Prediction Serving

what happens after learning?

Joseph E. Gonzalez
Asst. Professor, UC Berkeley
jegonzal@cs.berkeley.edu
Co-founder, Dato Inc.
joseph@dato.com
Learning

Big Data

Training

Big Model

Timescale: minutes to days
Systems: offline and batch optimized
Heavily studied ... major focus of the AMPLab
Big Data

Learning

Training

Big Model

Inference

Query

Decision

Application
**Inference**

**Big Model**

**Query**

**Decision**

**Application**

**Timescale:** ~10 milliseconds

**Systems:** *online* and *latency* optimized

*Less studied …*
Learning

Big Data

Training

Timescale: hours to weeks

Systems: combination of systems

Inference

Decision

Application

Feedback
Learning

Big Data

Big Model

Training

Adaptive (~1 seconds)

Inference

Query

Responsive (~10ms)

Prediction

Responsive (~1 seconds)

Application

Feedback
**Key Insight:**

*Decompose models into fast and slow changing components*
Hybrid Offline + Online Learning

Update feature functions offline using batch solvers
- Leverage high-throughput systems (Tensor Flow)
- Exploit slow change in population statistics

Update the user weights online:
- Simple to train + more robust model
- Address rapidly changing user statistics

\[ f(x; \theta)^T \mathbf{w}_u \]
Common modeling structure

\[ f(x; \theta)^T w_u \]

Matrix Factorization

Items

Users

Ratings

Deep Learning

Ensemble Methods

Input
Big Data Training

Learning

Inference

Slow Changing Model

Fast Changing Model

Query

Decision

Application

Feedback
Big Data

Learning

Training

Fast Changing Model per user

Inference

Query

Decision

Application

Slow Changing Model

Feedback

Fast Feedback

Slow Feedback
Velox Online Learning for Recommendations (20-News Groups)

Online Updates: 0.4 ms
Retraining: 7.1 seconds

>4 orders-of-magnitude faster adaptation given sufficient offline training data
Velox Online Learning for Recommendations (20-News Groups)

Partial Updates: 0.4 ms
Retraining: 7.1 seconds

>4 orders-of-magnitude faster adaptation
Big Data Training

Learning

Slow Changing Model

Fast Changing Model per user

Velox

Inference

Query

Decision

Application

Feedback

Fast Feedback

Slow Feedback
**VELOX**: the Missing Piece of BDAS

**Berkeley Data Analytics Stack**

- Spark Streaming
- BlinkDB
- Graph Frames
- Keystone ML
- Spark SQL
- GraphX
- MLLib
- Spark
- Mesos
- Tachyon
- HDFS, S3, ...

**BDAS**
**VELOX**: the Missing Piece of BDAS

- **Spark Streaming**
- **BlinkDB**
- **Graph Frames**
- **Keystone ML**
- **Spark SQL**
- **GraphX**
- **MLLib**
- **Spark**
- **Mesos**
- **Tachyon**
- **HDFS, S3, ...**
**VELOX**: the Missing Piece of BDAS

### Learning
- Spark Streaming
- BlinkDB
- Graph Frames
- Keystone ML
- Spark SQL
- GraphX
- MLLib

### Management and Serving
- Velox

### Infrastructure
- Spark
- Mesos
- Tachyon
- HDFS, S3, ...

---

**Berkeley Data Analytics Stack**
**VELOX Architecture**

**Fraud Detection**

- Keystone ML
- MLLib
- Spark

**Content Rec.**

**Velox**

Single JVM Instance
VELOX Architecture

Fraud Detection

Content Rec.

Personal Asst.

Robotic Control

Machine Translation

Keystone ML

MLLib

Spark

Velox

Single JVM Instance

Velox

Netfliex

Caffe

TensorFlow

Dato

Caffe

KALDI

Theano

scikit

mxnet

dmlc
VELOX as a Middle Layer Arch?

- Fraud Detection
- Content Rec.
- Personal Asst.
- Robotic Control
- Machine Translation

Generalize Velox?

- theano
- Dato
- Create
- Caffe
- TensorFlow
- scikit learn
- KALDI
- VW
- mxnet
Clipper
A Low-Latency Online Prediction Serving System

Daniel Crankshaw
Xin Wang
Michael Franklin
Joseph E. Gonzalez
Ion Stoica
Clipper *Generalizes* Velox Across ML Frameworks

- Fraud Detection
- Content Rec.
- Personal Asst.
- Robotic Control
- Machine Translation

Clipper

Frameworks and Tools:
- Theano
- Dato
- Create
- Caffe
- TensorFlow
- scikit-learn
- keystoneML
- VW
- MXNet
- KALDI
Key Insight:
The challenges of prediction serving can be addressed between end-user applications and machine learning frameworks.

As a result, Clipper is able to:

- **hide complexity**
  - by providing a common prediction interface

- **bound latency** and **maximize throughput**
  - through approximate caching and adaptive batching

- **enable robust online learning** and **personalization**
  - through generalized split-model correction policies

**without modifying** machine learning frameworks or end-user applications
Clipper Design Goals

Low and **bounded** latency predictions
  ➢ interactive applications need reliable latency objectives

Up-to-date and personalized predictions **across models** and **frameworks**
  ➢ generalize the split model decomposition

Optimize **throughput** for performance under heavy load
  ➢ single query can trigger many predictions

**Simplify** deployment
  ➢ serve models using the original code and systems
Clipper Architecture

- **Fraud Detection**
- **Content Rec.**
- **Personal Asst.**
- **Robotic Control**
- **Machine Translation**

**Clipper**

- theano
- Dato Create
- Caffe
- TensorFlow
- scikit learn
- dmlc mxnet
- VW
- KALDI
Clipper Architecture

Applications

Predict

RPC/REST Interface

Observe

Clipper

theano  Dato  Create  TensorFlow  scikit  learn  VW

Keystone  ML  Caffe  TensorFlow  dmlc  mxnet  KALDI
Clipper Architecture

Applications

Predict ▲

RPC/REST Interface

Observe ▲

Clipper

RPC

Model Wrapper (MW)

KeystoneML

RPC

MW

Caffe

RPC

MW

RPC

MW

RPC

MW

...
Clipper Architecture

Predict $\uparrow$

RPC/REST Interface

Observe $\uparrow$

Clipper

**Correction Layer**

*Improve accuracy through ensembles, online learning and personalization*

**Model Abstraction Layer**

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

Applications

Predict

RPC/REST Interface

Observe

Clipper

Correction Policy

Correction Layer

Approximate Caching

Model Abstraction Layer

Adaptive Batching

RPC

Model Wrapper (MW)

Keystone

Caffe

MW

RPC

MW

RPC

MW

RPC

MW

Correction Policy

Model Wrapper (MW)

Keystone

Caffe

MW

RPC

MW

RPC

MW

Correction Policy

Model Wrapper (MW)

Keystone

Caffe

MW

RPC

MW

RPC

MW

Correction Policy

Model Wrapper (MW)

Keystone

Caffe

MW

RPC

MW

RPC

MW
Provides a unified generic prediction API across frameworks

- **Reduce Latency** → Approximate Caching
- **Increase Throughput** → Adaptive Batching
- **Simplify Deployment** → RPC + Model Wrapper
<table>
<thead>
<tr>
<th>Model Wrapper (MW)</th>
<th>Model Abstraction Layer</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeystoneML</td>
<td>Approximate Caching</td>
<td></td>
</tr>
<tr>
<td>Caffe</td>
<td>Adaptive Batching</td>
<td></td>
</tr>
<tr>
<td>TensorFlow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scikit-learn</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Common Interface ➔ Simplifies Deployment:

- Evaluate models using original code & systems
- Models run in separate processes
  - Resource isolation
Common Interface \(\rightarrow\) Simplifies Deployment:

- Evaluate models using original code & systems
- Models run in separate processes
  - Resource isolation
  - Scale-out

**Problem:** frameworks optimized for **batch processing** not **latency**
Adaptive Batching to Improve Throughput

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

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

Clipper Solution:
- *be as slow as allowed*…
- Inc. batch size until the latency objective is exceeded *(Additive Increase)*
- If latency exceeds SLO cut batch size by a fraction *(Multiplicative Decrease)*

Hardware Acceleration
Helps amortize system overhead
Tensor Flow Conv. Net (GPU)

Throughput (Queries Per Second)

Latency (ms)

Optimal Batch Size

Latency Deadline

P99 Latency

Throughput

Batch Sizes (Queries)

Avg. Latency

Throughput (Queries Per Second)

Optimal Batch Size
Comparison to TensorFlow Serving

**Takeaway:** Clipper is able to match the average latency of TensorFlow Serving while reducing tail latency (2x) and improving throughput (2x)
Approximate Caching to Reduce Latency

- Opportunity for caching
  Popular items may be evaluated frequently

- Need for approximation

Clipper Solution: Approximate Caching

apply locality sensitive hash functions

High Dimensional and continuous valued queries have low cache hit rate.
Clipper Architecture

Applications

Predict ↑

Clipper

RPC/REST Interface

Observe ↑

Correction Policy

Correction Layer

Approximate Caching

Model Abstraction Layer

Adaptive Batching

RPC

Model Wrapper (MW)

RPC

MW

Caffe

RPC

MW

scikit
Goal:

Maximize **accuracy** through **ensembles**, **online learning**, and **personalization**

Generalize the **split-model** insight from Velox to achieve:

- **robust predictions** by combining multiple models & frameworks
- **online learning** and **personalization** by correcting and personalizing predictions in response to feedback
Correction Policy

Improves prediction accuracy by:
- Incorporating real-time feedback
- Managing personalization
- Combine models & frameworks
  - enables frameworks to compete
# Improved Prediction Accuracy (ImageNet)

<table>
<thead>
<tr>
<th>System</th>
<th>Model</th>
<th>Error Rate</th>
<th>#Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caffe</td>
<td>VGG</td>
<td>13.05%</td>
<td>6525</td>
</tr>
<tr>
<td>Caffe</td>
<td>LeNet</td>
<td>11.52%</td>
<td>5760</td>
</tr>
<tr>
<td>Caffe</td>
<td>ResNet</td>
<td>9.02%</td>
<td>4512</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>Inception v3</td>
<td>6.18%</td>
<td>3088</td>
</tr>
</tbody>
</table>

sequence of pre-trained state-of-the-art models
## Improved Prediction Accuracy

<table>
<thead>
<tr>
<th>System</th>
<th>Model</th>
<th>Error Rate</th>
<th>#Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caffe</td>
<td>VGG</td>
<td>13.05%</td>
<td>6525</td>
</tr>
<tr>
<td>Caffe</td>
<td>LeNet</td>
<td>11.52%</td>
<td>5760</td>
</tr>
<tr>
<td>Caffe</td>
<td>ResNet</td>
<td>9.02%</td>
<td>4512</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>Inception v3</td>
<td>6.18%</td>
<td>3088</td>
</tr>
<tr>
<td>Clipper</td>
<td>Ensemble</td>
<td>5.86%</td>
<td>2930</td>
</tr>
</tbody>
</table>

5.2% relative improvement in prediction accuracy!
Cost of Ensembles

Increased Load

- **Solutions:**
  - **Caching** and **Batching**
  - **Load-shedding** correction policy can prioritize frameworks

Stragglers

- e.g., framework fails to meet SLO
- **Solution:** **Anytime** predictions
  - Correction policy must render predictions with missing inputs
  - e.g., built-in correction policies **substitute expected value**
Evaluation of Throughput Under Heavy Load

Takeaway: Clipper is able to gracefully degrade accuracy to maintain availability under heavy load.
Conclusion

Clipper sits between applications and ML frameworks to

- simplify deployment
- bound latency and increase throughput
- and enable real-time learning and personalization across machine learning frameworks
Ongoing & Future Research Directions

- Serving and updating RL models
- Bandit techniques in correction policies
- Splitting inference across the cloud and the client to reduce latency and bandwidth requirements
- Secure model evaluation on the client (model DRM)
Coarsening + Anytime Predictions

\[ f_i(x; \theta) \approx f_i(z; \theta) \]

Better

\[ \mathbb{E}[f_i(x; \theta)] \]