Sprinklr-Serving

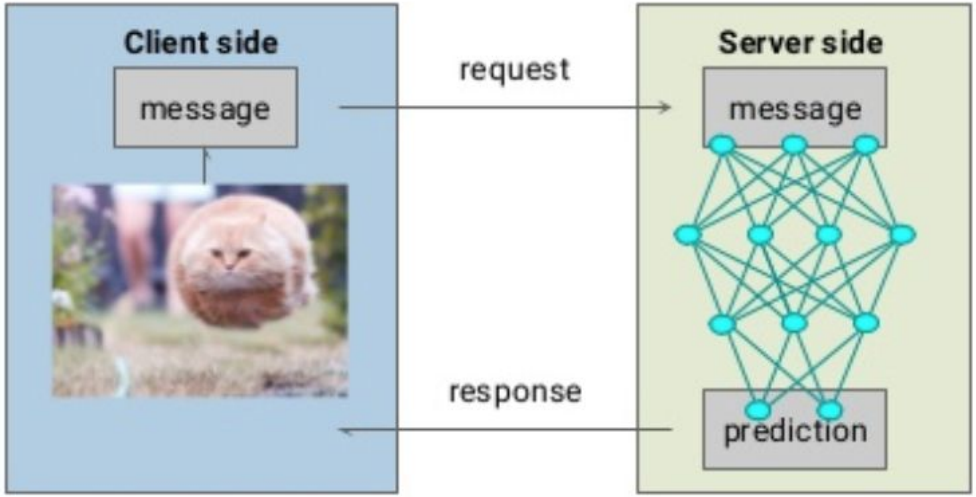
Flexible, High-Performance ML Serving

Course : PC-423 & PC-424 B.Tech Project
Mentor : Prof. Bakul Gohel

Ayub Subhaniya - 201501433

What is Serving?

- Serving is how you apply a model, after you have trained it.



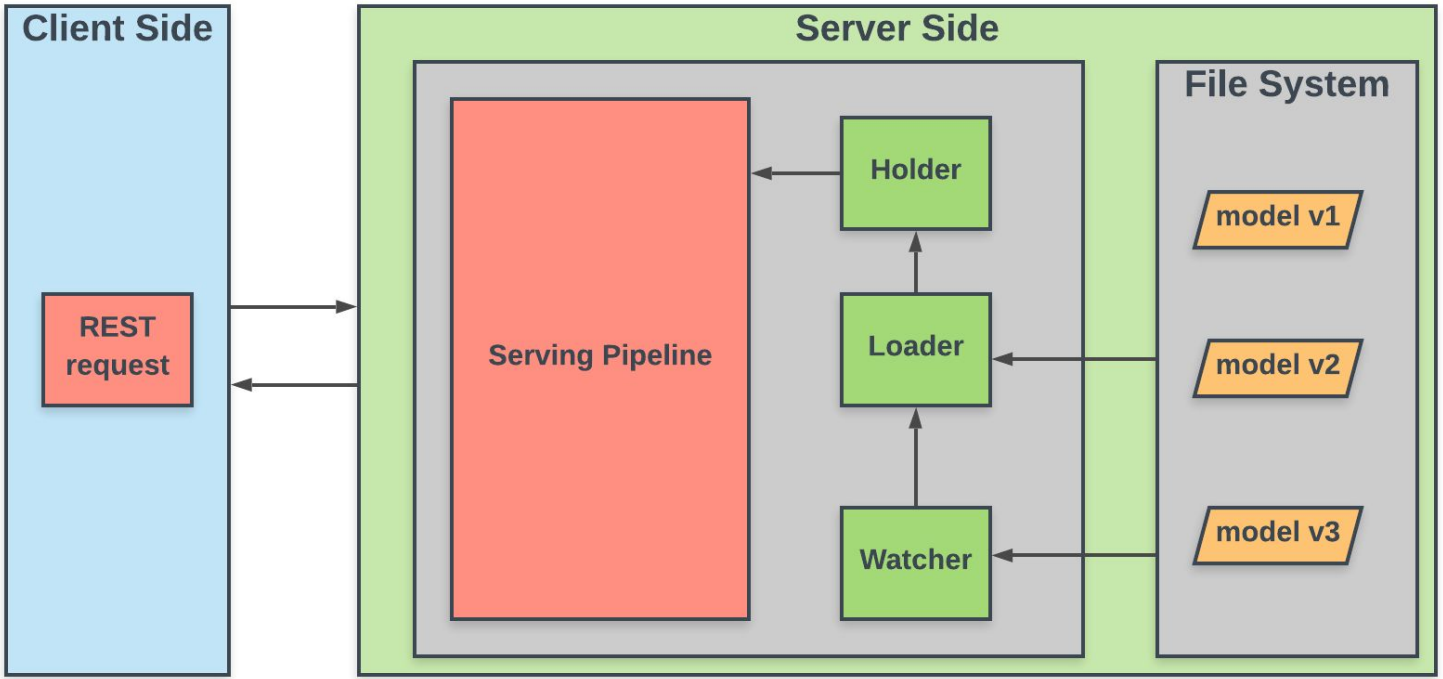
Serving ML model

- Machine learning (ML) lies at the heart of a growing number of important problems in many fields.
- While there is vast literature and software devoted to training ML models, there has been little systematic effort around deploying trained models in production.
- One of the biggest challenges is that serving a model (i.e. accepting requests and returning a prediction) is only part of the problem. There is a long list of adjacent requirements. These include, for example:
    - Model versioning
    - Retraining (automatically or on-demand)
    - Preprocessing input before predictions
    - Scaling infrastructure on-demand

What is Sprinklr-Serving?

- Flexible, high performance serving system for machine learning models, designed for production environment.
- **Goals**
  - Online, low-latency
  - Multiple models, multiple versions
  - Should scale nodes with demand

Main Architecture

- 

Current Serving Framework

-

Approach

- Problems in existing serving framework
  - **Rolling updates**- update ML models without server downtime
  - ScaleSecond problem was to optimize it to serve more number of request per unit time.
- Main functionality of module is divided into two parts,
  - **Model Life-Cycle Management** which handles rolling update
  - **Serving Pipeline** which handles serving concurrent requests.

Model Life-Cycle Management

- Models are trained and exported to external storage (amazon s3).
- Model life-cycle management consists of three main subcomponents dependent on each other.
    - **Watcher**, which monitor's external storage system (amazon s3) to discover new model and their versions.
    - **Loader**, which loads new model into local system when new version of a model is detected by Watcher
    - **Holder**, which holds latest model loaded by Loader
- When new model is successfully served, old model is removed from memory.
- While loading new model into local system it should not affect resources available to current system serving old model.
- Check working of model before making new holder for it. It might happen that corrupted file is uploaded on s3, in this cases we should not create holder for this.

Model Serving Pipeline

- **Pipelining-** the technique of decomposing a repeated sequential process into sub-processes, each of which can be executed efficiently on a special dedicated autonomous module that operates concurrently with the others.
- Serving code for ML models is a single sequential process. It can be divided into three sub-processes, given a input as a batch of messages,
    - **Pre-processing -** which do the required cleaning of input before feeding into ML model
    - **Prediction-** perform prediction using ML model using cleaned input
    - **Post-processing-** perform post-processing and decoration on predicted output by ML model.
- Each components of pipeline are executed **asynchronously.**
- **Asynchronous model-** allows multiple things to happen at the same time. Calls to a function is non-blocking.

Model Serving Pipeline

- **Pipelining-** the technique of decomposing a repeated sequential process into sub-processes, each of which can be executed efficiently on a special dedicated autonomous module that operates concurrently with the others.
- Serving code for ML models is a single sequential process. It can be divided into three sub-processes, given a input as a batch of messages,
    - **Pre-processing -** which do the required cleaning of input before feeding into ML model
    - **Prediction-** perform prediction using ML model using cleaned input
    - **Post-processing-** perform post-processing and decoration on predicted output by ML model.
- Each components of pipeline are executed **asynchronously.**
- **Asynchronous model-** allows multiple things to happen at the same time. Calls to a function is non-blocking.

Optimization

- **Cache**, ML models are roughly of size more than 500 MB. And in our case there would be multiple models at a time in memory. This will eat up all available RAM. So if request for any model has not arrived since some time interval, that model removed from memory and stored in a cache. When request for that model is arrived again it is loaded from cache. This will cause a slight latency in request. So it is a trade-off between speed and memory.

Deploying Model

- For each deployment two configuration files (YAML format) are needed.
- One of this configuration file contains the pipeline information, order of components and some specific property related to a component like for pre- processor component we should specify which pre-processor’s to use.
    - pre- processor like remove urls, remove punctuations
- Other configuration file contains information regarding server like batch size, i.e. break the input batch into groups of batch size and then feed into model.

Setting Up Server

-

Observation

- From the above graphs we have seen that as we increase the no. of channels we are getting better accuracy.
- Also, as we increase number of trials we get better accuracy.
- Benefits of our model
  - Flexibility
    - To add new subject we don’t have to train whole model.
    - We just have to train n binary models of new subject with existing subject and append a new row and column in our matrix of binary models.
  - Simplicity

Thank You