LOGICAL CLOCKS

MSc projects 2020/2021

Who are Logical Clocks and what is Hopsworks?

Logical Clocks are a venture-capital funded startup building with roots in KTH and RISE that are developing and selling support for the open-source Hopsworks platform. Hopsworks is Europe's leading platform for Data Science (DatSci Technology of the Year 2019), and it enables the design and operation of AI applications at scale. You can write end-to-end machine learning pipelines that can be horizontally scaled at all stages, and that use our file system, HopsFS (winner of the IEEE Scale Prize 2017), the world's highest performance distributed hierarchical file system.

An Analytics Microservice for a Feature Store

Hopsworks is an open-source platform for the development and operation of AI applications at scale, that includes the industry's first and most advanced feature store for machine learning. The Hopsworks Feature store is a dual database, with Apache Hive as the scalable SQL layer for storing large volumes of features for creating train/test data and MySQL Cluster as the feature serving layer, providing features to online models at low latency.
Although the Hopsworks Feature Store's primary API is a Dataframe API, it is also queryable using SQL. However, Hopsworks provides no support for running interactive queries and analytics on the feature store. In this project, you will design and develop support for interactive analytics of the feature store as a microservice on Kubernetes.
To this end, Reddash is an open-source web application for designing dashboards and running interactive SQL queries. Reddash supports containerized operation on Kubernetes. Your project will develop support for interactive analytics of the Hopsworks Feature Store using Reddash.
You are expected to have good programming skills, in particular in Java, Python, and some knowledge of Javascript is beneficial. Good Linux skills are also useful.

Required skills

- Java, Python, Linux, Javascript (bonus), Kubernetes (bonus)

References

- https://github.com/getredash/redash
- https://www.hopsworks.ai/
- https://github.com/logicalclocks/hopsworks

DeepSpeed and ZeRO on Maggy

In 2020, Microsoft released the current state-of-the-art framework for scaling data-parallel training of deep neural networks: the DeepSpeed library and Zero Redundancy Optimiser

(ZeRO). ZeRO-2, the successor, now allows training large AI models with 170 billion parameters.
Maggy is the leading open-source framework for distributed machine learning on Spark. It is an open-source framework for writing Tensorflow/Keras and PyTorch programs that can be run in Python program on your laptop, and run with minimal changes on a large cluster for parallel hyperparameter tuning, parallel ablation studies, and distributed (data-parallel) training.
In this project, you will design and develop support for ZeroSpeed in the Maggy framework.
This project is suitable for candidates with strong Python programming skills, knowledge of machine learning and, if possible, also distributed computing.

Required skills

- Deep understanding of the Python programming language, Linux

References

- https://www.microsoft.com/en-us/research/blog/zero-2-deepspeed-shattering-barriers-of-deep-learning-speed-scale/?OCID=msr_blog_deepspeed2_build_fb
- https://www.microsoft.com/en-us/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/
- https://github.com/microsoft/DeepSpeed
- https://databricks.com/session_na20/from-python-to-pyspark-and-back-again-unifying-single-host-and-distributed-deep-learning-with-maggy

AutoLoggers and MLflow for Maggy

Hopsworks provides support for machine learning (ML) experiments. That is, it can automatically track the artifacts, graphs, performance, logs, metadata, and dependencies of your ML programs. Many of you already know about platforms like MLflow, so why should you read about Hopsworks Experiments? Because you do not have to rewrite your TensorFlow/PyTorch/Scikit-learn programs to get **tracking and distributed ML for free**.
Currently, Hopsworks and Maggy supports only auto-logging for Keras models. Other open-source projects like MLFlow or Ray's Tune have shown that Monkey Patching is a convenient way to modify user written training code at run time without altering the intended functionality by the user.
With this project we want to extend the auto-logging capabilities of the Hopsworks Platform and Maggy to Sklearn, Xgboost, LightGBM, PyTorch, and possibly Sparks' MLlib. Furthermore, we would like to achieve API compatibility, so user can easily integrate their experiments in Maggy with MLflow.
Up to this point, there is no research exploring the full possibilities of this method with regard to machine learning experiments as well as itss performance impact. For example, how big is the

overhead of monkey patching Keras' .fit() function to include a callback for automatic model savepoint logging.

Required skills

- Deep understanding of the Python programming language, Linux

References

- https://www.logicalclocks.com/blog/hopsworks-ml-experiments
- https://mlflow.org/docs/latest/tracking.html#automatic-logging
- https://stackoverflow.com/questions/5626193/what-is-monkey-patching

Data Distribution Search

Deep Neural Networks rely on very large datasets for training. The order and distribution of how the samples are then presented to the learning algorithm at training time can influence the final generalization error of the learned model. Most training pipelines deploy a simple shuffle heuristic for each epoch to generate random batches. With this project we would like to study the effects of data splits, oversampling factors and overall data distribution within batches on the model performance and training efficiency - or how Andrej Karpathy puts it: "The subtle art of composing batches of data during training", and how we can automate it.
In order to do so, we will have to address several research goals:

1. What is the state-of-the-art to generate well distributed training data sets?
2. Are there additional statistics available to compose better batches/training data sets, possibly enabled by a feature store? Can we learn policies for batch creation?
3. How can we leverage a Feature Store to precompute needed statistics over the available data, in order to make efficient use of it at the time of training data set creation and model training?
4. Having the needed statistics available can we leverage the information efficiently during the training time using for example tensorflow's tf.data API or comparable APIs in PyTorch?
5. Introducing overhead in terms of slower training, does it get amortized by better results? And how far can we reduce the overhead by pre-computing statistics?

Required skills

- Deep understanding of the Python programming language, Linux

References

- https://twitter.com/karpathy/status/1175138379198914560
- https://arxiv.org/abs/1905.12737
- https://arxiv.org/abs/1704.07433
- https://github.com/songhwanjun/ActiveBias
- https://www.tensorflow.org/guide/data_performance

- http://jmlr.org/papers/volume20/18-789/18-789.pdf
- https://twitter.com/mlpowered/status/1312087614296477696

Develop the UI for a data intensive product

Data scientists and machine learning engineers are now using tools to enhance their processes. It can include automation, data exploration, or simply a better collaboration between engineers. While dealing with a huge amount of data, they started to use Feature Stores: which is a tool to easily manage data in one single place. It is now mostly accessible through code or command lines. This has a lot of limits: a difficult and time-consuming exploration of the data, it requires a long setup to be used at its best or even a hard exploration for non-coders. We are currently making it accessible through a user interface developed in React. It means, a much faster familiarisation with data for data scientists coming on a new project and visual representation of data and its metadata. This product will reach the production level by the end of the year.

We are looking for a product passionate UI engineer for helping us in this mission. You will work alongside a designer and an experienced React developer. Remote work is accepted and the time schedule is flexible.

Required skills: javascript, front end dev with modern UI tool (React, Angular, Vue)

HopsFS for Kubernetes

Implementing a Container Storage Interface (CSI) driver for HopsFS so we can leverage HopsFS as a persistent storage in K8S via Persistent Volumes. The current set of available drivers are these ones.

Required skills
- Java, Kubernetes, Linux, (C/C++ a bonus)

References

Distributed Database Kernel Development for an Online Feature Store

You will get to work the world-renowned database luminary Mikael Ronström on the storage engine for MySQL Cluster, NDB (Network Database). We are using NDB as the storage engine for the Hopsworks Feature Store, and there are some limitations in NDB that we would like to address in this project. For example, NDB has a maximum of 512 columns, and we would like to increase this number significantly. NDB is also designed primarily for operation in on-premises environments with predictable resource availability and no virtual machine migration challenges to deal with.You will need excellent knowledge of Linux and programming experience in C.

Required skills

- Deep understanding of C or C++, Strong Linux background, Database knowledge.

References

- http://www.it.uu.se/edu/course/homepage/dbastekn2/vt11/Multithread_NDB_arch.pdf
- http://mikaelronstrom.blogspot.com/2020/02/use-cases-for-mysql-ndb-cluster-80.html
- https://www.logicalclocks.com/blog/feature-store-the-missing-data-layer-in-ml-pipelines

UX research and prototype a Feature Store

Machine Learning engineers are dealing with an increasing amount of data. They have data coming from different sources, following different formats, and it is constantly updating. On top of that, this data has to be shared across projects and users. That challenge is currently a very boiling topic in the world of AI. Some solutions have been explored, we call them Feature Stores. An interesting example is the in house solution from Uber: Michelangelo. Still, there is no proper product for solving these challenges. We are developing this product which has never been designed before. This Feature Store will help machine learning professionals to import and explore data while sharing a single source of truth for building their models.

Thus, for inventing this new user interface, we are looking for product passionate UX designer. You will work alongside designers and developers. Remote work is accepted and the time schedule is flexible.

Required skills: HCI research, user-centric method, prototyping, Figma basis

AutoML Data Wrangler for the Feature Store

AutoML is a recent term “automated machine learning” that implies it automates the whole machine learning process. Traditionally, AutoML refers to building architectures to optimize the training of models. There is truth in the cliche that 90% of data science is feature engineering. This project is about extending AutoML to feature engineering.
Feature engineering with a Feature Store involves transforming data from backend data systems - data warehouses, data lakes, and files into numerical data that is then cached in a Feature Store, from which data scientists can easily generate train/test data for training models.

MSc projects 2020/2021

Hopsworks currently supports PySpark and Pandas as frameworks for feature engineering. However, many feature engineering tasks are well-known transformations, such as one-hot encoding strings, normalizing numerical data, and imputing missing values. Many of the feature engineering transformations can be predicted with high probability, which means our Feature Store could suggest possible transformations on database columns, for example.
This project will develop a domain-specific language for feature transformations and a UI for suggesting and implementing feature engineering. The user will be able to import data from external data sources, and your UI will show them information about the features (columns in database-speak) and suggestions for how to transform them. The actual feature transformations will be executed as PySpark applications. This project would be the first-in-the-world automated feature engineering tool for a feature store.

References

- https://www.trifacta.com/blog/transform-by-example-your-data-cleaning-wish-is-our-command/ (Example of such a data wrangling tool - a very good one)
- https://www.logicalclocks.com/blog/feature-store-the-missing-data-layer-in-ml-pipelines
- https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning