

# 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

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Zhifeng Zhang, Bo Chen, Ming Tang, Paul Vasserman, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zhang

Google Brain

### Abstract

TensorFlow is a machine learning system that operates at scale on many datasets, and moving to a distributed system. TensorFlow is based on a first-generation system,

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

Trishul Chilimbi   Yutaka Suzue   Johnson Apacible   Karthik Kalyanaraman  
Microsoft Research

### ABSTRACT

Large deep neural networks have demonstrated state-of-the-art performance on a variety of recognition tasks. However, training such models is extremely time consuming, due to the amount of compute required. The implementation of a distributed training system must be comprised of commodity hardware that exhibits high task accuracy and low latency. This paper presents a system that achieves high efficiency by using a system co-design approach to workload computation and communication. Asynchrony through performance and shared memory of trained models is used to achieve more efficient and faster training. Our results show that our system is more efficient and faster than previously held the record for training large deep neural networks.

### ABSTRACT

Caffe provides multimedia scientists and practitioners with a clean and modifiable framework for state-of-the-art deep learning algorithms and a collection of reference models. The framework is a BSD-licensed C++ library with Python and MATLAB bindings for training and deploying general-purpose convolutional neural networks and other deep models efficiently on commodity architectures. Caffe fits industry and internet-scale media needs by CUDA GPU computation, processing over 40 million images a day on a single K40 or Titan GPU ( $\approx 2.5$  ms per image). By separating model representation from actual implementation, Caffe allows experimentation and seamless switching among platforms for ease of development and deployment from prototyping machines to cloud environments.

Caffe is maintained and developed by the Berkeley Vision and Learning Center (BVLC) with the help of an active community of contributors on GitHub. It powers ongoing research projects, large-scale industrial applications, and

## Caffe: Convolutional Architecture for Fast Feature Embedding\*

Yangqing Jia<sup>1</sup>, Evan Shelhamer<sup>1</sup>, Jeff Donahue, Sergey Karayev,  
Jonathan Long, Ross Girshick, Sergio Guadarrama, Trevor Darrell  
SUBMITTED to ACM MULTIMEDIA 2014 OPEN SOURCE SOFTWARE COMPETITION  
UC Berkeley EECS, Berkeley, CA 94702  
 [{jiayq,shelhamer,jdonahue,sergeyk,jonlong,rbg,squada,trevor}@eecs.berkeley.edu](mailto:{jiayq,shelhamer,jdonahue,sergeyk,jonlong,rbg,squada,trevor}@eecs.berkeley.edu)

### 1. INTRODUCTION

A key problem in multimedia data analysis is discovery of effective representations for sensory inputs—images, sound-waves, haptics, etc. While performance of conventional, handcrafted features has plateaued in recent years, new developments in deep compositional architectures have kept performance levels rising [8]. Deep models have outperformed hand-engineered feature representations in many domains, and made learning possible in domains where engineered features were lacking entirely.

We are particularly motivated by large-scale visual recognition, where a specific type of deep architecture has achieved a commanding lead on the state-of-the-art. These *Convolutional Neural Networks*, or CNNs, are discriminatively trained via back-propagation through layers of convolutional filters and other operations such as rectification and pooling. Following the early success of digit classification in the 90's, these models have recently surpassed all known methods for large-scale visual recognition, and have been adopted by in-

## Spark: Cluster Computing with Working Sets

Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica  
University of California, Berkeley

MapReduce/Dryad job, each job must reload the data from disk, incurring a significant performance penalty. • **Interactive analytics:** Hadoop is often used to support ad-hoc exploratory queries on large datasets, through SQL interfaces such as Pig [21] and Hive [1]. Ideally, a user would be able to load a dataset of interest into memory across a number of machines and query it repeatedly. However, with Hadoop, each query incurs significant latency (tens of seconds) because it runs a separate MapReduce job and reads data from disk.

This paper presents a new cluster computing framework

which supports applications providing similar scalability to MapReduce.

The key idea behind Spark is that of a *working set*, which represents the data that is loaded across a set of machines. When a partition is lost, the RDD has to be recomputed across machines using a *parallel operation*. Through a notion of *working sets*, the RDD has to be derived from other partitions. Although memory abstraction limits expressivity on the one hand, it is useful for a variety of applications. In Scala [5], a functional programming language for the distributed data abstraction, Spark can be used to define RDDs, map them in parallel, and reduce them. Spark is the first distributed system to support programming abstractions for processing large datasets. The presentation of Spark in this paper is based on the system as it exists today, which is expected to support Hadoop workloads and can handle a dataset with sub-second latency as follows. Sec-

## GraphLab: A New Framework For Parallel Machine Learning

Yucheng Low  
Carnegie Mellon University  
[ylow@cs.cmu.edu](mailto:ylow@cs.cmu.edu)

Joseph Gonzalez  
Carnegie Mellon University  
[jegonzal@cs.cmu.edu](mailto:jegonzal@cs.cmu.edu)

Aapo Kyrola  
Carnegie Mellon University  
[akyrola@cs.cmu.edu](mailto:akyrola@cs.cmu.edu)

Danny Bickson  
Carnegie Mellon University

Carlos Guestrin  
Carnegie Mellon University

Joseph M. Hellerstein  
UC Berkeley

## GraphX: Graph Processing in a Distributed Dataflow Framework

Joseph E. Gonzalez\*, Reynold S. Xin\*<sup>†</sup>, Ankur Dave\*, Daniel Crankshaw\*

Michael J. Franklin\*, Ion Stoica\*<sup>†</sup>

\*UC Berkeley AMPLab      <sup>†</sup>Databricks

### Abstract

PageRank   Connected   K-core   Triangle

## Parameter Server for Distributed Machine Learning

Mu Li<sup>1</sup>, Li Zhou<sup>1</sup>, Zichao Yang<sup>1</sup>, Aaron Li<sup>1</sup>, Fei Xia<sup>1</sup>,  
David G. Andersen<sup>1</sup> and Alexander Smola<sup>1,2</sup>

<sup>1</sup>Carnegie Mellon University

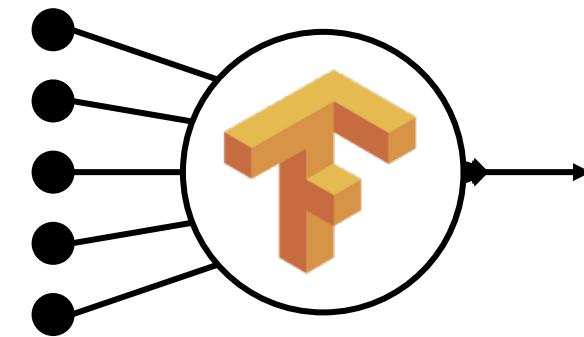
<sup>2</sup>Google Strategic Technologies

[{muli, lizhou, zichao, aaronli, feixia, dga}@cs.cmu.edu](mailto:{muli, lizhou, zichao, aaronli, feixia, dga}@cs.cmu.edu),  [alex@smola.org](mailto:alex@smola.org)

### Abstract

We propose a parameter server framework to solve distributed machine learning problems. Both data and workload are distributed into client nodes, while server nodes maintain globally shared parameters, which are represented as sparse vectors and matrices. The framework manages asynchronous data communications between clients and servers. Flexible consistency models, elastic scalability and fault tolerance are supported by this framework. We present algorithms and theoretical analysis for challenging nonconvex and nonsmooth problems. To demonstrate the scalability of the proposed framework, we show experimental results on real data with billions of parameters.

# Learning



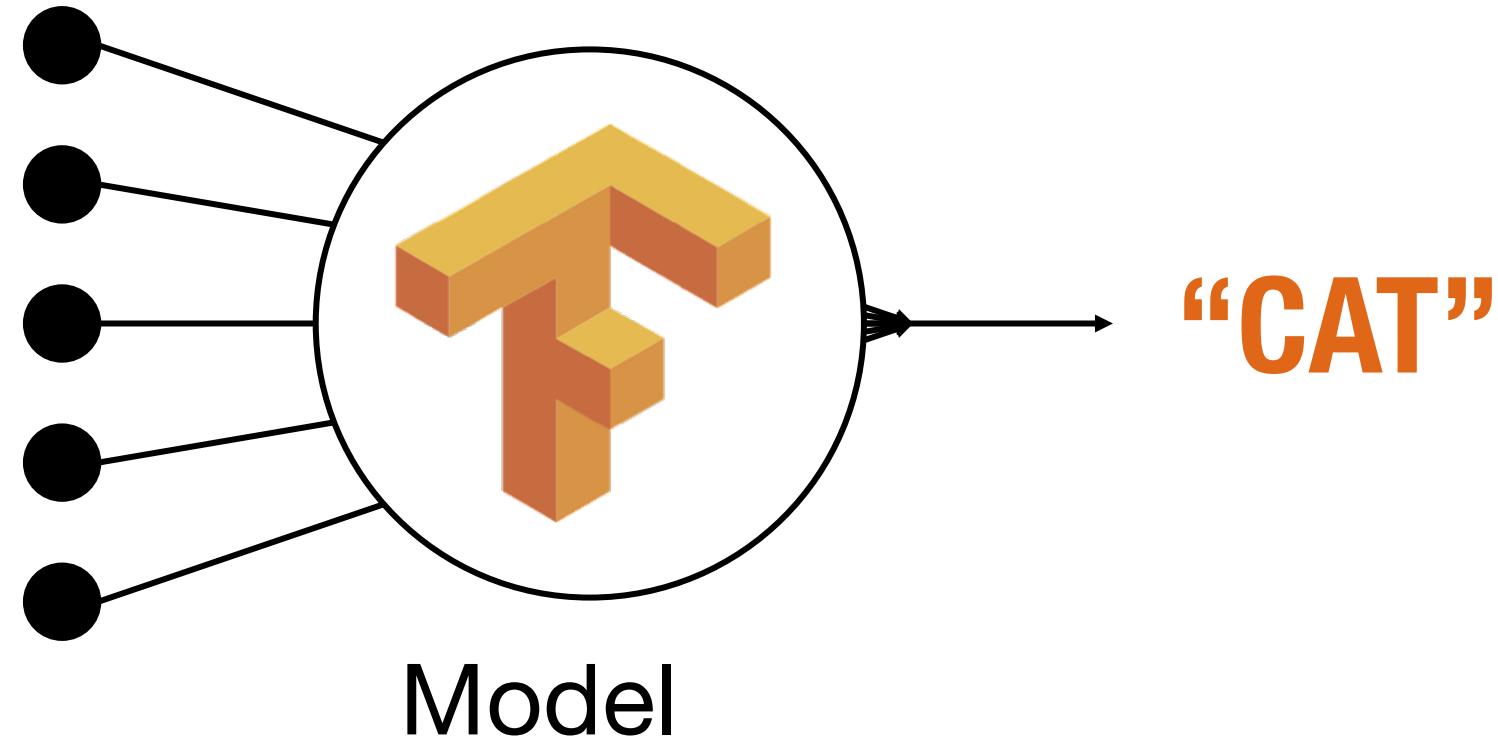
Complex Model

# Learning Produces a Trained Model

*Query*



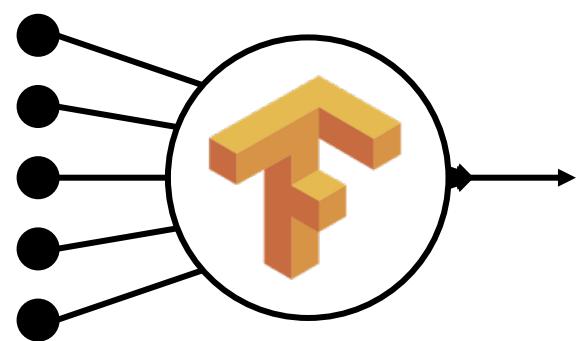
*Decision*



# Learning



Training

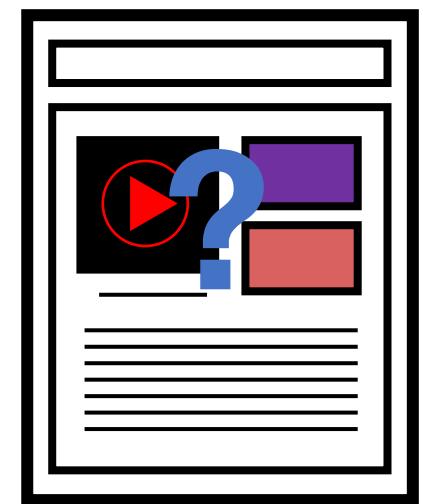


Model

# Serving

Query

Decision

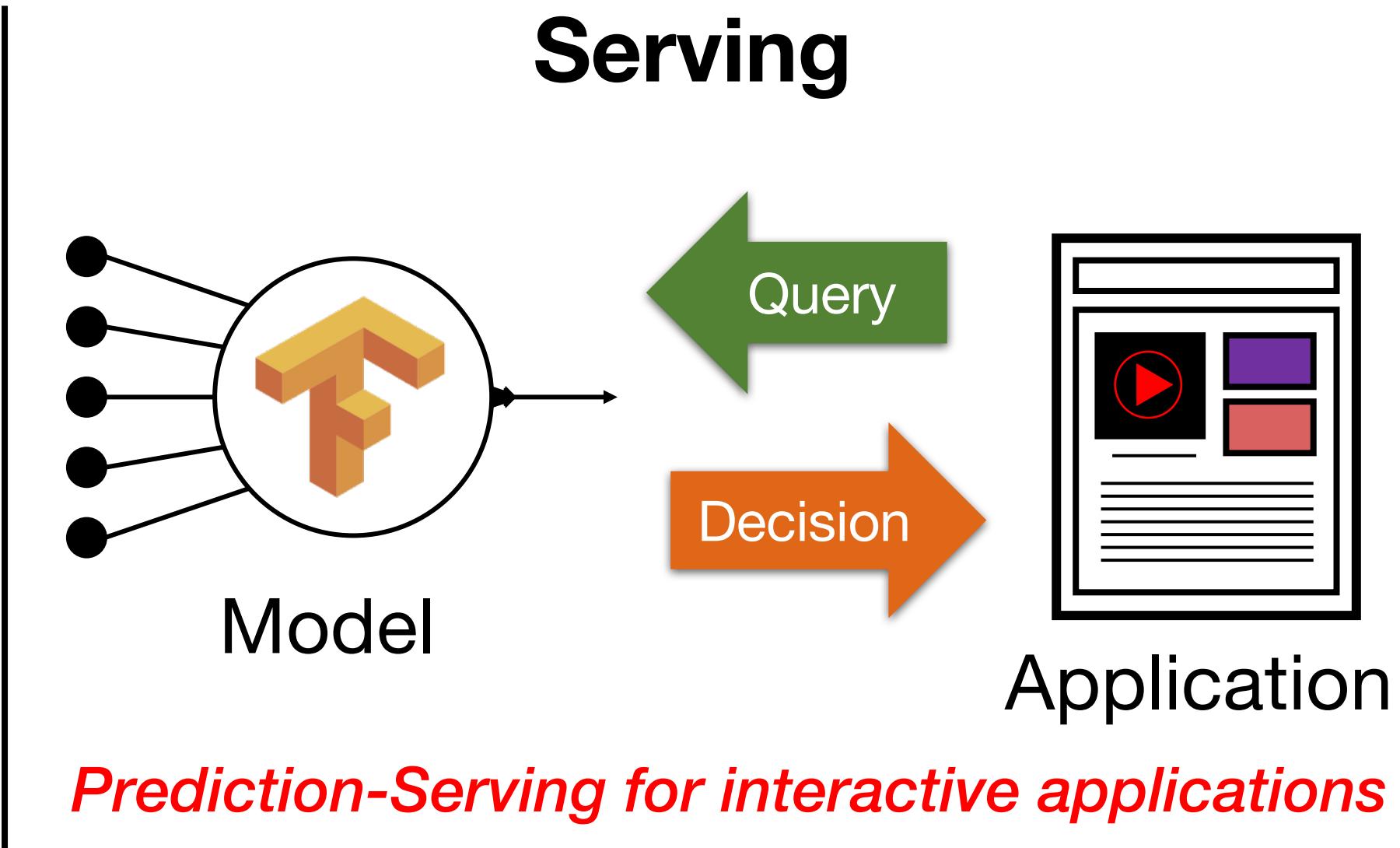


Application

# Learning

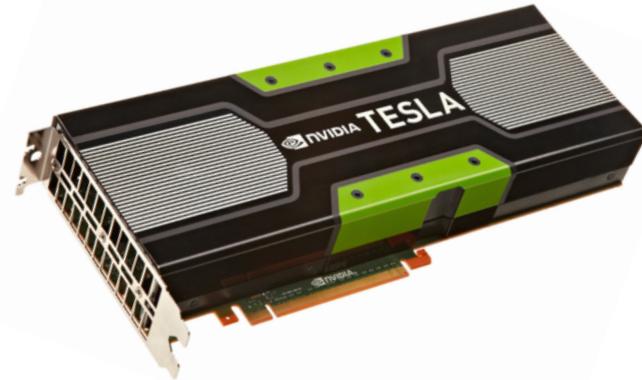
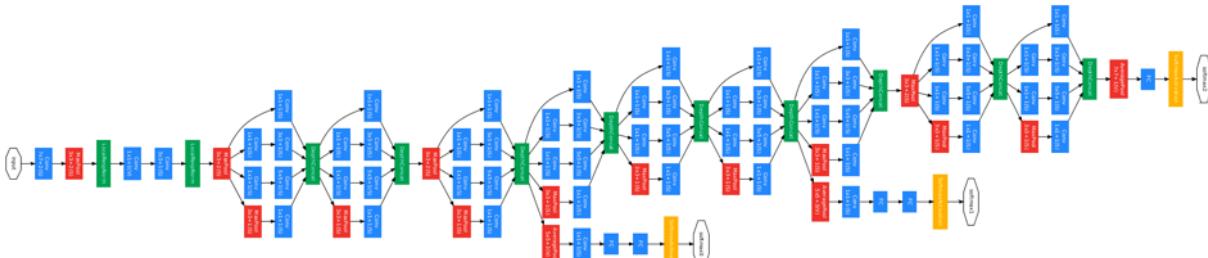


Training



# *Prediction-Serving Raises New Challenges*

# *Prediction-Serving Challenges*

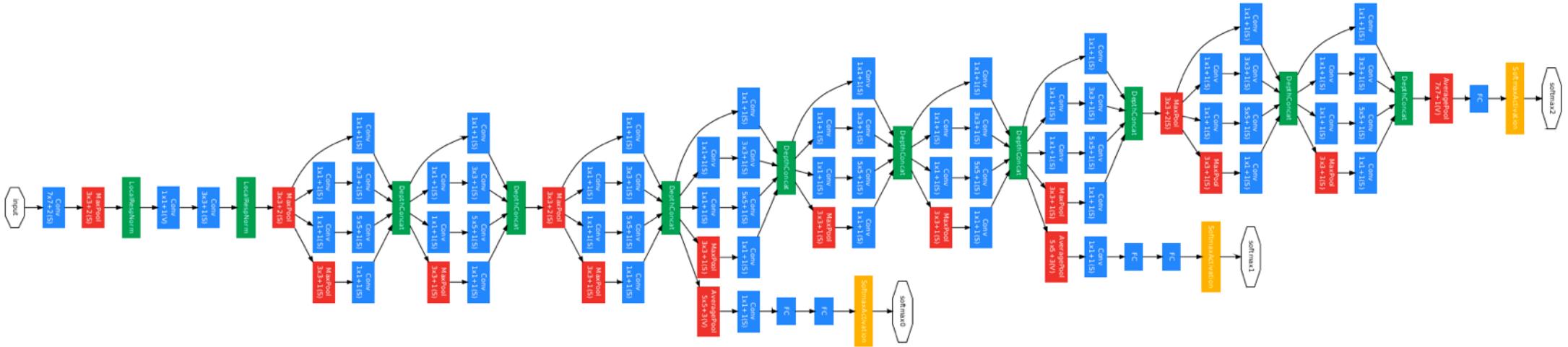


*Support low-latency, high-throughput serving workloads*



*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



## *Using specialized hardware for predictions*

[1] Deep Residual Learning for Image Recognition. He et al. CVPR 2015.

# Google Translate

## Serving



82,000 GPUs  
running 24/7

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi  
`yonghui,schuster,zhifengc,qvl,mnorouzi@google.com`

Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean

***Invented New Hardware!  
Tensor Processing Unit  
(TPU)***

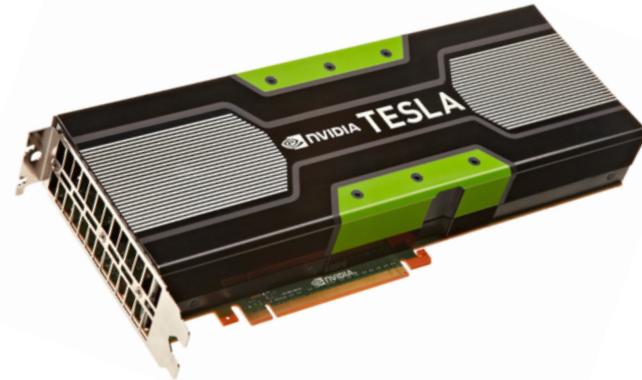
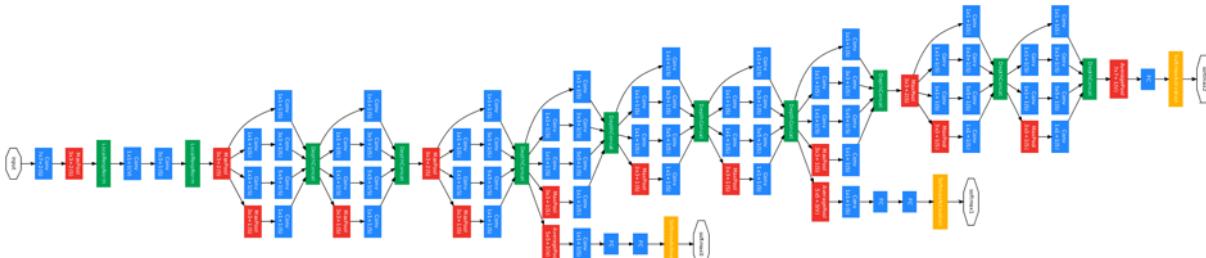
[1] <https://www.nytimes.com/2016/12/14/magazine/the-great-ai-awakening.html>

# Big Companies Build One-Off Systems

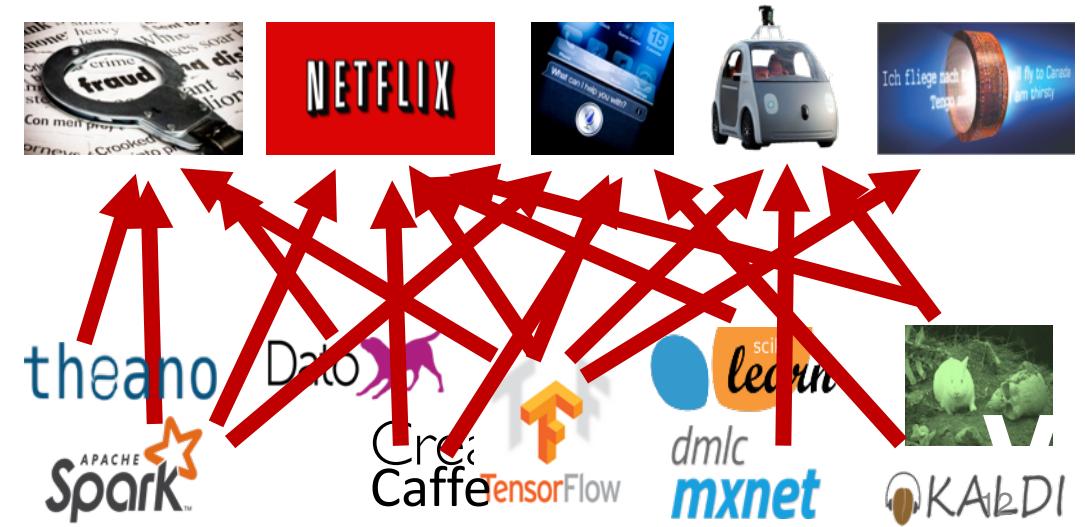
## Problems:

- 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*



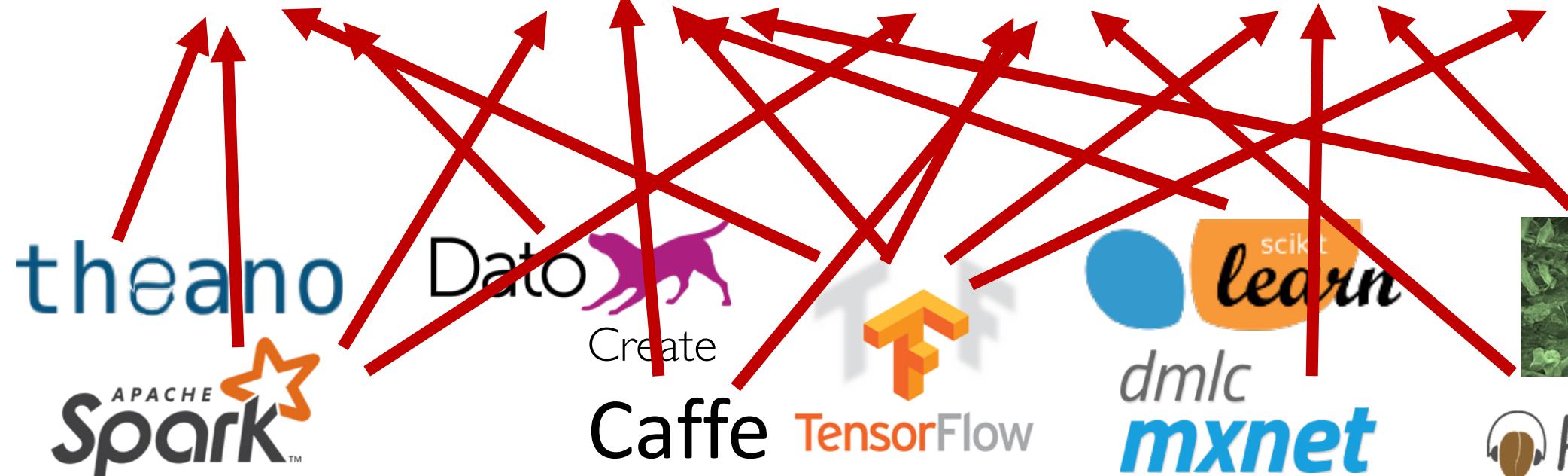
*Support low-latency, high-throughput serving workloads*



*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



*Large and growing ecosystem of ML models and frameworks*

***Difficult to deploy and  
brittle to manage***

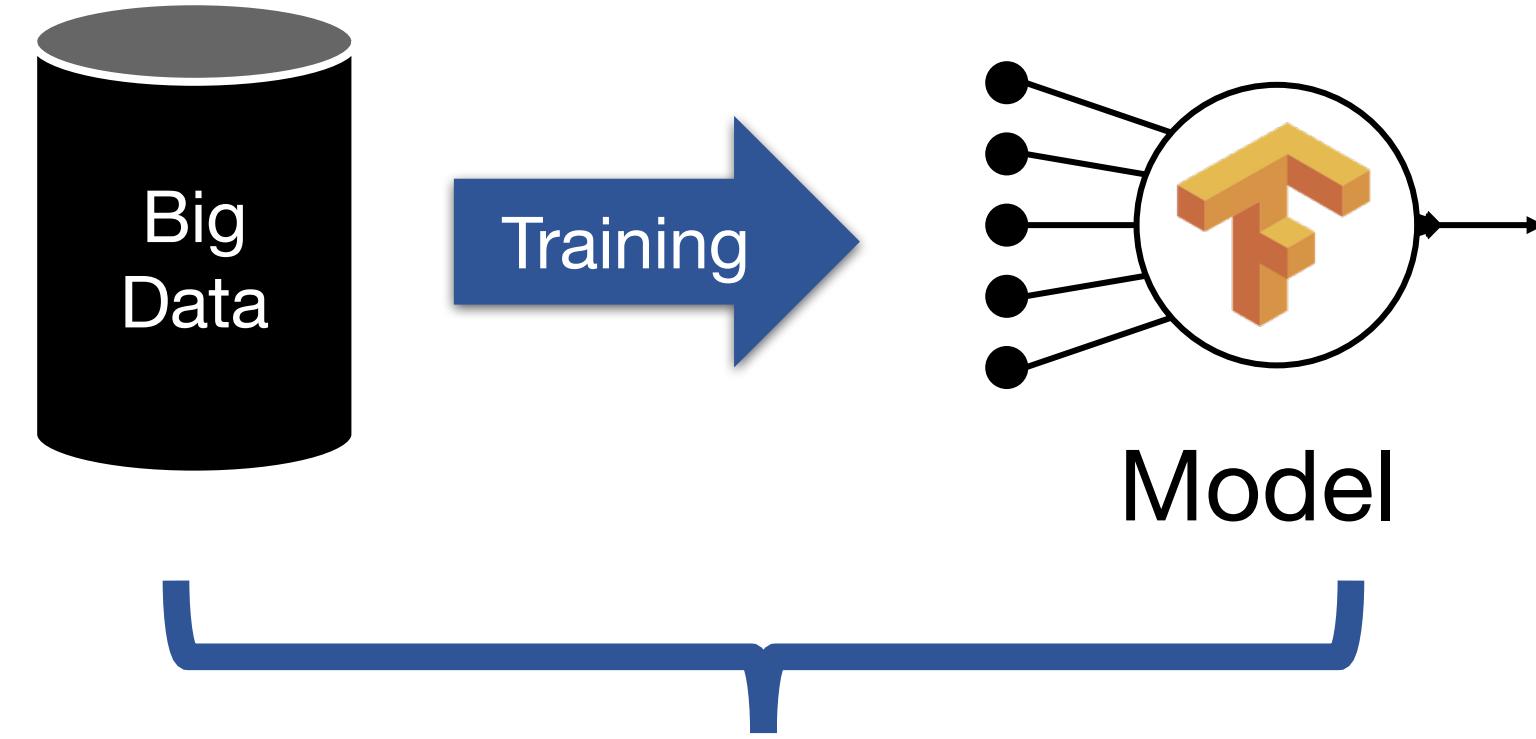


***Varying physical  
resource requirements***



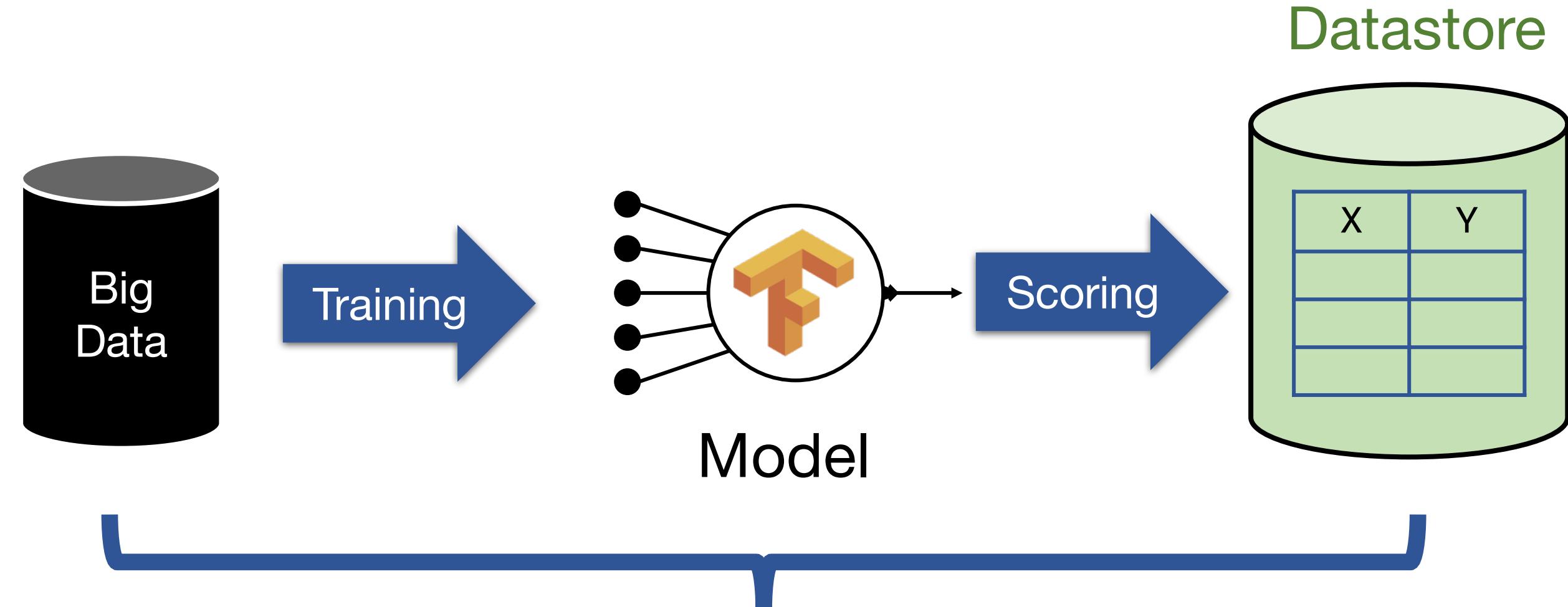
*But most companies  
can't build new  
serving systems... .*

# Use existing systems: Offline Scoring



**Batch Analytics**

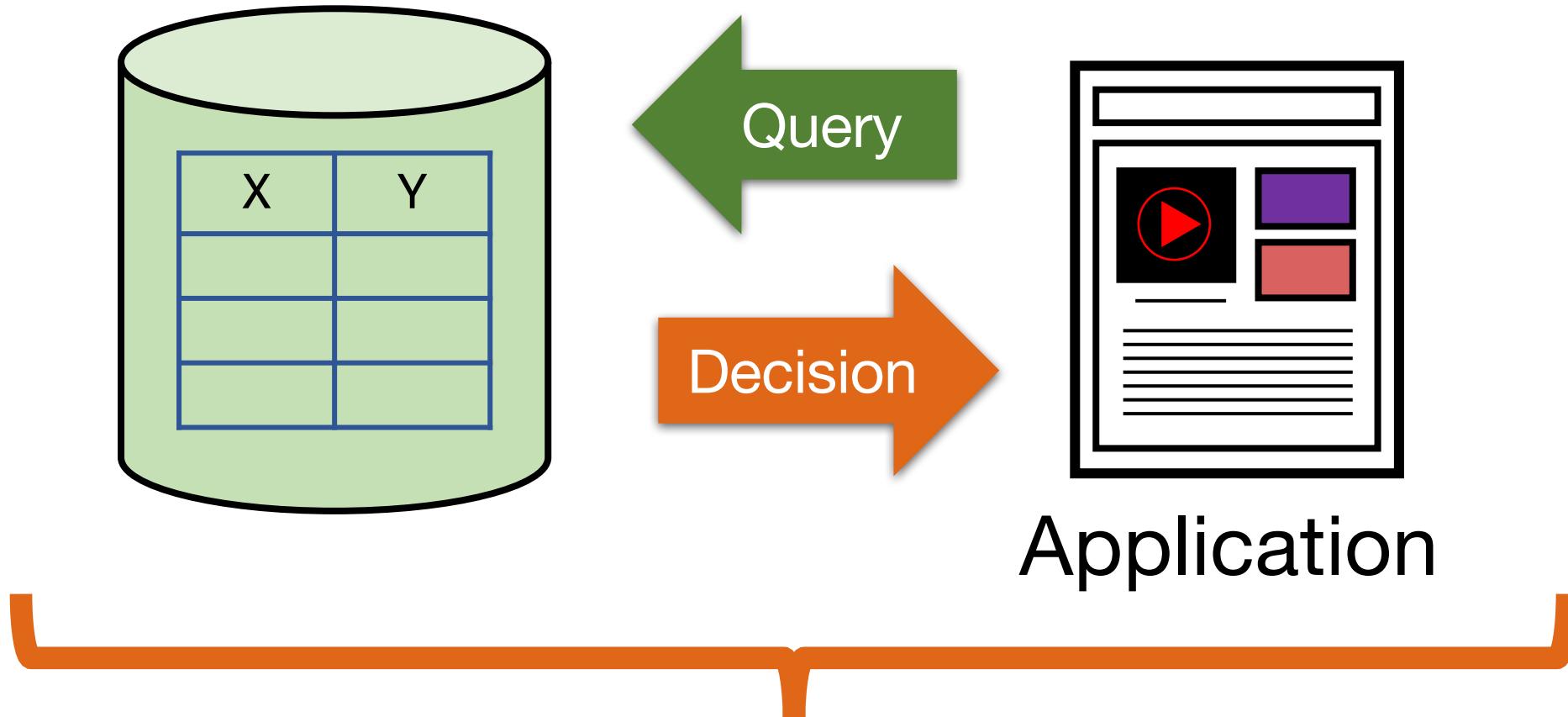
# Use existing systems: Offline Scoring



**Batch Analytics**

# Use existing systems: Offline Scoring

Look up decision in datastore



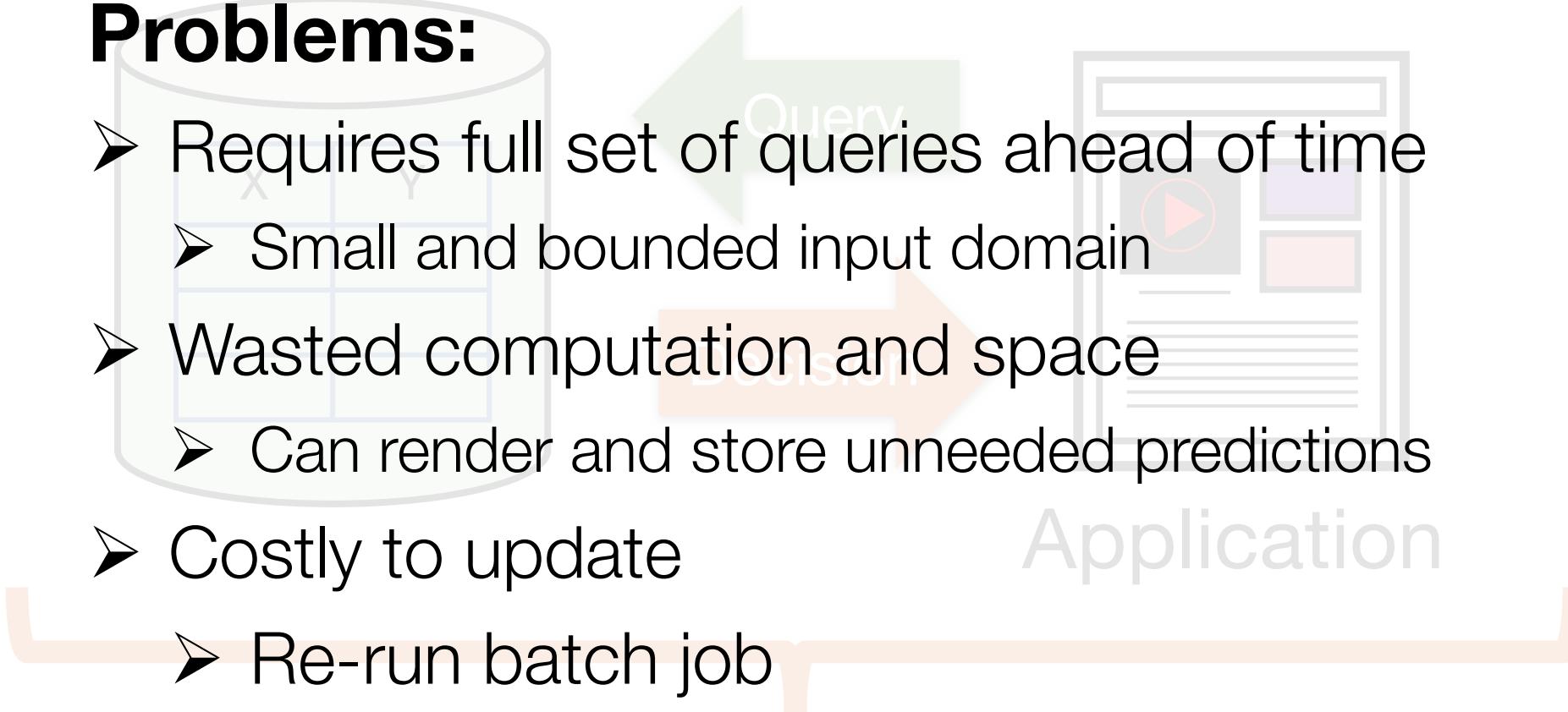
**Low-Latency Serving**

# Use existing systems: Offline Scoring

Look up decision in datastore

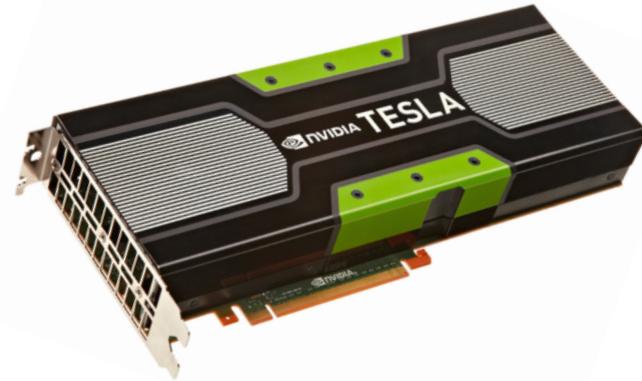
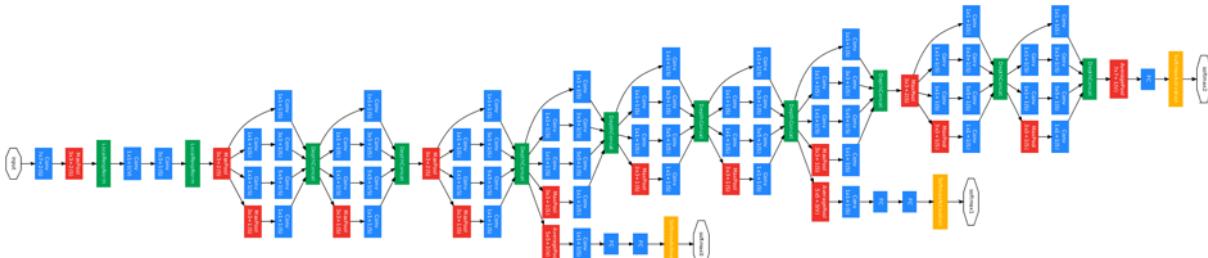
## Problems:

- 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

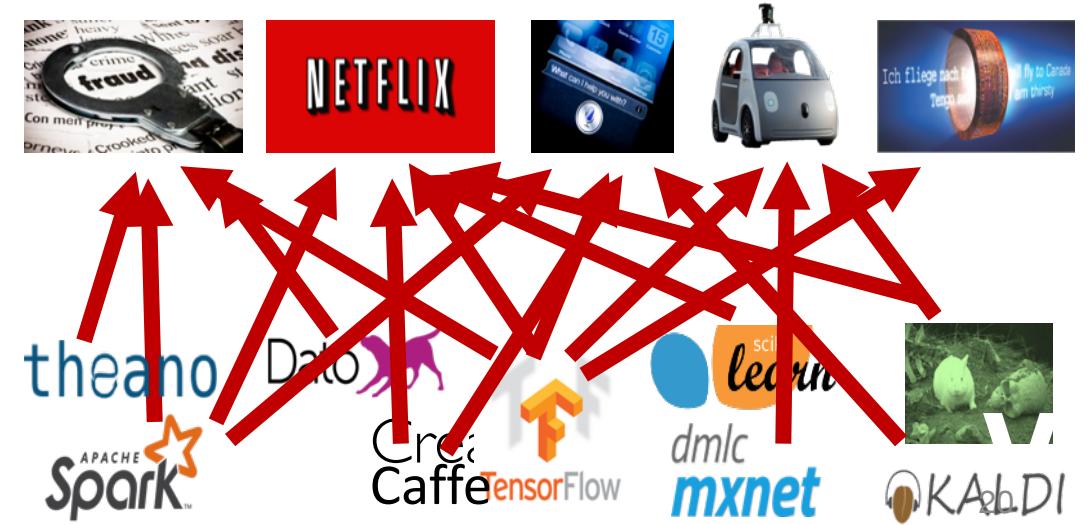


Low-Latency Serving

# *Prediction-Serving Challenges*



*Support low-latency, high-throughput serving workloads*



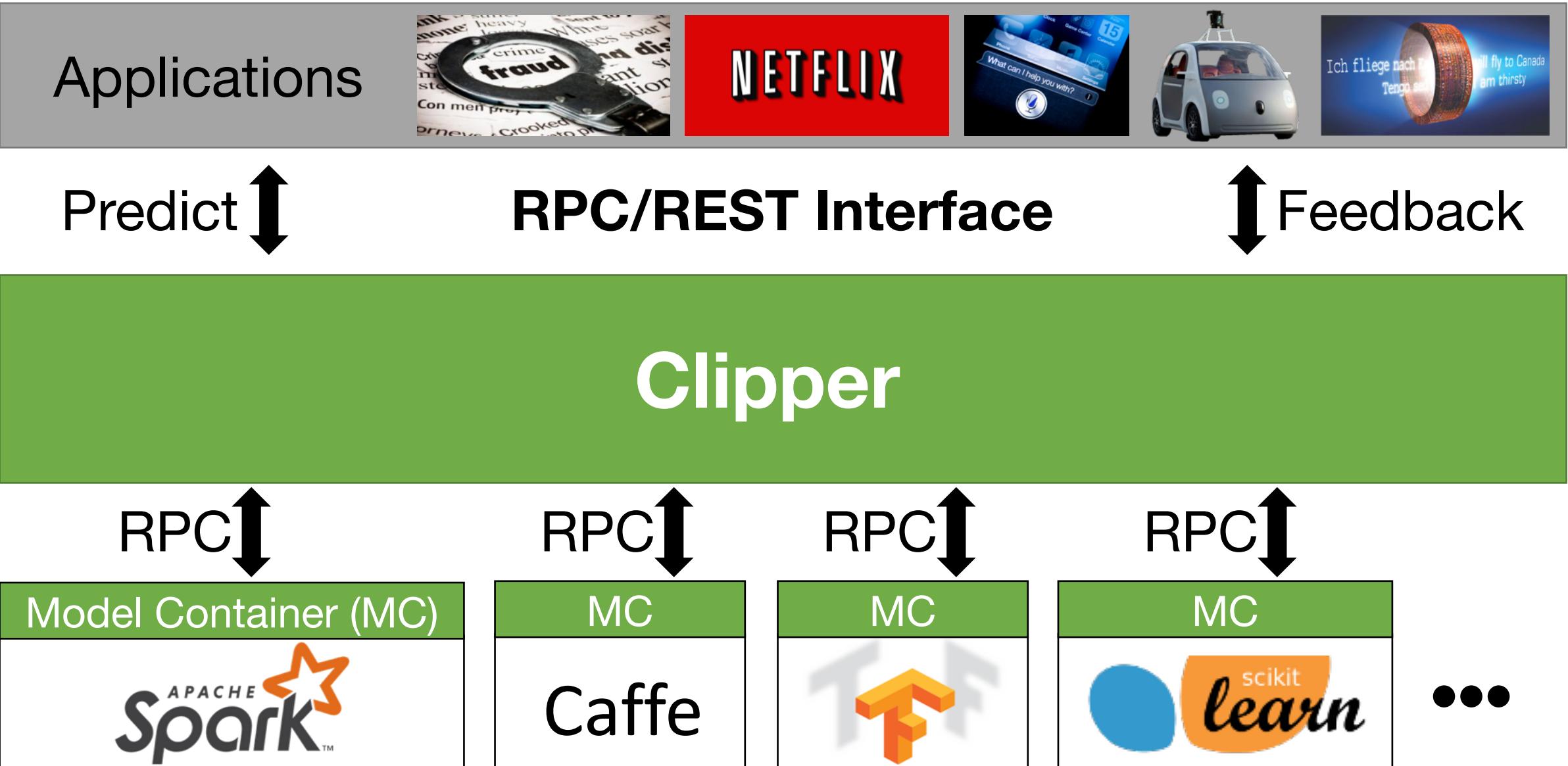
*Large and growing ecosystem  
of ML models and frameworks*

*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



# Clipper Architecture

Applications



NETFLIX



Predict

RPC/REST Interface

Observe

## Clipper

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

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

Model Selection Layer

Model Abstraction Layer

RPC

RPC

RPC

RPC

Model Container (MC)



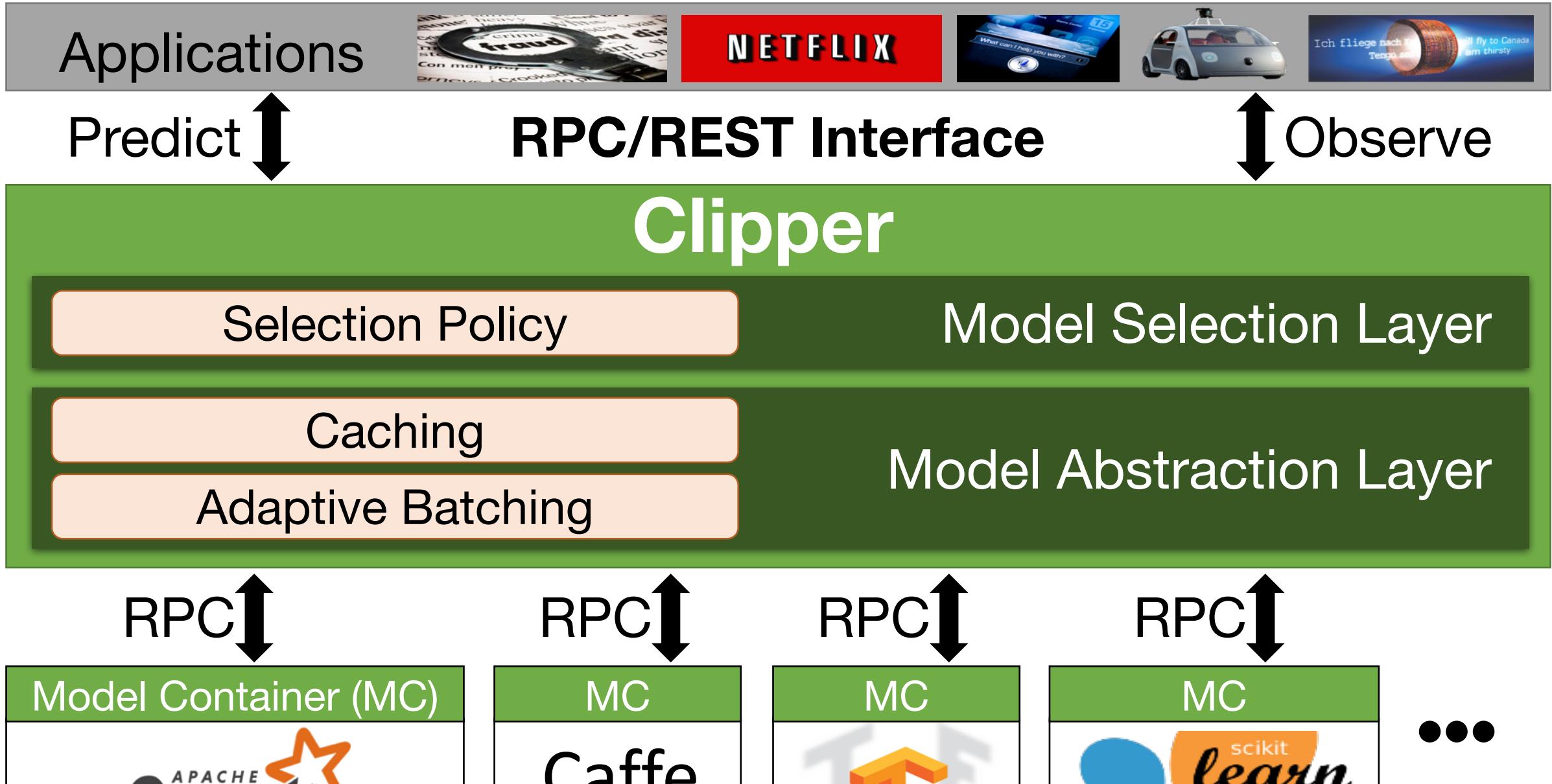
MC  
Caffe

MC  
TensorFlow

MC  
scikit-learn

...

# Clipper Architecture



# Clipper Implementation

Applications



NETFLIX



Predict

RPC/REST Interface

Observe

## Clipper

Core system: 5000 lines of Rust



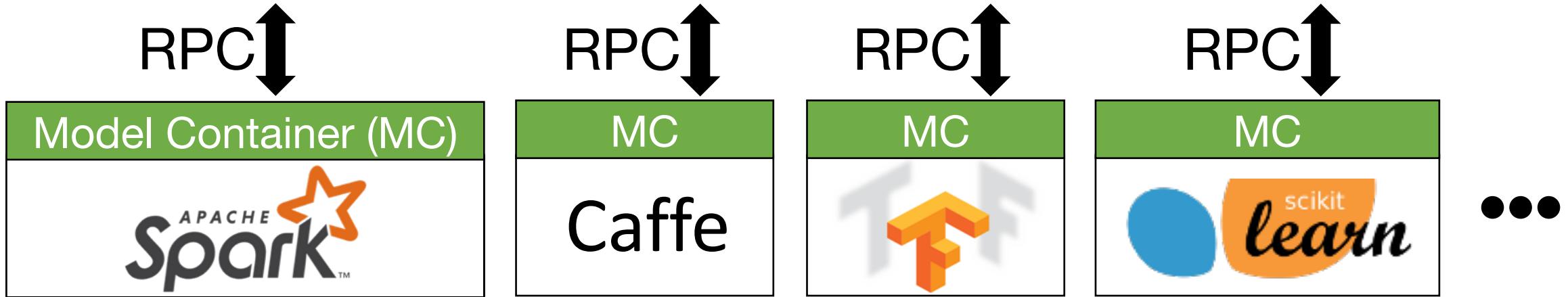
RPC:

- 100 lines of Python
- 250 lines of Rust
- 200 lines of C++

Caching

Adaptive Batching

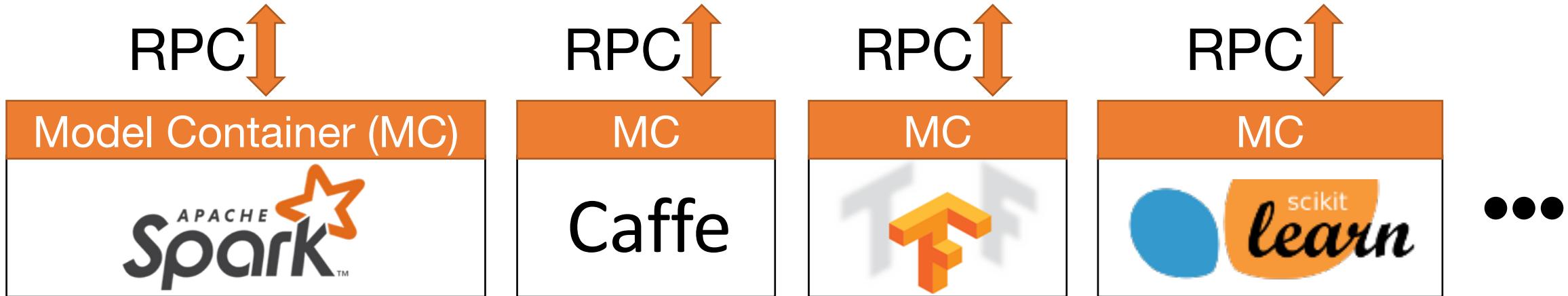
Model Abstraction Layer



Caching

Adaptive Batching

Model Abstraction Layer



**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:*

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

- Implemented in many languages
  - Python
  - Java
  - C/C++

# Container-based Model Deployment

*Model implementation packaged in container*

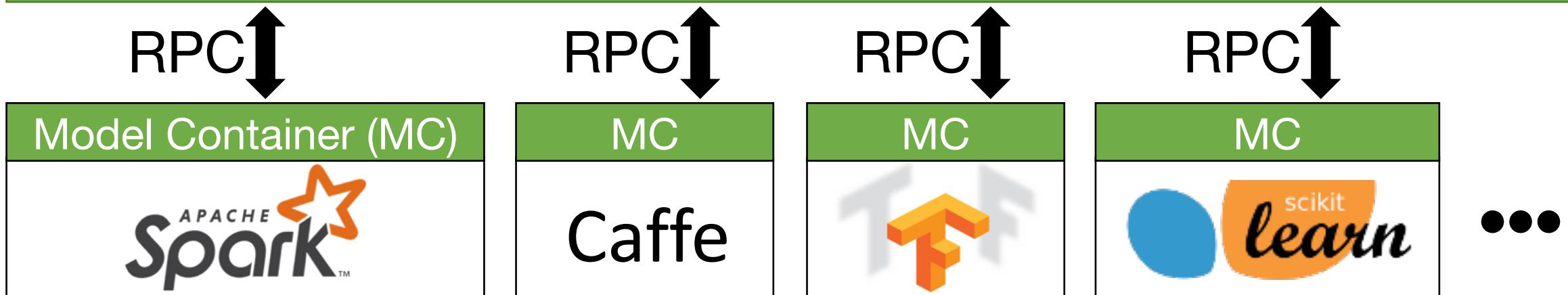
## Model Container (MC)

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



# Container-based Model Deployment

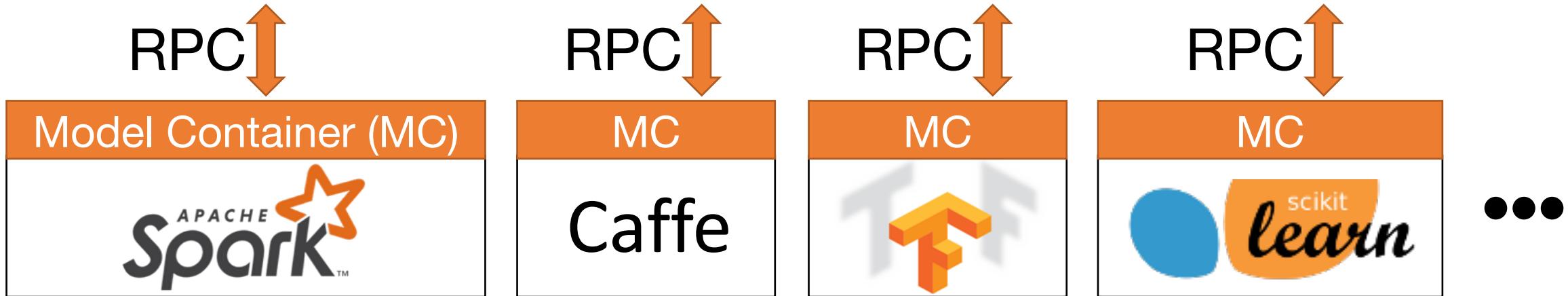
## Clipper



Caching

Adaptive Batching

Model Abstraction Layer



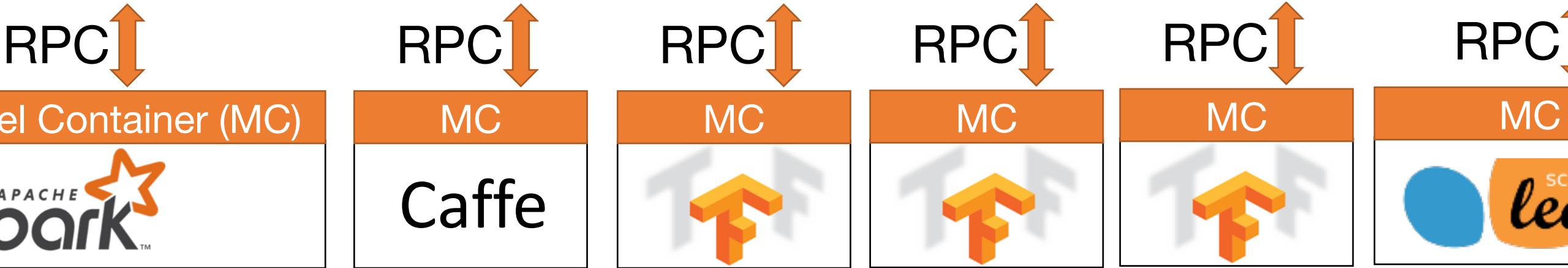
## Common Interface → Simplifies Deployment:

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

Caching

Adaptive Batching

Model Abstraction Layer



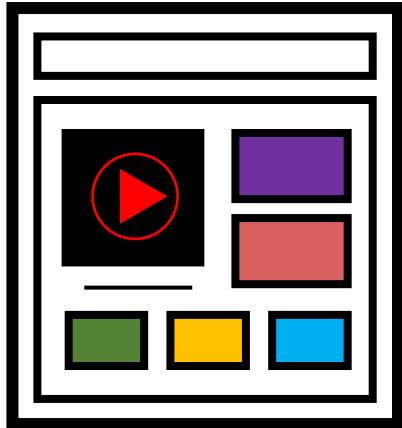
## Common Interface → Simplifies Deployment:

- 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**

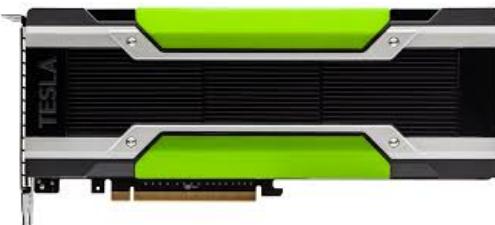
# Batching to Improve Throughput

- Why batching helps:



A single page load may generate many queries

Hardware Acceleration



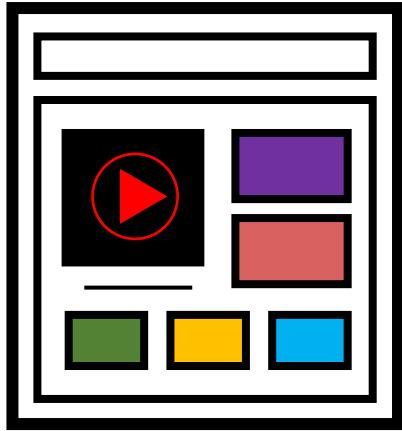
Helps amortize system overhead

- Optimal batch depends on:

- hardware configuration
- model and framework
- system load

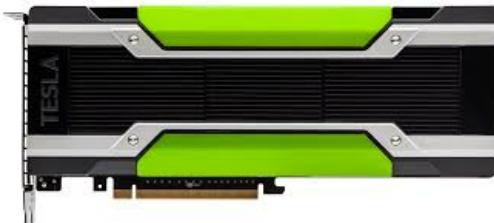
# Adaptive Batching to Improve Throughput

- Why batching helps:



A single page load may generate many queries

Hardware Acceleration



Helps amortize system overhead

- Optimal batch depends on:

- hardware configuration
- model and framework
- system load

## Clipper Solution:

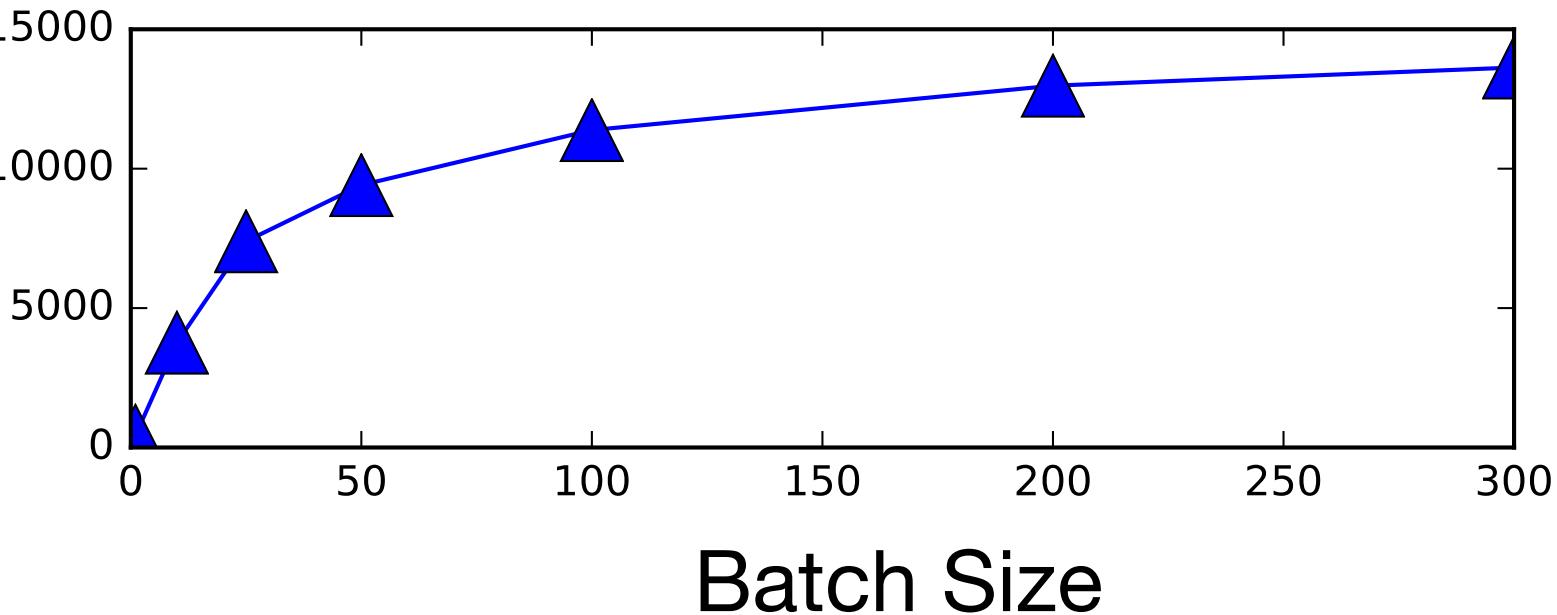
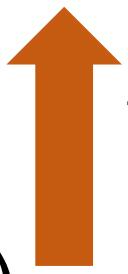
*Adaptively tradeoff latency and throughput...*

- Inc. batch size until the latency objective is exceeded (**Additive Increase**)
- If latency exceeds SLO cut batch size by a fraction (**Multiplicative Decrease**)

# Tensor Flow Conv. Net (GPU)

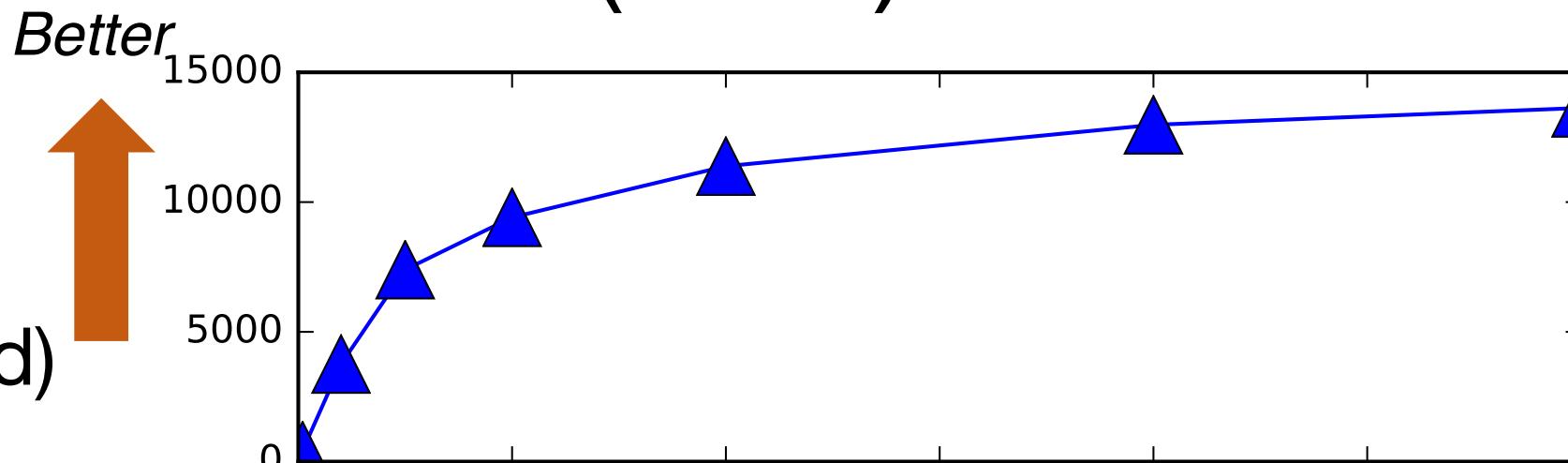
Throughput  
(Queries Per Second)

*Better*

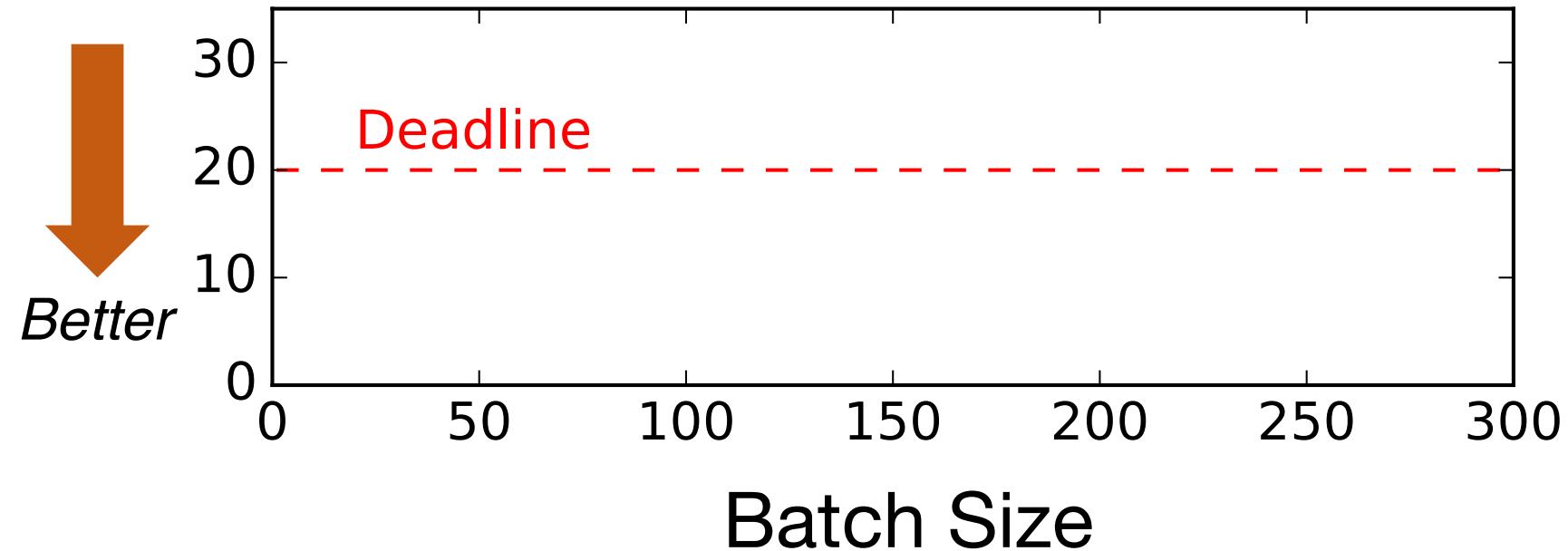


# Tensor Flow Conv. Net (GPU)

Throughput  
(Queries Per Second)

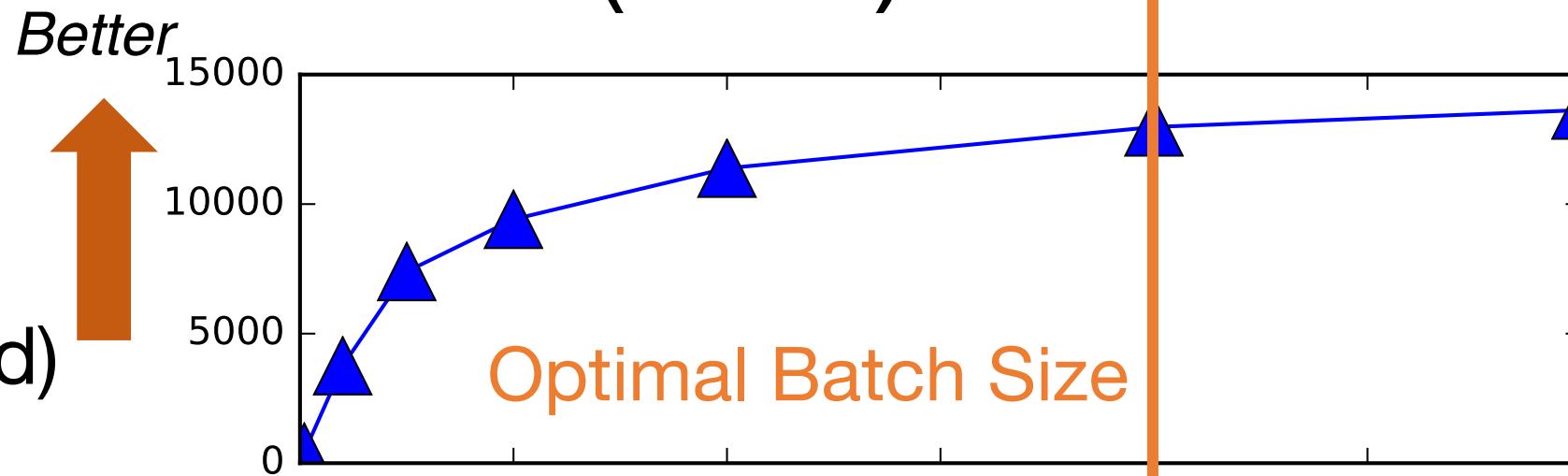


Latency (ms)

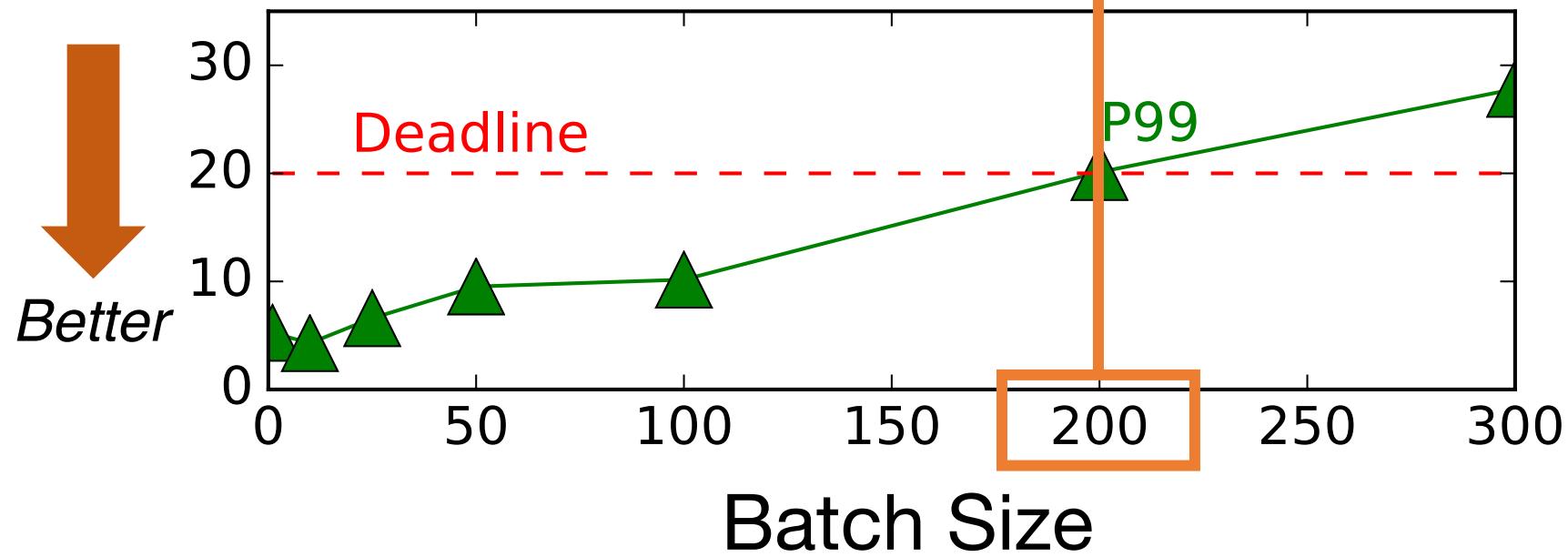


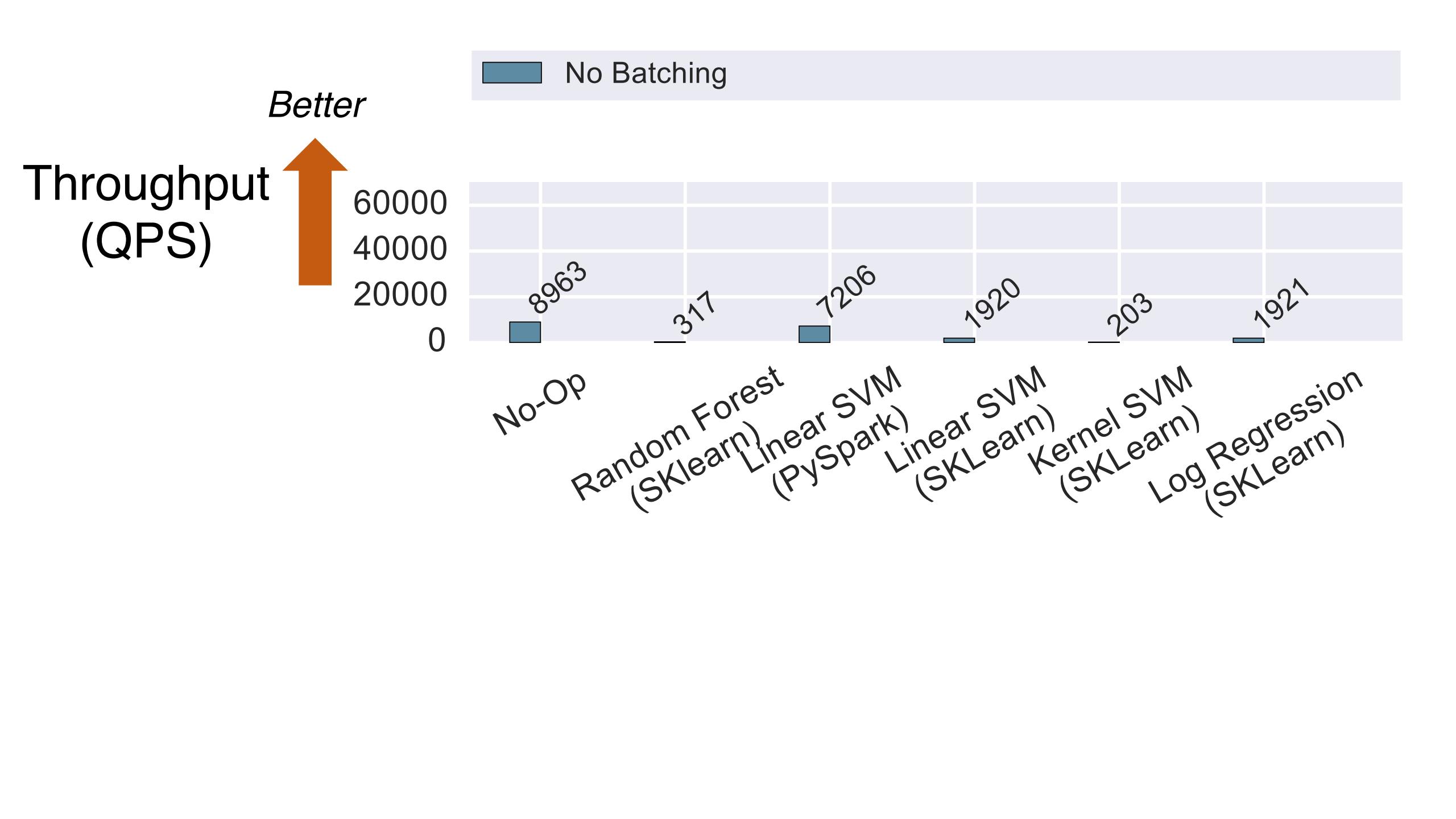
# Tensor Flow Conv. Net (GPU)

# Throughput (Queries Per Second)



# Latency (ms)

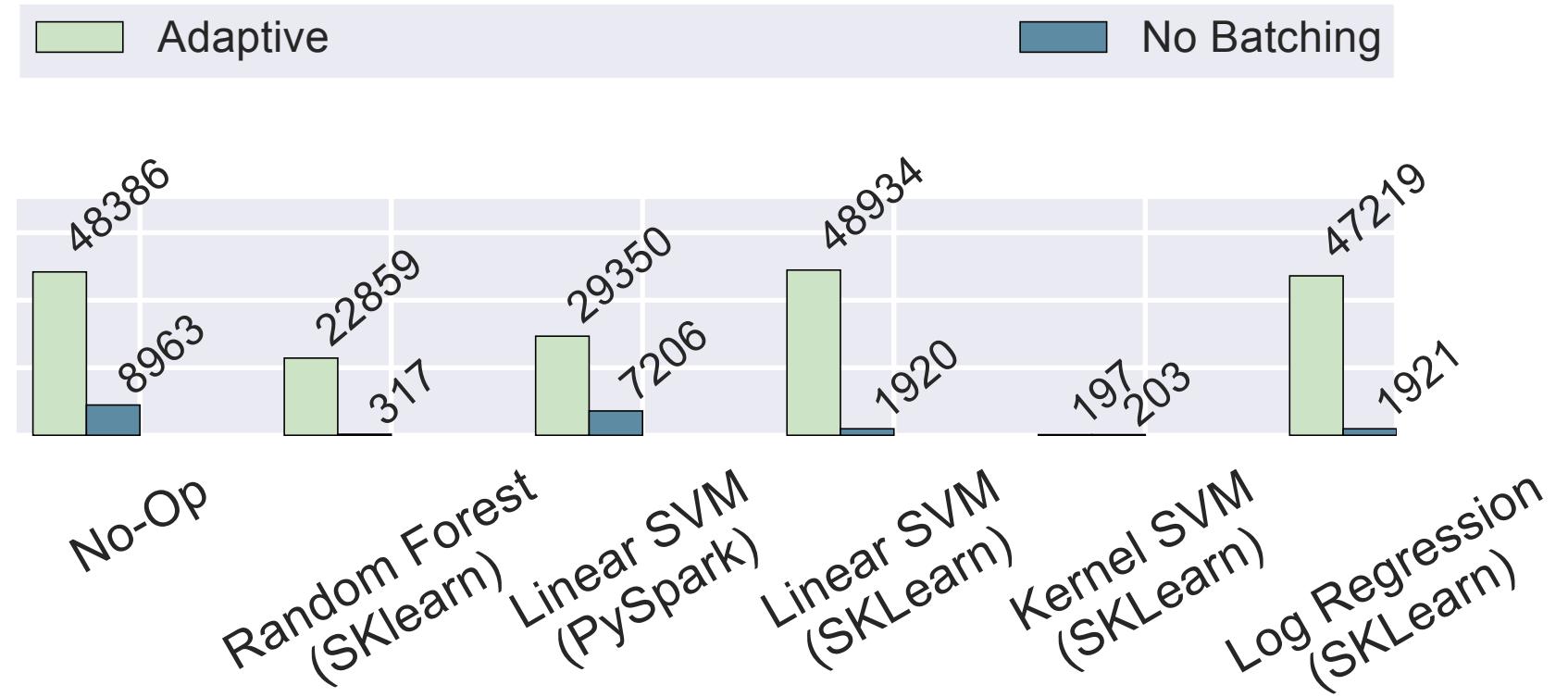




*Better*

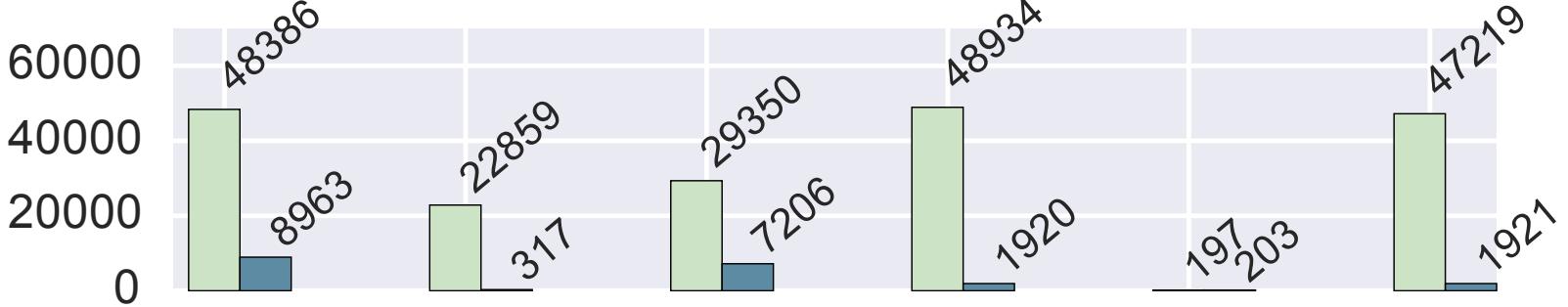
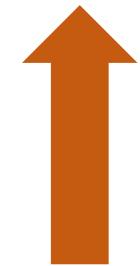
Throughput  
(QPS)

↑



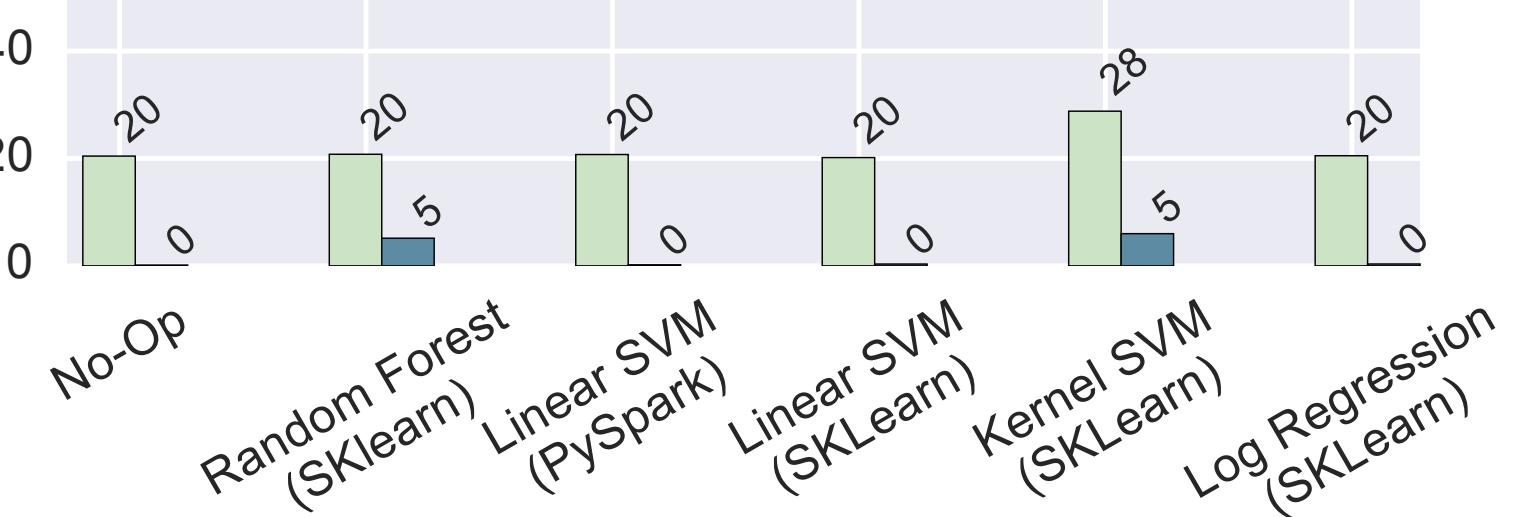
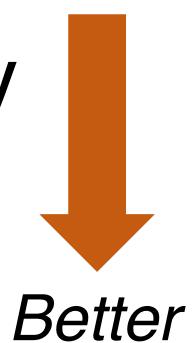
Throughput  
(QPS)

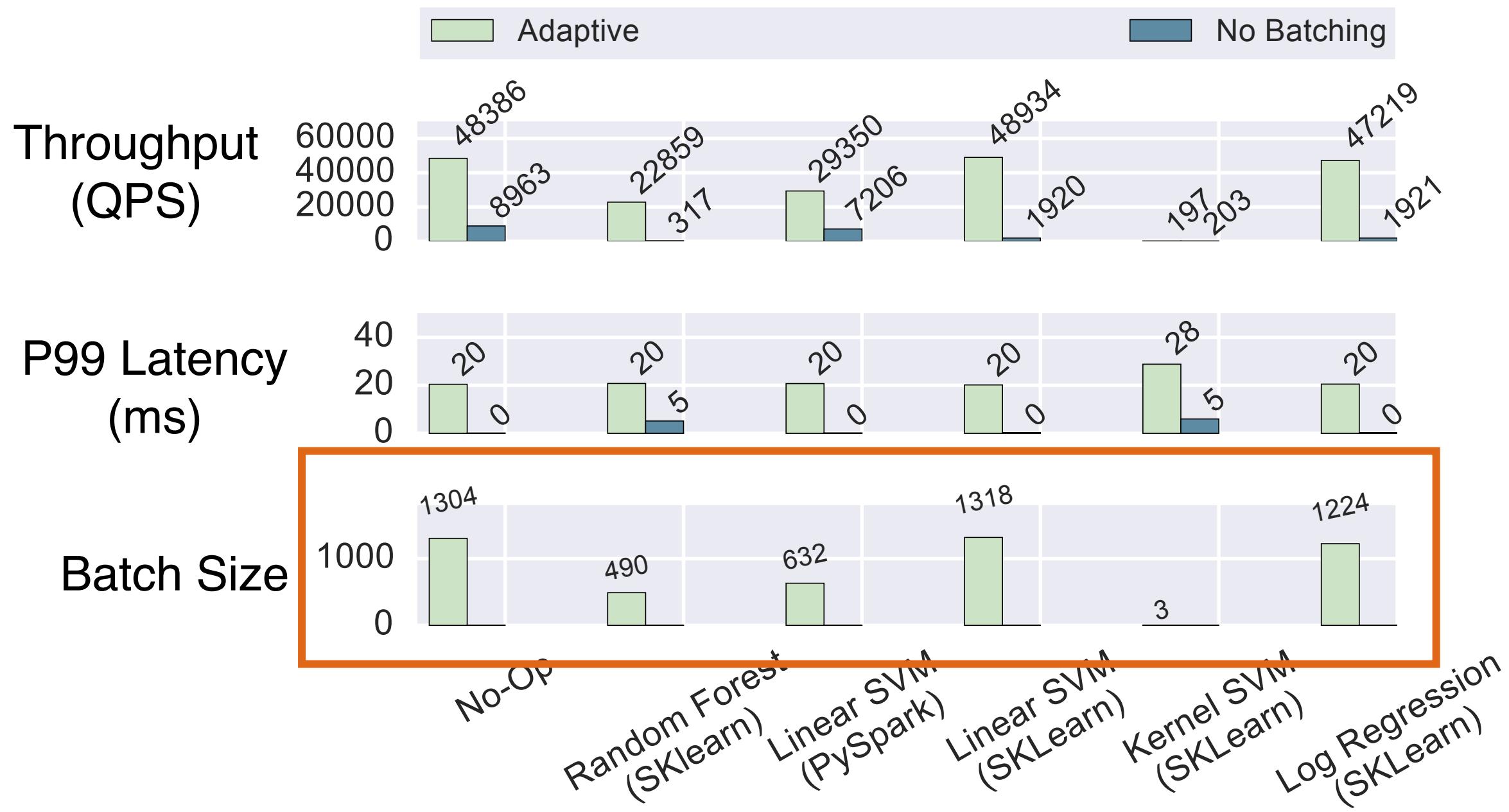
Better



P99 Latency  
(ms)

*20 ms is  
Fast Enough*





# Overhead of decoupled architecture

Applications



Predict

RPC/REST Interface

Feedback

Clipper

RPC

RPC

RPC

RPC

MC

MC

MC

MC

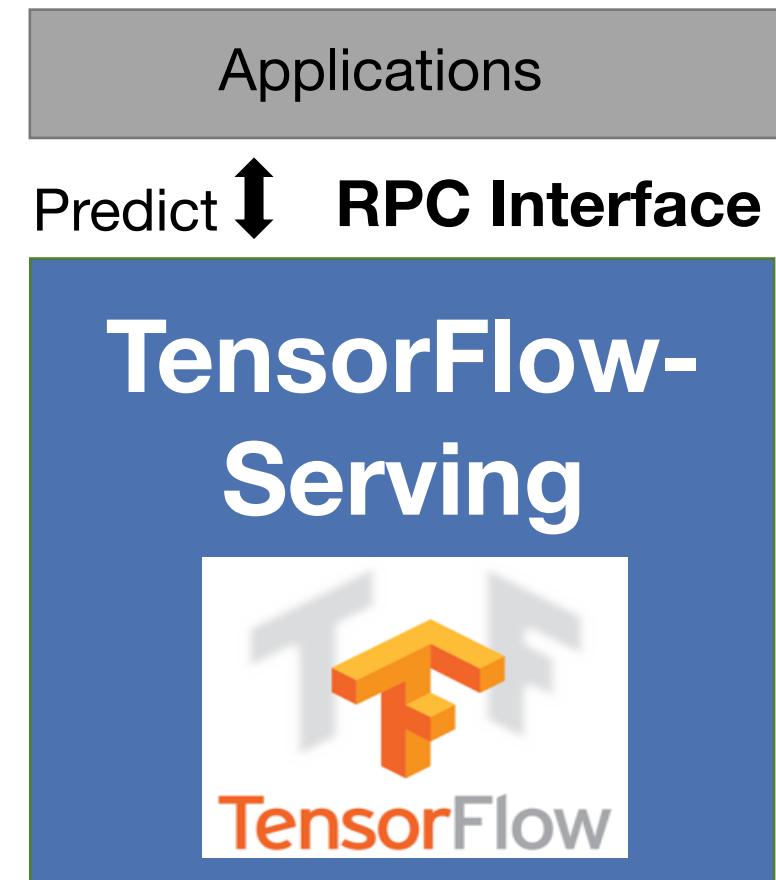
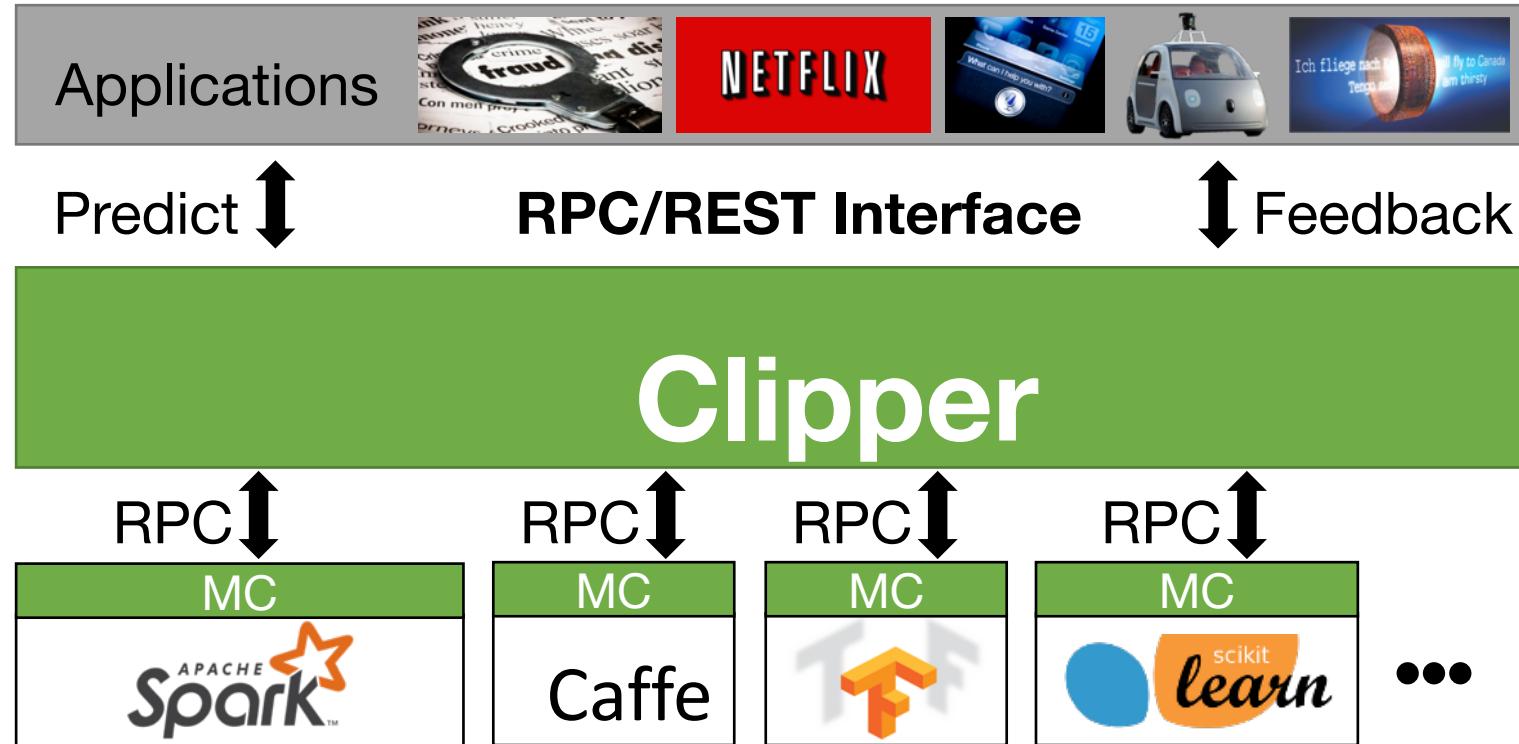


Caffe

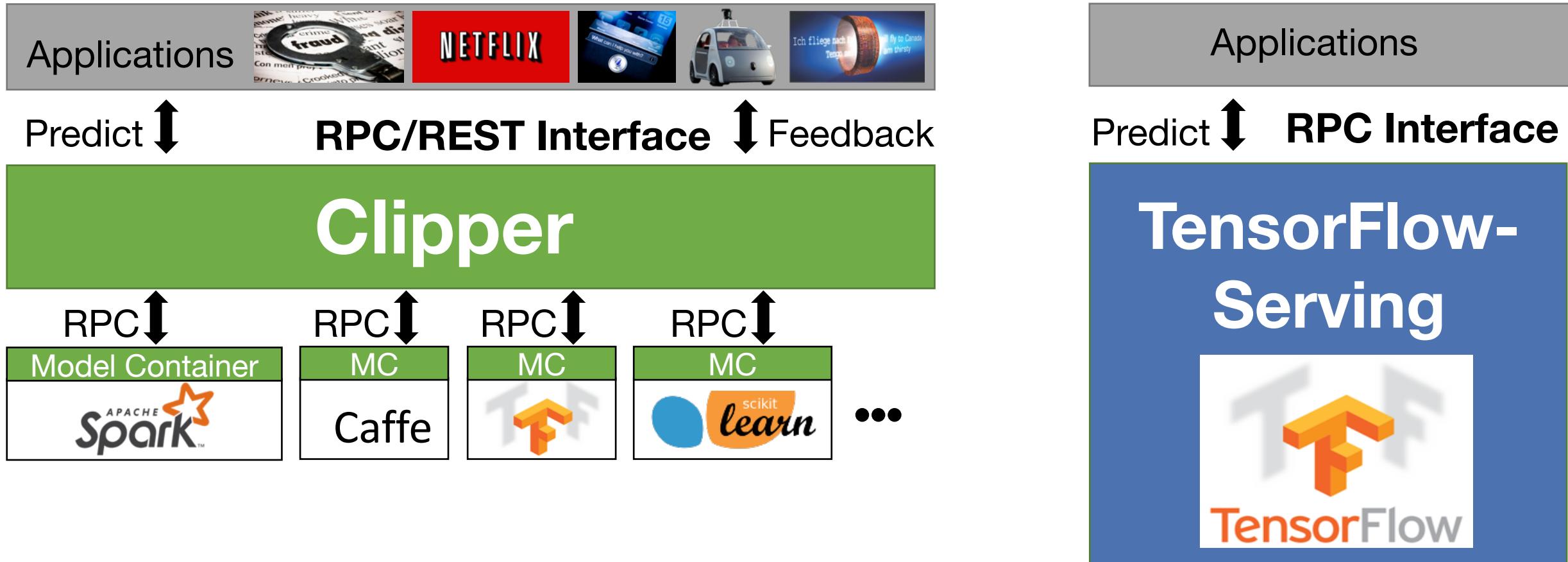


...

# Overhead of decoupled architecture

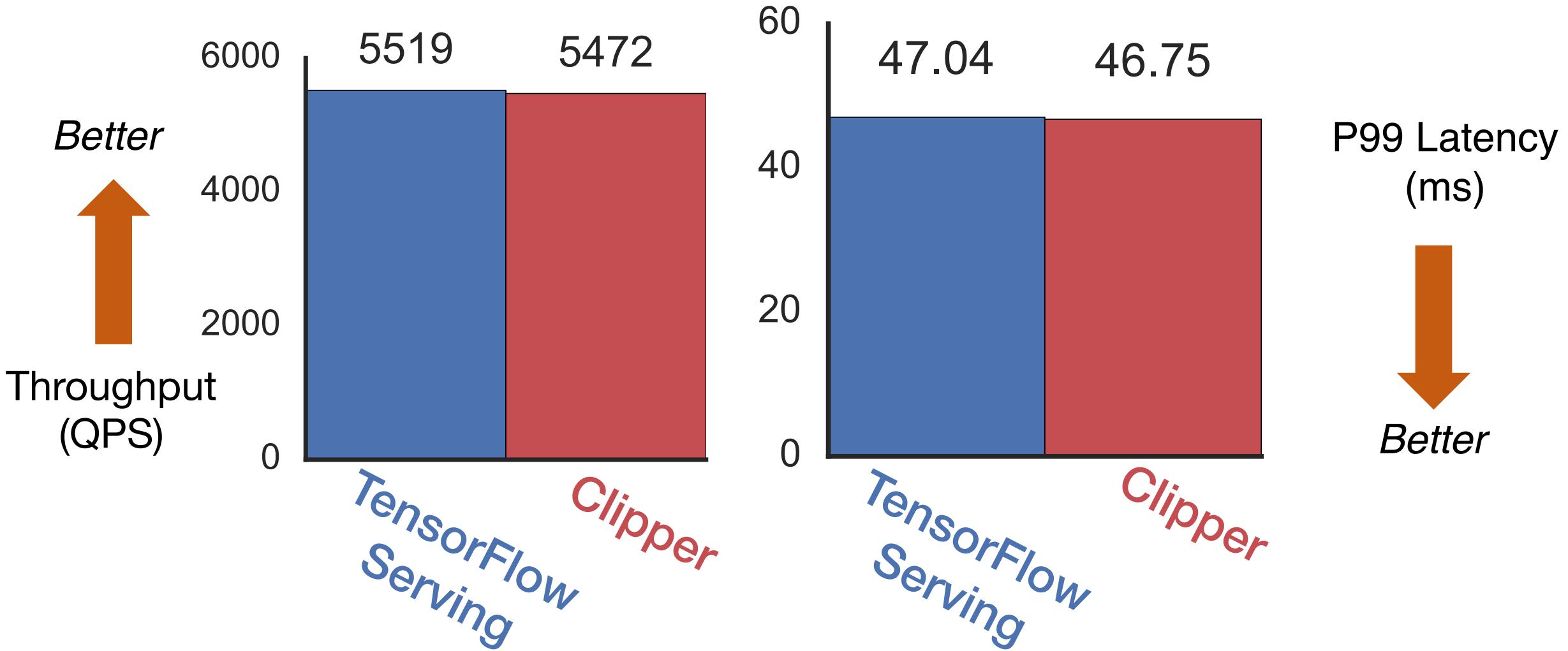


# Overhead of decoupled architecture



# Overhead of decoupled architecture

*Model: AlexNet trained on CIFAR-10*



# Clipper Architecture

Applications



NETFLIX



Predict

RPC/REST Interface

Observe

## Clipper

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

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

Model Selection Layer

Model Abstraction Layer

RPC

RPC

RPC

RPC

Model Container (MC)



MC  
Caffe

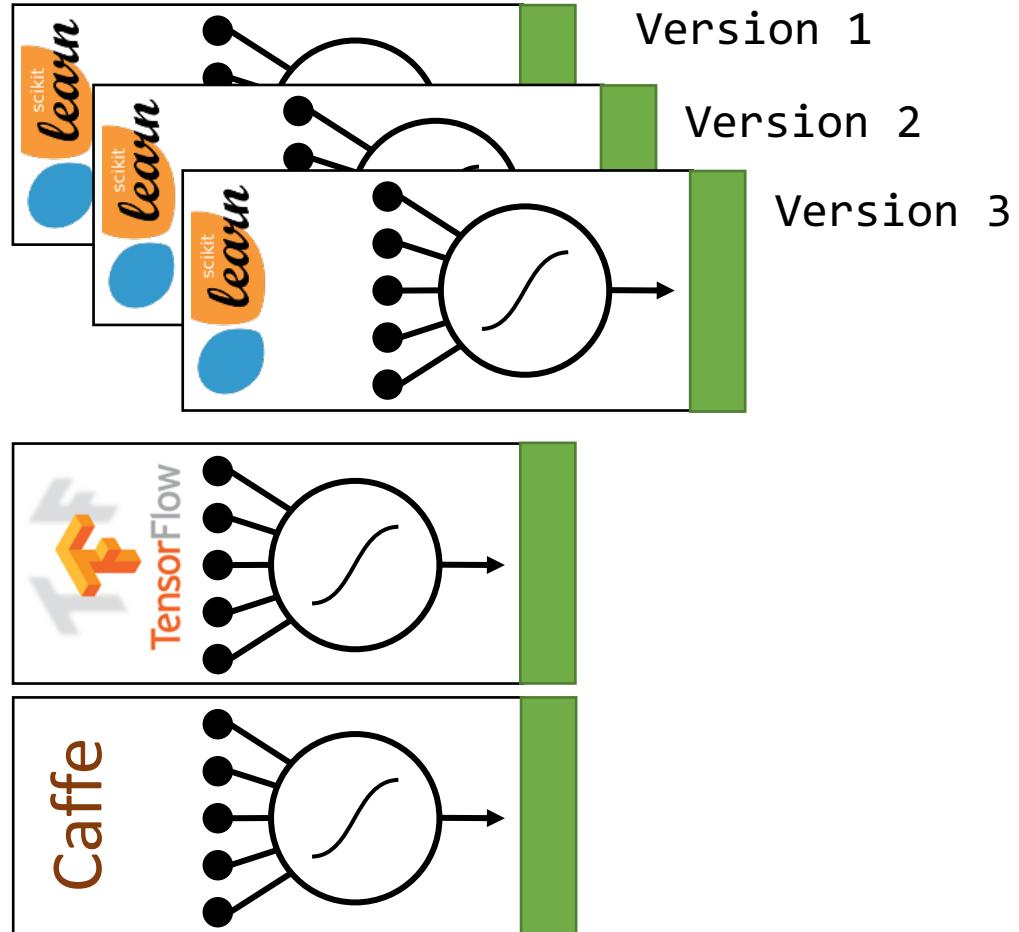
MC  
TensorFlow

MC  
scikit-learn

...

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

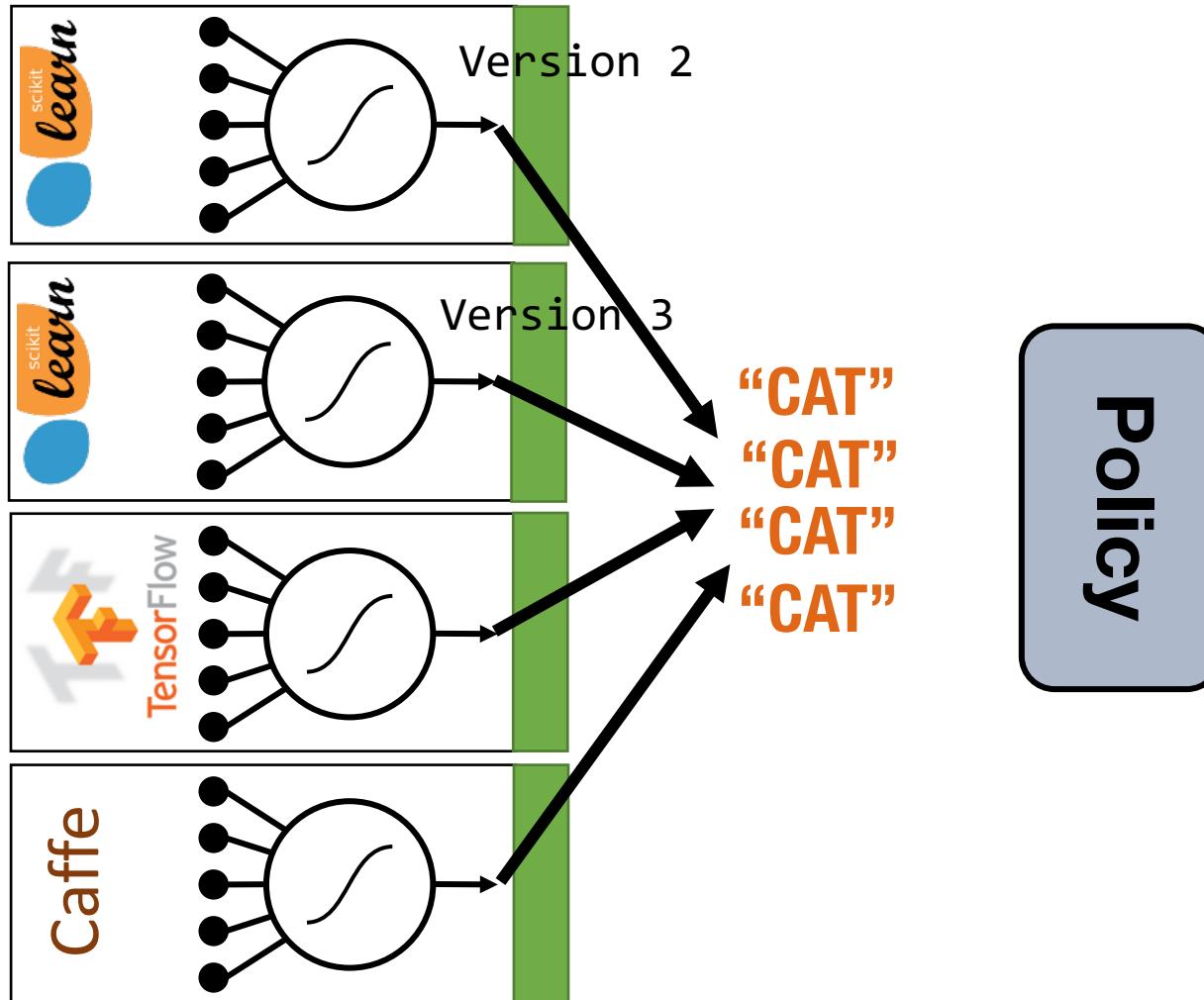
## Model Selection Layer



*Periodic retraining*

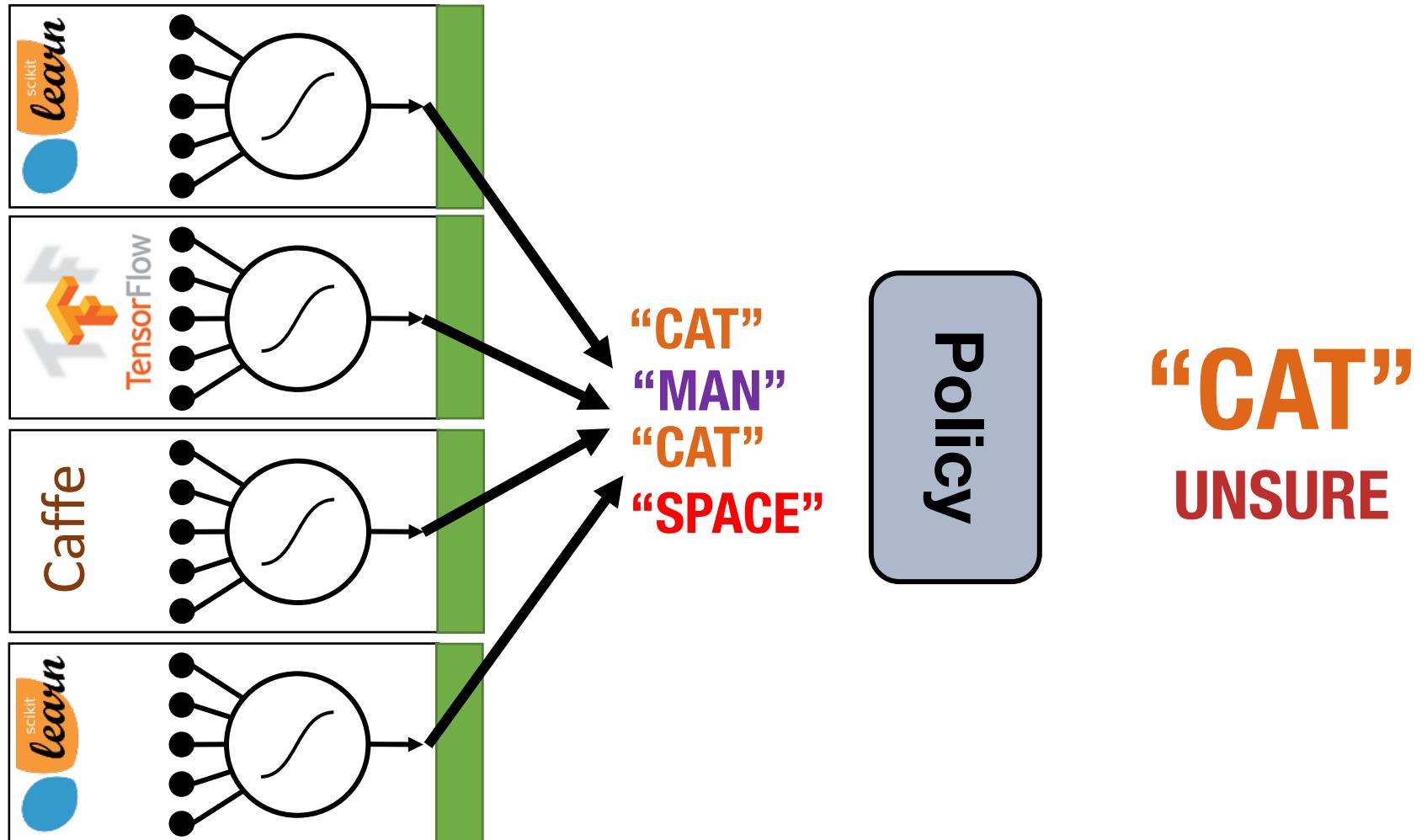
*Experiment with new models and frameworks*

# Selection Policy: Estimate confidence



“CAT”  
CONFIDENT

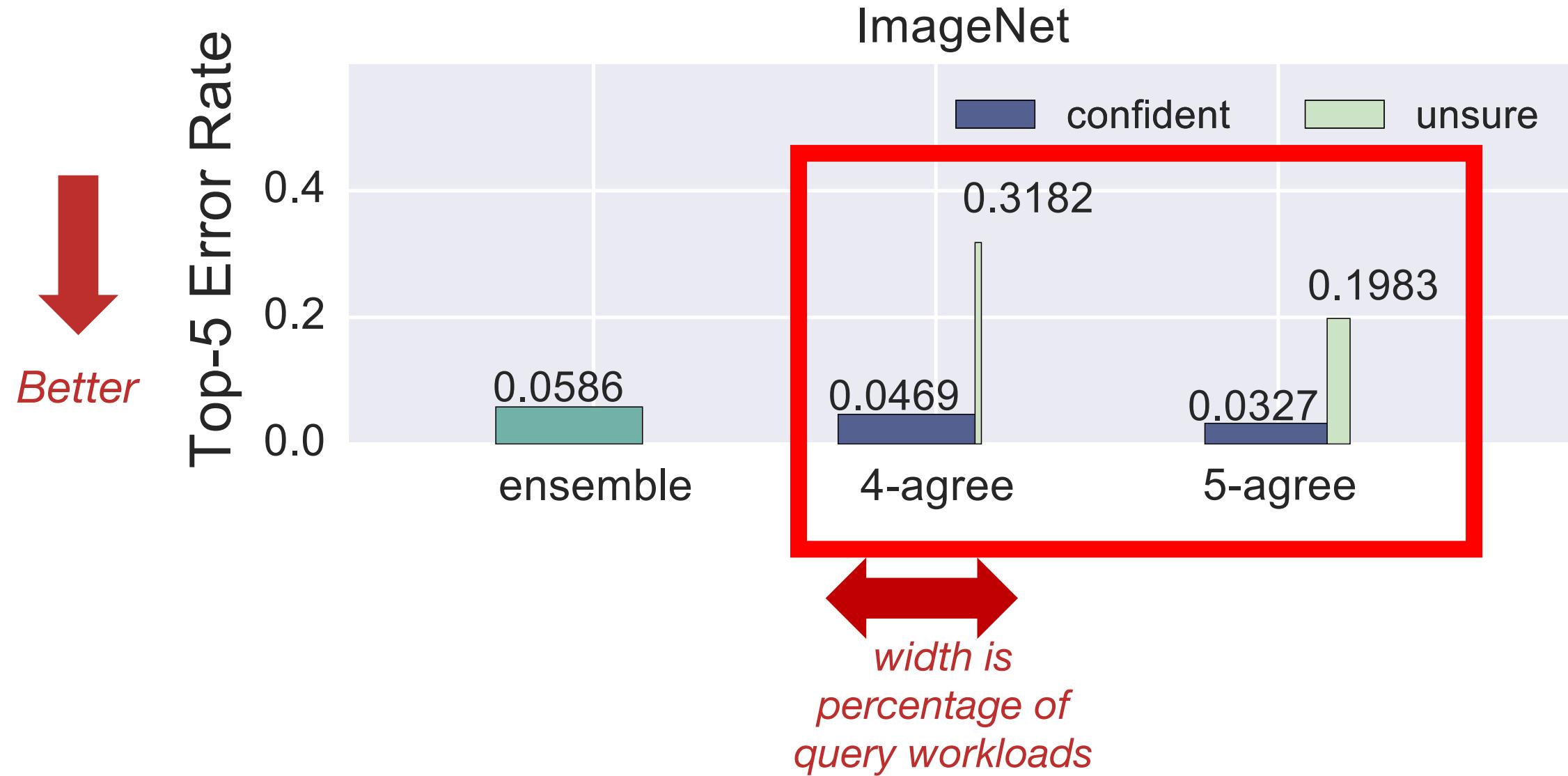
# Selection Policy: Estimate confidence



# Selection Policy: Estimate confidence



# Selection Policy: Estimate confidence



## *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*

