Clipper
A Low-Latency Online Prediction Serving System

Daniel Crankshaw
Xin Wang, Giulio Zhou, Michael Franklin, Joseph Gonzalez, Ion Stoica

NSDI 2017
March 29, 2017
Learning

TensorFlow: A system for large-scale machine learning

Project Adam: Building an Efficient and Scalable Deep Learning Training System

Caffe: Convolutional Architecture for Fast Feature Embedding

GraphLab: A New Framework for Parallel Machine Learning

GraphX: Graph Processing in a Distributed Dataflow Framework

Parameter Server for Distributed Machine Learning
Big Data

Training

Complex Model

Learning
Learning Produces a Trained Model

Query

Model

"CAT"

Decision
Big Data -> Training -> Model -> Serving

Learning

Serving

Big Data

Training

Model

Query

Decision

Application
Big Data Training

Learning

Model

Predictions - Serving for interactive applications
Timescale: ~10s of milliseconds
Prediction-Serving Raises New Challenges
Prediction-Serving Challenges

Support low-latency, high-throughput serving workloads

Large and growing ecosystem of ML models and frameworks
Support low-latency, high-throughput serving workloads

Models getting more complex
- 10s of GFLOPs [1]

Deployed on critical path
- Maintain SLOs under heavy load

Google Translate

Serving

82,000 GPUs running 24/7

140 billion words a day\(^1\)


---

Invented New Hardware!

Tensor Processing Unit (TPU)
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
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

Fraud Detection

Content Rec.

Personal Asst.

Robotic Control

Machine Translation

Netflix

Caffe

TensorFlow

mxnet

VW

Kaldi
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

Big Data → Training → Model → Batch Analytics
Use existing systems: Offline Scoring

Batch Analytics

Big Data

Training

Model

Scoring

Datastore

Use existing systems:
Offline Scoring
Use existing systems: Offline Scoring

Look up decision in datastore

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
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

Caffe

RPC

MC

RPC

MC

RPC

MC

RPC

MC

...
Clipper Architecture

Applications

Predict ➤ RPC/REST Interface ➤ Observe

Clipper

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.

Model Abstraction Layer
Clipper Implementation

Applications

Predict

RPC/REST Interface

Observe

Clipper

Core system: 5000 lines of Rust

RPC:

- 100 lines of Python
- 250 lines of Rust
- 200 lines of C++
<table>
<thead>
<tr>
<th>Model Container (MC)</th>
<th>Caffe</th>
<th>TensorFlow</th>
<th>scikit-learn</th>
</tr>
</thead>
</table>

**Model Abstraction Layer**

- **Caching**
- **Adaptive Batching**

RPC

Provide a common interface to models while...

...
Common Interface → Simplifies Deployment:
- Evaluate models using original code & systems
Container-based Model Deployment

**Implement Model API:**

class ModelContainer:
    def __init__(model_data)
    def predict_batch(inputs)
Container-based Model Deployment

Implement Model API:

```python
class ModelContainer:
    def __init__(model_data)
    def predict_batch(inputs)
```

- Implemented in many languages
  - Python
  - Java
  - C/C++
Model implementation packaged in container

class ModelContainer:
    def __init__(self, model_data):
    def predict_batch(inputs)
Container-based Model Deployment

Clipper

Model Container (MC)

RPC

Caffe

RPC

MC

RPC

MC

RPC

MC
Common Interface → Simplifies Deployment:

- Evaluate models using original code & systems
- Models run in separate processes as Docker containers
  - Resource isolation
Common Interface → Simplifies Deployment:

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

**Problem:** frameworks optimized for **batch processing** not latency
**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

- **Hardware Acceleration**

- **GRPC**

  Helps amortize system overhead
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
Throughput (Queries Per Second)

Tensor Flow Conv. Net (GPU)
Tensor Flow Conv. Net (GPU)

Throughput (Queries Per Second)

Latency (ms)

Better

Better

Batch Size

Deadline
Tensor Flow Conv. Net (GPU)

Throughput (Queries Per Second)

Latency (ms)

Optimal Batch Size

Better

Deadline

P99

Better
### Throughput (QPS)

<table>
<thead>
<tr>
<th>Model</th>
<th>Throughput (QPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No-Op</td>
<td>8963</td>
</tr>
<tr>
<td>Random Forest (SKlearn)</td>
<td>317</td>
</tr>
<tr>
<td>Linear SVM (PySpark)</td>
<td>7206</td>
</tr>
<tr>
<td>Linear SVM (SKLearn)</td>
<td>1920</td>
</tr>
<tr>
<td>Kernel SVM (SKLearn)</td>
<td>203</td>
</tr>
<tr>
<td>Log Regression (SKLearn)</td>
<td>1921</td>
</tr>
</tbody>
</table>

**Better**

- **No Batching**
Throughput (QPS)

Better

<table>
<thead>
<tr>
<th>Method</th>
<th>Adaptive</th>
<th>No Batching</th>
</tr>
</thead>
<tbody>
<tr>
<td>No-Op</td>
<td>48386</td>
<td>8963</td>
</tr>
<tr>
<td>Random Forest (SKLearn)</td>
<td>22859</td>
<td>317</td>
</tr>
<tr>
<td>Linear SVM (PySpark)</td>
<td>29350</td>
<td>7206</td>
</tr>
<tr>
<td>Linear SVM (SKLearn)</td>
<td>48934</td>
<td>1920</td>
</tr>
<tr>
<td>Kernel SVM (SKLearn)</td>
<td>197197</td>
<td>203197</td>
</tr>
<tr>
<td>Log Regression (SKLearn)</td>
<td>47219</td>
<td>1921</td>
</tr>
</tbody>
</table>
### Throughput (QPS)

- **Better**

### P99 Latency (ms)

- **20 ms is Fast Enough**
- **Better**

#### Throughput (QPS)

- **Adaptive**
- **No Batching**

#### P99 Latency (ms)

- **No-Op**
- **Random Forest (SKlearn)**
- **Linear SVM (PySpark)**
- **Linear SVM (SKlearn)**
- **Kernel SVM (SKlearn)**
- **Log Regression (SKLearn)**

**20 ms is Fast Enough**
Overhead of decoupled architecture

Applications

Predict

RPC/REST Interface

Feedback

Clipper

RPC

MC

Spark

Caffe

MC

learn
Overhead of decoupled architecture
Overhead of decoupled architecture
Overhead of decoupled architecture

Model: AlexNet trained on CIFAR-10

Throughput

<table>
<thead>
<tr>
<th></th>
<th>TensorFlow Serving</th>
<th>Clipper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput (QPS)</td>
<td>5519</td>
<td>5472</td>
</tr>
</tbody>
</table>

P99 Latency (ms)

<table>
<thead>
<tr>
<th></th>
<th>TensorFlow Serving</th>
<th>Clipper</th>
</tr>
</thead>
<tbody>
<tr>
<td>P99 Latency</td>
<td>47.04</td>
<td>46.75</td>
</tr>
</tbody>
</table>

Better
Clipper Architecture

Applications

Predict $\uparrow$

RPC/REST Interface

Clipper

Observe $\uparrow$

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

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

Model Selection Layer

Model Abstraction Layer

RPC

Model Container (MC)

RPC

MC

RPC

MC

RPC

MC

RPC

MC
Clipper

Model Selection Layer

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

Periodic retraining

Experiment with new models and frameworks

Version 1

Version 2

Version 3
Selection Policy: Estimate confidence

Policy

Version 2

Version 3

"CAT"
"CAT"
"CAT"
"CAT"

“CAT” CONFIDENT
Selection Policy: Estimate confidence
Selection Policy: Estimate confidence

<table>
<thead>
<tr>
<th>Top-5 Error Rate</th>
<th>ImageNet</th>
<th>ensemble</th>
<th>4-agree</th>
<th>5-agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.0586</td>
<td>0.0469</td>
<td>0.0327</td>
</tr>
</tbody>
</table>

Better
Selection Policy: Estimate confidence

Better width is percentage of query workloads

ImageNet

- ensemble: 0.0586
- 4-agree: 0.0469
- 5-agree: 0.0327
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*
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
GPU Cluster Scaling

![Graph showing throughput and latency for different numbers of replicas.](image-url)