features have different implementations for training and serving. In this demo, we introduce the Feature Store as a new data layer in horizontally scalable machine learning pipelines.

1 INTRODUCTION

In this demonstration, we introduce an open-source data platform with support for machine learning (ML) pipelines, Hopsworks, where every stage in the pipeline can be scaled out to potentially hundreds of servers, enabling faster training of larger models with more data. An end-to-end machine learning pipeline covers all stages in the production and operation of ML models, from data ingestion and preparation, to experimentation, to training, to deployment and operation. Our demonstration will have the added twist of introducing a new data management layer, the Feature Store. We will show how the Feature Store simplifies such pipelines, providing an API for data engineers to produce features and an API with which data scientists can easily select features when designing new models. Our Feature Store is based on open-source frameworks, see Figure 3, using Apache Spark to compute features, Apache Hive to store feature data, and MySQL Cluster (NDB) for feature metadata. Our Feature Store also supports generating training data, from selected features, in native file formats for TensorFlow, Keras, and PyTorch. Our Feature Store is integrated in a multi-tenant security model that enables the governance and reuse of features as first-class entities within an organization. We also provide Python API support for both the Feature Store and for building pipelines in our framework. APIs ease the programmer's burden when discovering and using services such as Kafka, the Feature Store, our REST API, and model serving, as well as when performing distributed model training and hyperparameter optimization.

[1]KTH Royal Institute of Technology, Sweden [2]Logical Clocks AB, Sweden [3]RISE AB, Sweden. Correspondence to: Alexandru A. Ormenișan <aaor@kth.se>.

Proceedings of the 2^{nd} SysML Conference, Palo Alto, CA, USA, 2019.

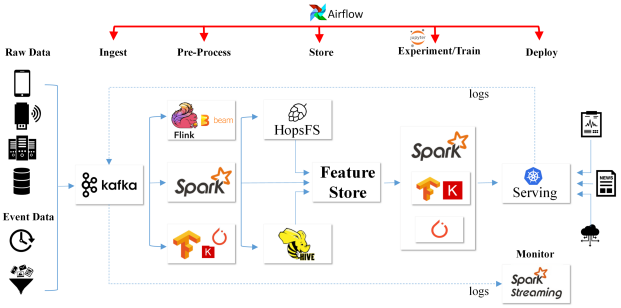

Figure 1. End-to-End ML Pipeline, orchestrated by Airflow

2 END-TO-END ML PIPELINES

In ML pipelines on Hopsworks, see Figure 1, horizontal scalability is provided using Apache Spark for feature engineering, Apache Hive for storing feature data, HopsFS (Niazi et al., 2017) (a next generation HDFS) as the distributed filesystem for training data, Ring-AllReduce using TensorFlow for distributed training, and Kubernetes to serve models. We also support Kafka for storing logs for ML models served from Kubernetes, and Spark streaming applications for processing those logs in near-realtime. Distributed training and hyperparameter optimization with TensorFlow and PyTorch use PySpark to distribute computations, replicated conda environments to ensure Python libraries are available at all hosts, and YARN to allocate GPUs, CPUs, and memory to applications. Our pipelines are driven and managed by a scalable, consistent, distributed shared memory service built on MySQL Cluster.

2.1 Demonstration

In our live demonstration, we will use a managed version of Hopsworks running from a web browser at www.hops.site. We will start with an Airflow job orchestrating an entire ML pipeline that consists of a series of different jobs, all implemented in Jupyter notebooks and invoked from Airflow by an operator that calls the REST API to Hopsworks, see

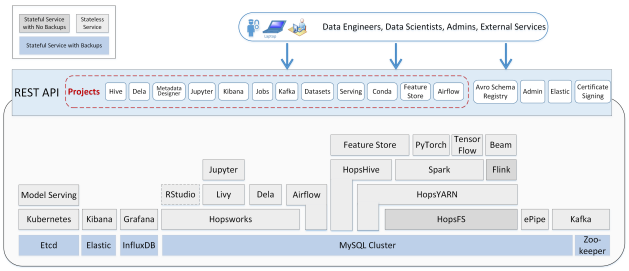

Figure 2. Hopsworks Data and ML Platform Architecture

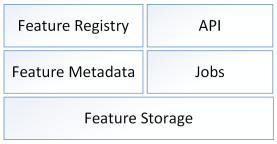

Figure 3. Feature Store Architecture

Figure 2. We will then go through the Jupyter notebooks individually. We will start with discovering and downloading a ML dataset to work with - Hopsworks supports peer-to-peer sharing of large datasets. We will then write/run a Spark job to engineer features from the dataset. We also support Beam/Flink as jobs, so we can show an additional data validation job using TensorFlow Extended (TFX) (Baylor et al., 2017). After the feature engineering stage, the features will be published to our Feature Store. The next stage is where a Data Scientist will write a notebook to select the features that will be used to train a model. This job will output training data in a chosen file format (such as .tfrecords, .csv, .parquet, .npy, .hdf5) to HopsFS. The next job in the pipeline is the experimentation or training step. Here, the data scientist designs (or evolves) a model architecture and discovers good hyperparameters by running lots of parallel experiments on GPUs. Similar to MLflow (Zaharia et al., 2018), Hopsworks manages and archives ML experiments for easy reproduction and archiving. The next stage is to train a model with the chosen hyperparameters using lots of GPUs and RingAll-Reduce for distributed training. The output of the training step is a model, saved to HopsFS. Now, the model can be analyzed and validated (such as using TFX model analysis), and then deployed for serving using TensorFlow Serving (with many options available, such as A/B testing and number of replicated instances behind a load-balancer). Finally, we will show client applications using the deployed model, and a streaming job will be run to monitor model predictions, notify of any anomalies, and collect more training data (by storing predictions that are later linked with outcomes).

2.2 Teaching Platform and Interactive Demo

Our research organization runs a managed version of Hopsworks with over 1000 users. This managed platform, see Figure 2, is also used for lab and project work in Deep Learning and Big Data courses at a large technical university. Hopsworks has a User Interface and strong security support, built around TLS certificates for applications, users, and services. Similar to Jupyter Hub, students can write interactive programs in Jupyter notebooks. In contrast to JupyterHub, users can run jobs that access up to 100s of TBs of data on 1000s of cores, and use 10s of GPUs for model experimentation or training. Hopsworks also includes a project-based multi-tenancy environment, so students can work in groups on sandboxed data. Our managed platform provides many popular datasets for machine learning, including Imagenet, Wikipedia, Youtube-8M, and Kinetics.

In the interactive demo, participants will be invited to use their laptops and an Internet connection to create accounts on our managed platform. With an account, a user can create a project, link in a large machine learning dataset, and run pre-canned notebooks or end-to-end machine learning pipelines. Participants will be given enough quota to use 100s of CPUs, and several GPUs, if they so choose.

2.3 Summary

We will demonstrate a data and ML platform, Hopsworks, that supports the design, debugging, and running of deep learning pipelines at scale. All stages of the pipeline can be scaled out horizontally, as the platform manages the whole stack from resource management (YARN with GPU support) to API support for running distributed training and hyperparameter optimization experiments.

Acknowledgment

This work was funded by the Swedish Foundation for Strategic Research projects “Smart Intra-body network, RIT15-0119” and “Continuous Deep Analytics, BD15-0006”, and by the EU Horizon 2020 project AEGIS under Grant Agreement no. 732189.

References

Baylor, D., Breck, E., Cheng, H.-T., Fiedel, N., Foo, C. Y., Haque, Z., Haykal, S., Ispir, M., Jain, V., Koc, L., et al. Tfx: A tensorflow-based production-scale machine learning platform. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1387–1395. ACM, 2017.

Niazi, S., Ismail, M., Haridi, S., Dowling, J., Grohsschmiedt, S., and Ronström, M. Hopsfs: Scaling hierarchical file system metadata using newsql databases. In *FAST*, pp. 89–104, 2017.

Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., et al. Accelerating the machine learning lifecycle with mlflow. *Data Engineering*, pp. 39, 2018.