Cranberries

Machine Learning models serving system

Samsung Seoul R&D Center

Presenter: Egor Suvorov

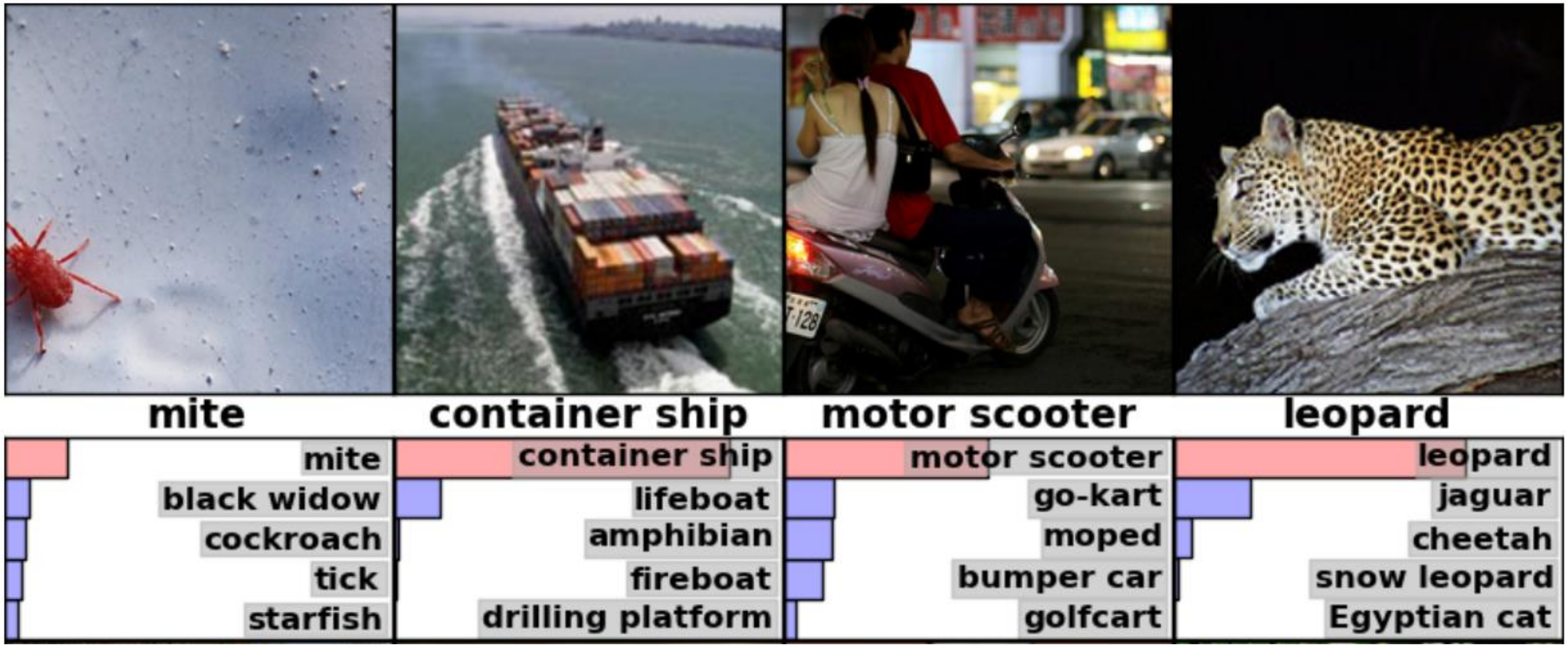
THE PROBLEM

What is Machine Learning about

- Method of solving hard problems without explicitly specifying the algorithm
- *Model* is a highly configurable algorithm
- Model can be *trained* on some set of data
- Trained model is typically *served* to end users
- Think of trained model as of an algorithm

Example problem: ImageNet

- Popular database with ~15’000’000 images
- Images are labeled, sometimes ambiguous



“Inception v3” model

- Model by Google which “solves” ImageNet
  - Implemented using TensorFlow
- 25 millions of adjustable parameters
- 5 billions operations per one prediction

Technical challenges

- Training
  - Very computationally expensive
  - Takes long time
  - Optimized for throughput
- Serving
  - Computationally expensive
  - Optimized for latency per one request
  - Not so well-learned, not much best practices

THE PROJECT

Goals: functionality, architecture

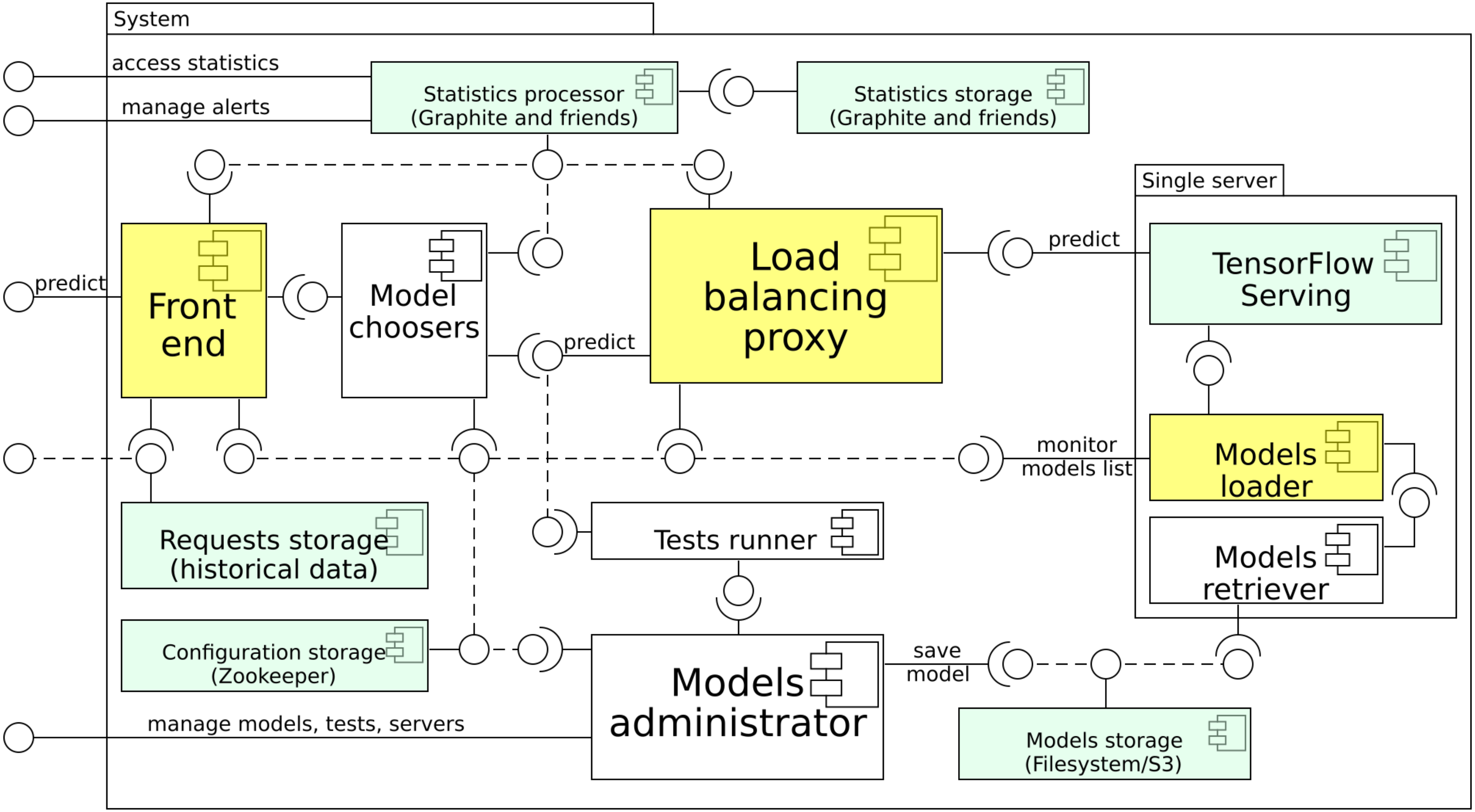
- Trained ML models → web services
- Scale horizontally, serve models of any size, serve independent “services”
- Re-use existing popular open-source solutions:
  - Expensive computations (e.g. SciPy, TensorFlow)
  - Model storage format
  - Operations (e.g. Docker, Graphite, Zookeeper)
- Low coupling between components

Basis: TensorFlow Serving

- Introduced by Google in February 2016
- C++ framework for serving ML models
  - Supports TensorFlow models out-of-the-box
  - Has APIs to add other types of models
  - Implements Google’s best practices for serving
    - Mini-batching, loading policies, computation reuse
- Sample gRPC server is provided
- No load balancing out-of-the-box

Architectural choices

- Multiple decoupled services
  - Load balancer, front-end, administration ui
- Coordination is done via shared storage
- Direction communications employ gRPC
- Do not reinvent operational solutions:
  - Apache Zookeeper for shared storage
  - Graphite for monitoring
  - Docker and Docker Compose for containers


System
access statistics
manage alerts
Statistics processor
(Graphite and friends)
Statistics storage
(Graphite and friends)
Single server
predict
Front
end
Model
choosers
Load
balancing
proxy
predict
TensorFlow
Serving
predict
monitor
models list
Models
loader
Models
retriever
Requests storage
(historical data)
Tests runner
Configuration storage
(Zookeeper)
Models
administrator
save
model
Models storage
(Filesystem/S3)
manage models, tests, servers

Implementation plans

- Implement a single component
  - And transfer ownership to teammate
- Make stubs for other components for demo

Implemented and not

- Model loader
  - Loads subset of models on a specific server
  - Reports status of models
  - Unit and simple integration tests
  - Handles some connectivity failures
- Can be improved
  - More connectivity failures handling
  - Fuller integration tests

Internship summary

- Challenges
  - Not much success stories in public
  - Implementation of model loader is highly tied to TensorFlow Serving
- New skills
  - Learned about ML serving systems
  - Designed a new one and presented to colleagues
- https://github.com/kimbyungsang/cranberries/