May 12, 2017

ZIYAD MUHAMMED MOHIYUDHEEN

TU Berlin

Matriculation No: 387646

Serving Large Machine Learning Models on Apache Flink

IT4BI Master Thesis

Table of Contents

[1. Introduction 3](#_Toc482963191)

[2. Background and related work 4](#_Toc482963192)

[Machine learning 4](#_Toc482963193)

[Model Serving 4](#_Toc482963194)

[Model Serving Frameworks 5](#_Toc482963195)

[1. Velox 5](#_Toc482963196)

[Apache Flink 6](#_Toc482963197)

[Queryable State in Flink 6](#_Toc482963198)

[Parameter Servers 6](#_Toc482963199)

[Large Machine Learning Models 6](#_Toc482963200)

[ALS 6](#_Toc482963201)

[3. Design and Implementation 7](#_Toc482963202)

[4. Experiments and Results 7](#_Toc482963203)

[5. Conclusions 7](#_Toc482963204)

[6. Future Work 7](#_Toc482963205)

[7. References 7](#_Toc482963206)

# Introduction

# Background and related work

## Machine learning

Machine learning is the field of computer science which gives computers the ability to learn and predict without being explicitly programmed [1]. Machine learning techniques are employed in various applications in today’s digital world, which includes recommendations on online shops, weather predictions, filtering spam emails, serving ads on the web, detecting credit card fraud etc. to name just a few. With advancement in technologies for large scale data processing (known as big data systems or platforms), it is now possible to train the machine learning models on the whole data sets, at affordable cost and time. This has led to more feature rich models which perform better.

## Model Serving

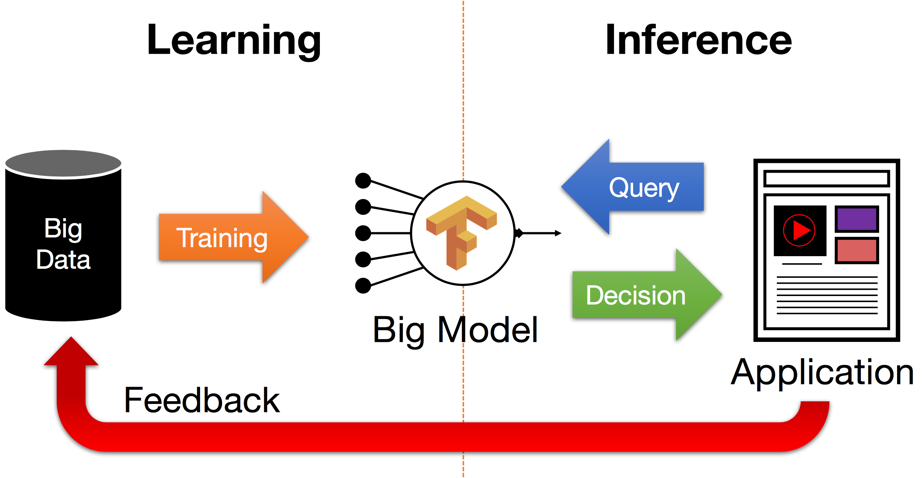


Figure 2. Model Serving of Large machine learning models [7]

As the data and features increases, the size of the machine learning model also increases. One of the critical components in machine learning applications in big data systems is the deployment of these large models, serving different applications which uses this model for predictions. The ability to update the models with the newly observed data after the initial training is also important.

Often big data frameworks are used only for the training of the models, and then it is moved to the serving layer, which typically resides outside of the big data framework. Incremental updates to the model is often not performed, instead a retraining of the entire model is carried out in scheduled frequencies. This is not ideal for many applications, and the research community is now trying to address this issue to have end-to-end machine learning systems within the big data frameworks, which can perform training and updating of the models and serve various external applications.

The two approaches towards model serving are Eager and Lazy serving. In Eager, we compute and materialize all the predictions and access them in query time. Whereas in Lazy serving, prediction computation happens in real-time. The advantage of Eager serving is that we can use offline training frameworks for efficient batch prediction computation, and use traditional data serving systems for serving the applications. The obvious disadvantage is that it demands frequent and heavy computations to reflect model updates, and can serve only when the set of possible queries is limited.

The lazy serving model has the advantage that computation is performed only for the necessary queries which need not be bound to a known set, and rapid updates to the model is possible. The disadvantage is that lazy serving is complex and involves substantial computation overhead of the serving system. It requires low and predictable latency from the model serving layer.

In general, Lazy serving is ideal for large machine learning models, if we can achieve the low latency (< 10 ms) and high throughput.

## Model Serving Frameworks

In this section we describe some of the pioneer works related to serving of large machine learning models.

***TODO***: LASER, Velox / Clipper, TensorFlow Serving, Oryx, Prediction I/O.

### Velox

Velox [5] try to address the deployment of large machine learning models in Berkeley Data Analytics Stack (BDAS). Velox is a model management system which facilitates online model management, maintenance and serving. Velox proposes an initial offline batch training to produce the model using Apache Spark, and the trained model is then deployed using Velox.

The primary components in Velox architecture are,

1. Velox model manager: orchestrates the computation and maintenance of pre-declared machine learning models, allows incorporating feedback and new observations, continuously evaluate model performance and trigger the retraining as the quality of the model decreases.
2. Velox model predictor: low-latency, up-to-date prediction interface to the deployed model for any application which uses this model. Model predictor also uses some pre-materialization strategies for efficient serving of the model.

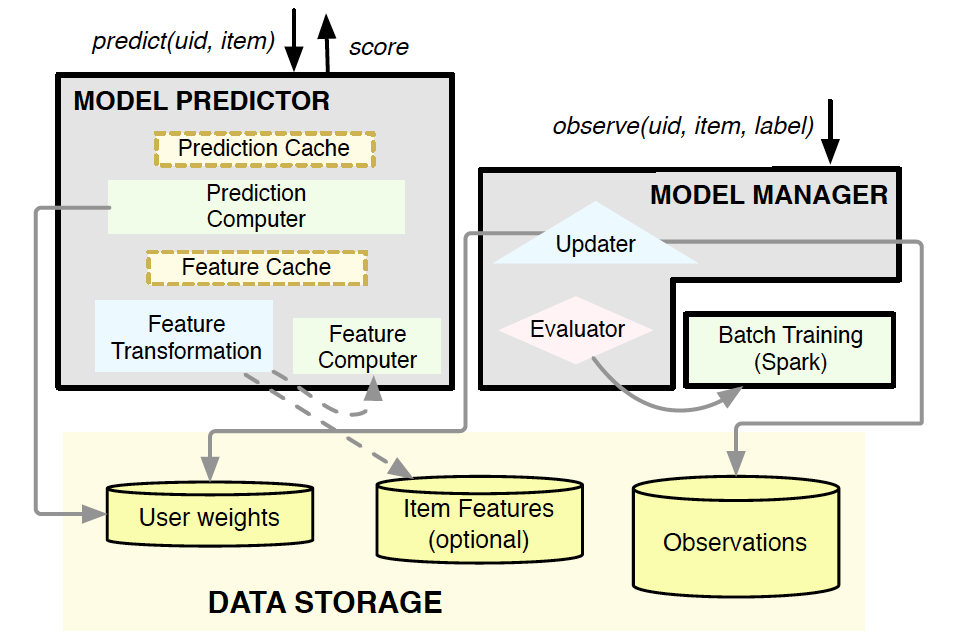


Fig.1: Velox Architecture [5]

In the prototype, Velox supports matrix factorization models which uses latent factor vectors. An example is a song prediction recommender model. The user and song latent factors of the model are deployed on Tachyon, the default storage system used in Velox, and an API is provided to the applications to query these latent factors, in order to obtain the prediction for any user-song pair. As the new observations arrive through the observe API, Velox model manager updates the user models, and also check the model quality in order to determine if a retraining is required. Only user vectors are updated in an online fashion, where as the batch retraining updates the item vectors and potentially user vectors as well.

Velox prediction service uses a Feature cache and a Prediction Cache in order to reduce the remote data access. The prediction cache is used to cache the final prediction for a given user-item pair where as the feature cache efficiently partition and replicate the materialized feature tables. Velox make use of the fact that while the total number of items might be large, item popularity often follows a Zipfian distribution, resulting in more frequent access of only a few items. So Velox suggests caching only the ‘hot’ items using a simple cache eviction strategy such as LRU, which will have a high hit rate.

For adding the new users, Velox uses the recent estimate of the average of the existing user weight vectors. And in order to overcome the feedback loops, Velox rely on form of a bandit algorithm, which assign an uncertainty score to each item on top of its predicted score.

## Apache Flink

Apache Flink [2] is a unified batch and stream processing framework for big data, which is often referred to as the 4th generation of big data frameworks, offering real time stream processing as well as batch processing on top of its streaming dataflow engine [3]. Its dedicated support for iterative computations including bulk and delta iterations [3] makes it ideal for many scalable machine learning applications. The machine learning library of Flink, called FlinkML [4] is a relatively new effort in the Flink community, with a vision to add scalable machine learning algorithms which are easy to use and with minimal glue code in end-to-end machine learning systems [4].

Flink community is also actively thinking about having a serving layer on Flink, called Flink-MS [6] for various machine learning models potentially trained in Flink or any other platforms. Flink MS will try to address the requirements such as live model updating, latency guarantees, bandit frameworks (for competing model selection) and model versioning. At the time of writing this paper, Flink-MS is only in the initial discussion phase.

### Queryable State in Flink

## Parameter Servers

## Large Machine Learning Models

### ALS

# Design and Implementation

# Experiments and Results

# Conclusions

# Future Work

# References

1. <https://en.wikipedia.org/wiki/Machine_learning>
2. Apache Flink, Accessed on 28/12/2016, <https://flink.apache.org>
3. P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, Apache flink: Stream and batch processing in a single engine, Data Engineering, p. 28, 2015.
4. Apache FlinkML, Accessed on 28/12/2016, <https://ci.apache.org/projects/flink/flink-docs-release-1.2/dev/libs/ml>
5. D. Crankshaw, P. Bailis, J. E. Gonzalez, H. Li, Z. Zhang, M. J. Franklin, A. Ghodsi, and M. I. Jordan. The missing piece in complex analytics: Low latency, scalable model management and serving with velox. In Conference on Innovative Data Systems Research (CIDR), 2015.
6. <https://docs.google.com/document/d/1CjWL9aLxPrKytKxUF5c3ohs0ickp0fdEXPsPYPEywsE/edit#heading=h.j2r3xzajbsl>
7. <https://ucbrise.github.io/cs294-rise-fa16/assets/slides/prediction-serving-systems-cs294-RISE_seminar.pdf>