Prediction Serving
what happens after learning?

Joseph E. Gonzalez
Asst. Professor, UC Berkeley
jegonzal@cs.berkeley.edu
Co-founder, GraphLab (now Turi Inc.)
joseph@dato.com
Prediction Serving

Graph Systems

Learning Systems

GraphX

Graph Frames

Time Series

Cluster Management

Frequency Domain Analytics Systems

Multi Task Learning for Job Scheduling

Cross-Cloud Perf. Estimation
Outline

Active Collaborators

Daniel Crankshaw
Xin Wang
Michael Franklin
Ion Stoica
Big Data

Learning

Big Model

Timescale: minutes to days

Systems: offline and batch optimized

Heavily studied ... major focus of the AMPLab
Big Data

Learning

Trainin

Big Model

Inference

Query

Decision

Application

Timescale: ~10 milliseconds

Systems: online and latency optimized

Less studied …
Learning

Big Data → Training → Big Model → Query → Decision → Application

Feedback
Big Data

Learning

Training

Timescale: hours to weeks
Systems: combination of systems
Less studied …

Inference

Decision

Application

Feedback
Learning

Adaptive (~1 seconds)

Inference

Responsive (~10ms)
Key Insight:

Decompose models into fast and slow changing components
Big Data Training Application
Learning

Inference

Feedback

Query
Decision
Application
Hybrid Offline + Online Learning

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

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

Update the user weights \textit{online}:
- Simple to train + more robust model
- Address rapidly changing user statistics
Common modeling structure

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

Matrix Factorization

Deep Learning

Ensemble Methods
Velox Online Learning for Recommendations (Simulated News Rec.)

Partial Updates: 0.4 ms
Retraining: 7.1 seconds

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

Learning

Inference

Fast Changing Model per user

Slow Changing Model

Query

Decision

Application

Feedback

Fast Feedback

Slow Feedback
Big Data Training Application

Learning

Big Data

Training

Velox

Fast Changing Model per user

Fast

Feedback

Inference

Query

Decision

Application

Feedback

Slow

Fast

Slow Changing Model
VELOX: the Missing Piece of BDAS

Learning

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

Berkeley Data Analytics Stack
 VELOX: the Missing Piece of BDAS

Learning
- Spark Streaming
- BlinkDB
- Graph Frames
- Spark SQL
- GraphX
- Keysten ML
- MLLib

Management and Serving
- Velox

Spark

Mesos

Tachyon

HDFS, S3, ...

Berkeley Data Analytics Stack
**VELOX**: the Missing Piece of BDAS

**Learning**
- Spark Streaming
- BlinkDB
- Spark SQL
- Graph Frames
- GraphX

**Management and Serving**
- Keystone ML
- MLLib
- Velox

**Stack**
- Mesos
- Tachyon
- HDFS, S3, ...

**BDAS**
- Berkeley
- Data Analytics

**amplab**
**VELOX Architecture**

**Fraud Detection**

**Content Rec.**

---

Keystone ML

MLLib

Spark

---

**Velox**

Single JVM Instance
**VELOX Architecture**

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

**Velox**

- Keystone ML
- MLLib
- Spark
- Single JVM Instance

**Tools and Technologies**

- Netflix
- Caffe
- Dato
- TensorFlow
- Theano
**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
- mXnet
- VW
- KALDI

---
Clipper Generalizes Velox Across ML Frameworks

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

Clipper

- theano
- Dato
- Create
- Caffe
- TensorFlow
- scikit learn
- VW
- KALDI
- Keystone ML
Clipper

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

mxnet

VW

KeystoneML

KALDI
Clipper Architecture

Applications

Predict

RPC/REST Interface

Observe

Clipper

Theano
Dato
Create
Caffe
TensorFlow
scikit-learn
dmlearn
mxnet
VW
Keystone
ML
KALDI
Clipper Architecture

Clipper

Model Abstraction Layer

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

Correction Layer

Improve accuracy through *ensembles*, *online learning* and *personalization*.

Applications

Predict

RPC/REST Interface

Observe

RPC

Model Wrapper (MW)

Keystone

ML

Caffe

scikit
Clipper Architecture

Applications

Predict

RPC/REST Interface

Observe

Clipper

Correction Policy

Correction Layer

Approximate Caching

Model Abstraction Layer

Adaptive Batching

RPC

Model Wrapper (MW)

RPC

MW

RPC

MW

RPC

MW

Keystone

ML

Caffe

scikit

learn

...
Provides a unified generic prediction API across **frameworks**

- **Reduce Latency** → Approximate Caching
- **Increase Throughput** → Adaptive Batching
- **Simplify Deployment** → RPC + Model Wrapper
Approximate Caching
Adaptive Batching

Model Abstraction Layer

RPC

Model Wrapper (MW)

KeystoneML

Caffe

TF

scikit learn

...
**Common Interface** → **Simplifies Deployment:**

- Evaluate models using original code & systems
- Models run in separate processes
  - Resource isolation
Common Interface ➔ 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
- Hardware Acceleration
  - Helps amortize system overhead

Optimal batch depends on:

- hardware configuration
- model and framework
- system load

Clipper Solution:

- be as slow as allowed...
- Application specifies latency objective
- Clipper uses TCP-like tuning algorithm to increase latency up to the objective
Throughput (Queries Per Second)

Latency (ms)

Batch Sizes (Queries)

Tensor Flow Conv. Net (GPU)

Optimal Batch Size

Throughput

Latency Deadline

P99 Latency

Avg. Latency

0

100

200

300

0

10000

5000

50

40

30

20

10

0

0

10000

5000

50

40

30

20

10

0
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
  - Bag-of-Words Model
  - Images
  - High Dimensional and continuous valued queries have low cache hit rate.

Clipper Solution: Approximate Caching

apply *locality sensitive hash functions*

![Diagram showing cache hits and misses with locality sensitive hash functions.](attachment:image.png)
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
Big Data

Learning

Inference

Velox

- Slow Changing Model
- Fast Changing User Model

Feedback

Application
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>ResNet</td>
<td>9.02%</td>
<td>4512</td>
</tr>
<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>Ensemble</td>
<td>5.86%</td>
<td>2930</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>Inception v3</td>
<td>6.18%</td>
<td>3088</td>
</tr>
</tbody>
</table>
Cost of Ensembles

Increased Load

\( \text{Solutions:} \)
- Caching and Batching
- Load-shedding correction policy can prioritize frameworks

Stragglers

- e.g., framework fails to meet SLO

\( \text{Solution: Anytime} \) predictions
- Correction policy must render predictions with missing inputs
- e.g., built-in correction policies substitute expected value
Anytime Predictions

Slow Changing Model

Fast Changing User Model

Clipper

20ms

Application
Anytime Predictions

\[ w_1^{\text{Gogh}} f_{\text{scikit}}(x) + w_2^{\text{Gogh}} \mathbb{E}_X [f_{\text{TF}}(X)] + w_3^{\text{Gogh}} f_{\text{Caffe}}(x) \]
Evaluation of Throughput Under Heavy Load

Takeaway: Clipper is able to **gracefully degrade accuracy** to maintain availability under heavy load.
Coarsening + Anytime Predictions

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

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

Clipper sits between applications and ML frameworks to

- to simplifying 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
  - Collaboration with MSR

- Splitting inference across the cloud and the client to reduce latency and bandwidth requirements

- Secure model evaluation on the client (model DRM)