Clipper
A Low-Latency Online Prediction Serving System

Presenter: Alexey Tumanov
Daniel Crankshaw
Corey Zumar, Guilio Zhou, Xin Wang
Michael Franklin, Joseph Gonzalez, Ion Stoica

BDD/RISE Mini-Retreat 2017
May 2, 2017
Big Data
Complex Model
Training
Learning

Project Adam: Building an Efficient and Scalable Deep Learning Training System
Trishul Chilimbi
Yutaka Suezu
Johnson Apacible
Karthik Kalyanaraman
Microsoft Research

ABSTRACT
Large deep neural networks require large datasets to be trained effectively. However, it is often extremely time consuming to obtain large datasets. In this paper, we present a new deep learning system called Project Adam that is designed to make training large models as easy as training small models. Project Adam provides a platform for training deep models and allows users to easily train large models.

Caffe: Convolutional Architecture for Fast Feature Embedding
Yanqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, Trevor Darrell

GraphLab: A New Framework For Parallel Machine Learning
Yucheng Low
Joseph Gonzalez
Aapo Kyrola
Danny Bickson
Carlos Guestrin
Joseph M. Hellerstein
UC Berkeley

Abstract
GraphLab is a high-level framework and computing environment for implementing large-scale machine learning algorithms. The framework is designed to support a wide range of machine learning algorithms, from simple linear regression to complex deep learning models. The framework provides a set of tools for data processing, model training, and model evaluation.

GraphX: Graph Processing in a Distributed Dataflow Framework
Joseph E. Gonzalez, Raymon S. Xin, Aikid Dave, Daniel Crankshaw
Michael J. Franklin, Ion Stoica
UC Berkeley

Abstract
GraphX is a distributed graph processing system that is designed to support large-scale graph algorithms. The system is implemented as a dataflow framework, and it provides a set of tools for graph data processing, model training, and model evaluation.

Parameter Server for Distributed Machine Learning
Mu Li, Li Zhi, Zhihao Yang, Aaron Li, Pei Xin, David G. Andersen, and Alexander Smola

Abstract
We propose a parameter server framework to solve distributed machine learning problems. Both data and weight are distributed to client nodes, where server algorithms are used to train the parameters. The framework manages asynchronous data communications between clients and servers. Flexible consensus models, elastic scalability and fault tolerance are supported by this framework. We present algorithms and theoretical analysis for challenging scenarios and non-smooth problems. To demonstrate the scalability of the proposed framework, we show experimental results on real data with billions of parameters.
Big Data → Training → Complex Model
Learning Produces a Trained Model

Query

Decision

“CAT”
Big Data → Training → Model → Serving

Learning

Serving

Query → Decision → Application
Big Data

Learning

Model

Serving

Query

Decision

Application

Prediction-Serving for interactive applications

Timescale: ~10s of milliseconds
Google Translate

Serving

140 billion words a day\(^1\)

82,000 GPUs running 24/7

Prediction-Serving Raises New Challenges
Prediction-Serving Challenges

Support low-latency, high-throughput serving workloads

Large and growing ecosystem of ML models and frameworks
Models getting more complex

- 10s of GFLOPs [1]

Deployed on critical path

- Maintain SLOs under heavy load

Using specialized hardware for predictions

Prediction-Serving Challenges

Support low-latency, high-throughput serving workloads

Large and growing ecosystem of ML models and frameworks
Large and growing ecosystem of ML models and frameworks
Big Companies Build One-Off Systems

**Problems:**

- Expensive to build and maintain
- Highly specialized and require ML and systems expertise
- Tightly-coupled model and application
- Difficult to change or update model
- Only supports single ML framework
Large and growing ecosystem of ML models and frameworks

Difficult to deploy and brittle to manage

Varying physical resource requirements
But most companies can’t build new serving systems...
Use existing systems: Offline Scoring

Batch Analytics
Use existing systems: Offline Scoring

Batch Analytics
Use existing systems: Offline Scoring

Look up decision in datastore

Application

Low-Latency Serving
Use existing systems: Offline Scoring

Look up decision in datastore

Problems:

- Requires full set of queries ahead of time
- Small and bounded input domain
- Wasted computation and space
- Can render and store unneeded predictions
- Costly to update
- Re-run batch job

Low-Latency Serving
**Prediction-Serving Challenges**

Support low-latency, high-throughput serving workloads

Large and growing ecosystem of ML models and frameworks
How does Clipper address these challenges?
Clipper Solutions

- Simplifies deployment through layered architecture
- Serves many models across ML frameworks concurrently
- Employs caching, batching, scale-out for high-performance serving
Clipper Decouples Applications and Models

Applications

Predict ⬅️ ➤️ RPC/REST Interface ⬅️ ➤️ Feedback

Clipper

RPC ⬅️ ➤️ Model Container (MC)
RPC ⬅️ ➤️ MC
RPC ⬅️ ➤️ MC
RPC ⬅️ ➤️ MC

RPC ⬅️ ➤️ ...

Applications: Clipper, Predict, Netflix, MC, RPC/REST Interface, Feedback
Clipper Architecture

Applications

Predict

RPC/REST Interface

RPC

Observe

Model Abstraction Layer

Provide a common interface to models while bounding latency and maximizing throughput.

Model Selection Layer

Improve accuracy through bandit methods and ensembles, online learning, and personalization.

Clipper

Applications

RPC

Observe
Clipper Architecture

Applications

Predict

RPC/REST Interface

Observe

Clipper

Selection Policy

Model Selection Layer

Caching

Model Abstraction Layer

Adaptive Batching

RPC

Model Container (MC)

RPC

MC

RPC

MC

RPC

MC

RPC

MC

RPC

MC

RPC

MC
Selection policies supported by Clipper

- Exploit multiple models to estimate confidence
- Use multi-armed bandit algorithms to learn optimal model-selection online
- Online personalization across ML frameworks

*See paper for details [NSDI 2017]*
**Batching to Improve Throughput**

- **Why batching helps:**
  - A single page load may generate many queries
  - Hardware Acceleration
  - Helps amortize system overhead

- **Optimal batch depends on:**
  - Hardware configuration
  - Model and framework
  - System load
Adaptive Batching to Improve Throughput

Why batching helps:
- A single page load may generate many queries

Hardware Acceleration
- Helps amortize system overhead

Clipper Solution:
Adaptively tradeoff latency and throughput...
- Inc. batch size until the latency objective is exceeded (Additive Increase)
- If latency exceeds SLO cut batch size by a fraction (Multiplicative Decrease)

Optimal batch depends on:
- hardware configuration
- model and framework
- system load
Conclusion

- Prediction-serving is an important and challenging area for systems research
  - Support low-latency, high-throughput serving workloads
  - Serve large and growing ecosystem of ML frameworks
- Clipper is a first step towards addressing these challenges
  - Simplifies deployment through layered architecture
  - Serves many models across ML frameworks concurrently
  - Employs caching, adaptive batching, container scale-out to meet interactive serving workload demands
- Beyond academic prototype to build a real, open-source system

https://github.com/ucbrise/clipper

crankshaw@cs.berkeley.edu