Clipper

A Low-Latency Online Prediction Serving System

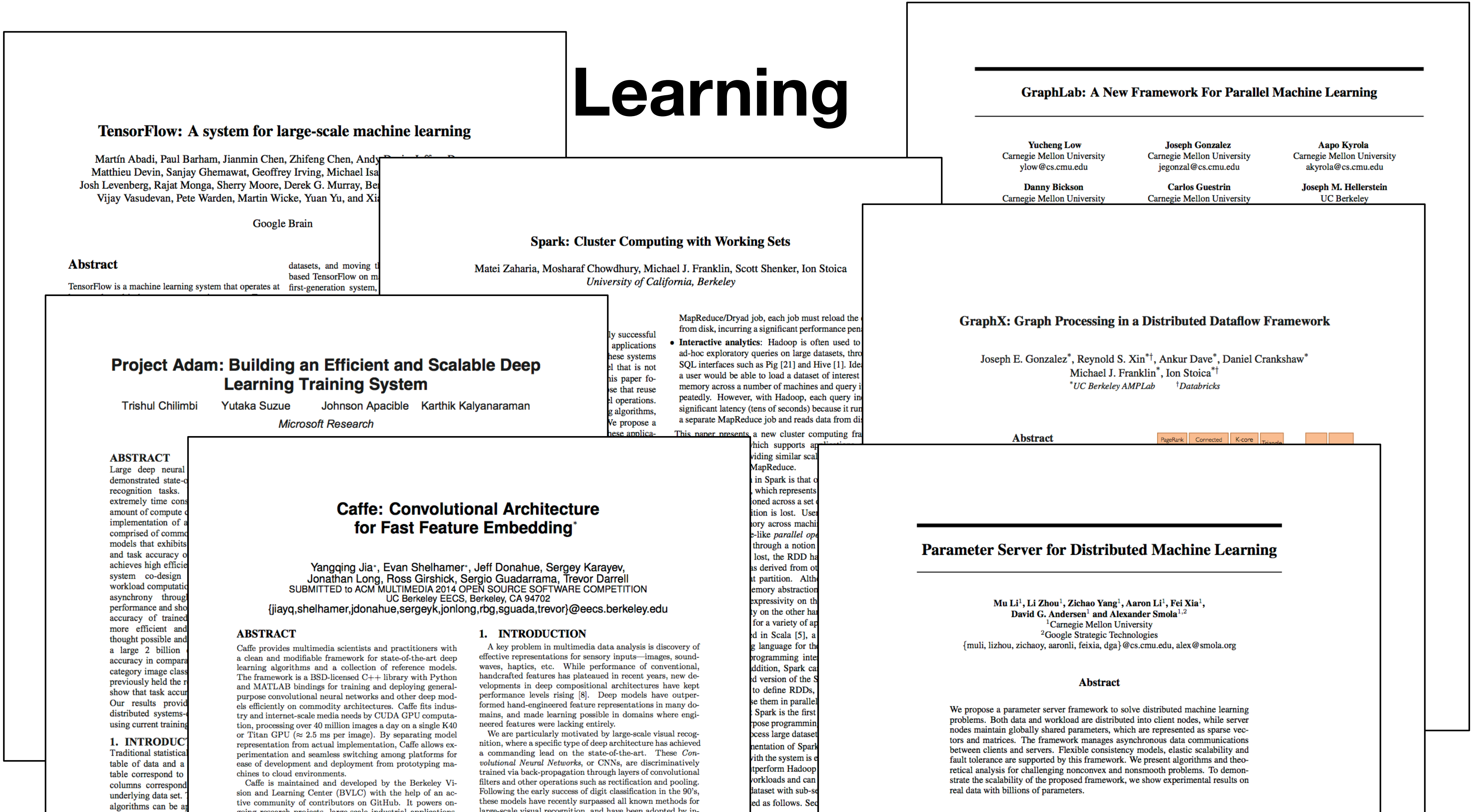
Daniel Crankshaw

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

NSDI 2017

March 29, 2017

Learning

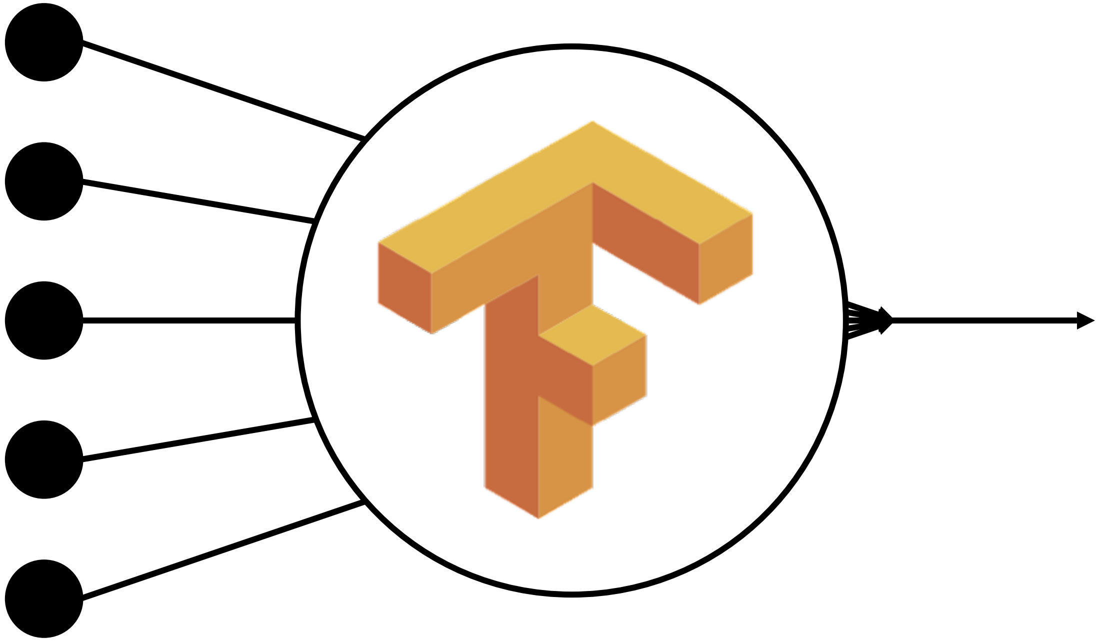

TensorFlow: A system for large-scale machine learning
Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy
Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isa
Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Be
Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xi
Google Brain
Abstract
Spark: Cluster Computing with Working Sets
Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica
University of California, Berkeley
GraphLab: A New Framework For Parallel Machine Learning
Yucheng Low, Carnegie Mellon University, ylow@cs.cmu.edu
Joseph Gonzalez, Carnegie Mellon University, jegonzal@cs.cmu.edu
Aapo Kyrola, Carnegie Mellon University, 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*†, Ankur Dave*, Daniel Crankshaw*
Michael J. Franklin*, Ion Stoica*†
*UC Berkeley AMPLab †Databricks
Abstract
Project Adam: Building an Efficient and Scalable Deep Learning Training System
Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, Karthik Kalyanaraman
Microsoft Research
ABSTRACT
Caffe: Convolutional Architecture for Fast Feature Embedding*
Yangqing Jia*, Evan Shelhamer*, 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,sguada,trevor}@eecs.berkeley.edu
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 (≈ 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.
1. INTRODUCTION
A key problem in multimedia data analysis is discovery of effective representations for sensory inputs—images, soundwaves, 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
Parameter Server for Distributed Machine Learning
Mu Li[1], Li Zhou[1], Zichao Yang[1], Aaron Li[1], Fei Xia[1], David G. Andersen[1] and Alexander Smola[1,2]
[1]Carnegie Mellon University
[2]Google Strategic Technologies
{muli, lizhou, zichaoy, aaronli, feixia, dga}@cs.cmu.edu, 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

Learning Produces a Trained Model

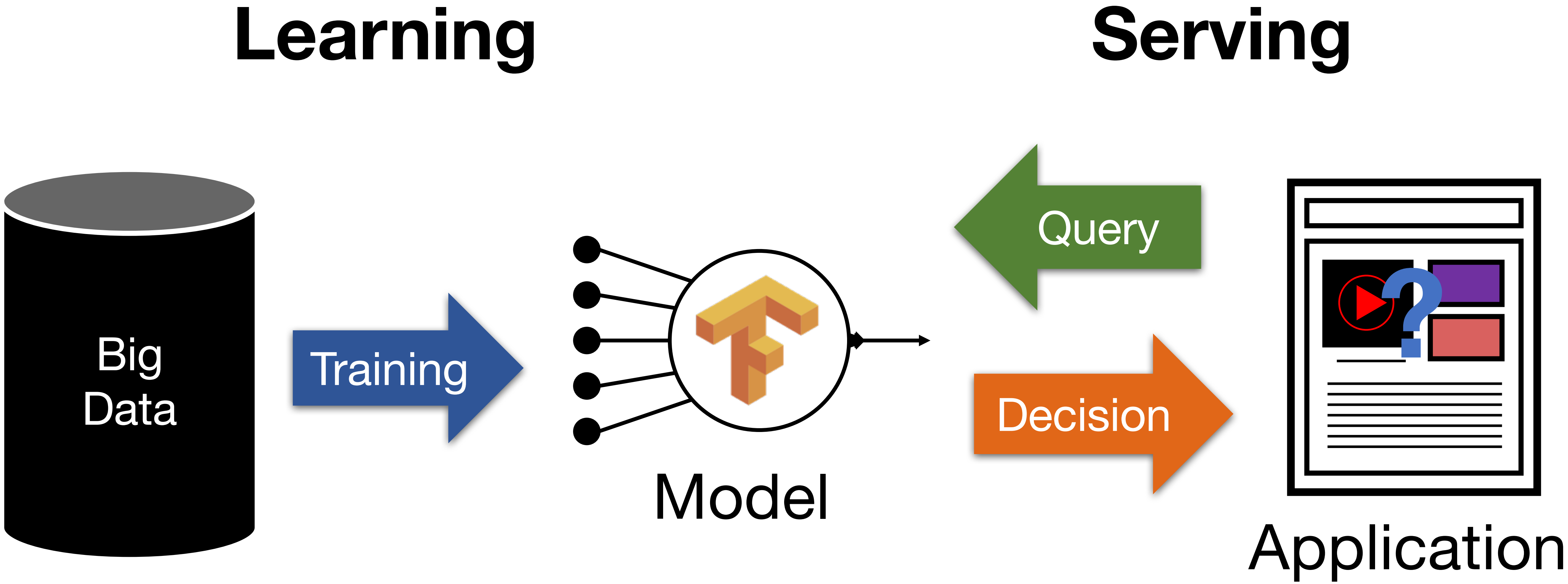
Query

Decision

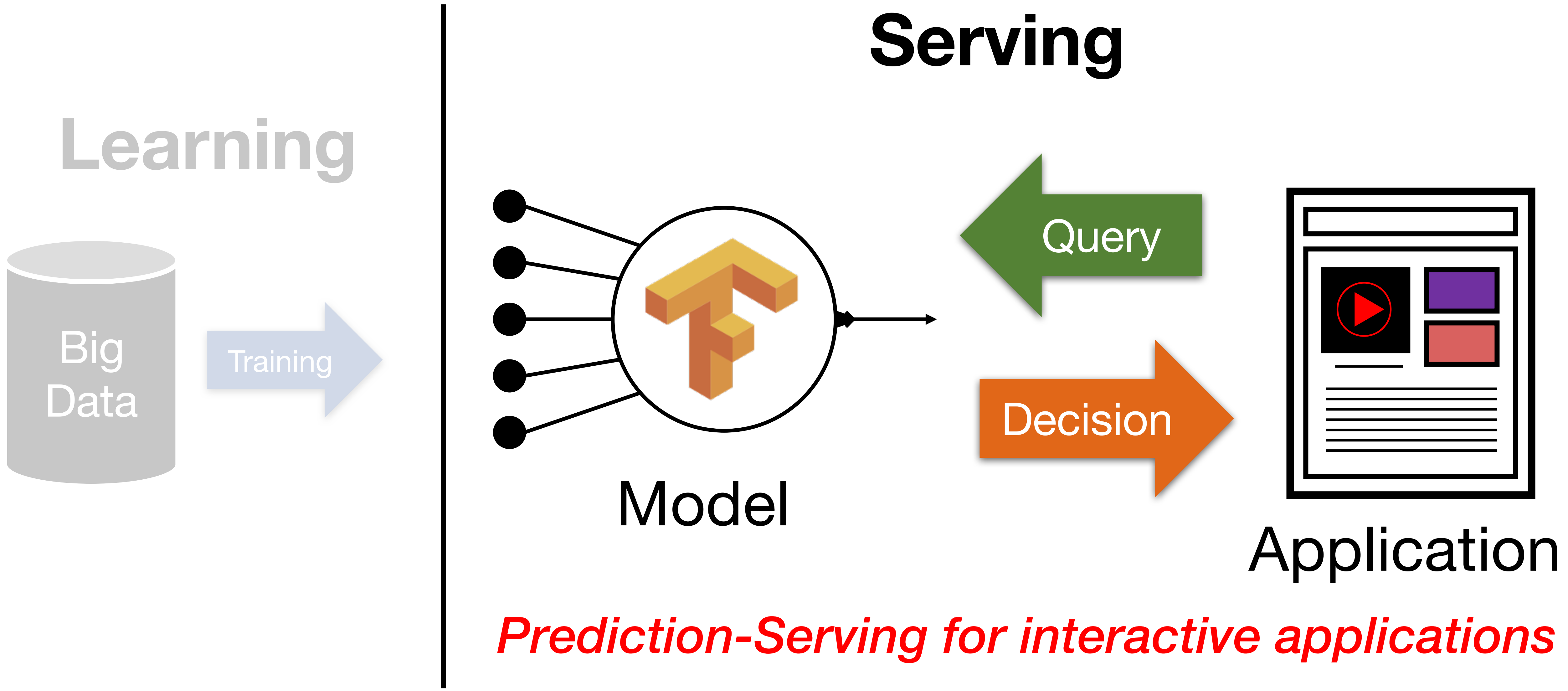


“CAT”

Model


Learning
Serving
Big
Data
Training
Model
Query
Decision
Application

Serving

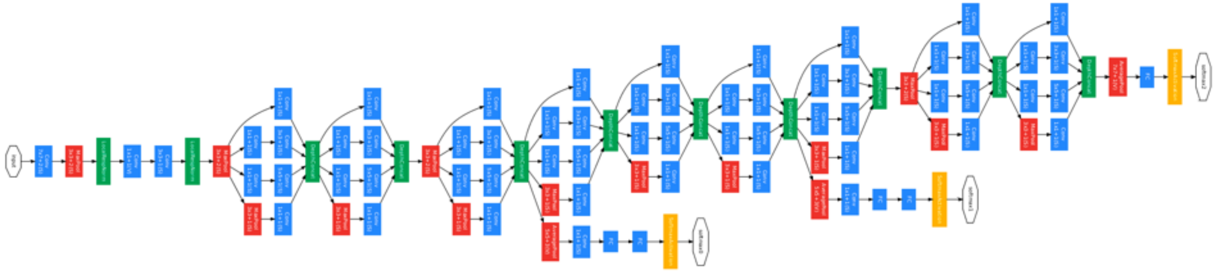


Prediction-Serving for interactive applications

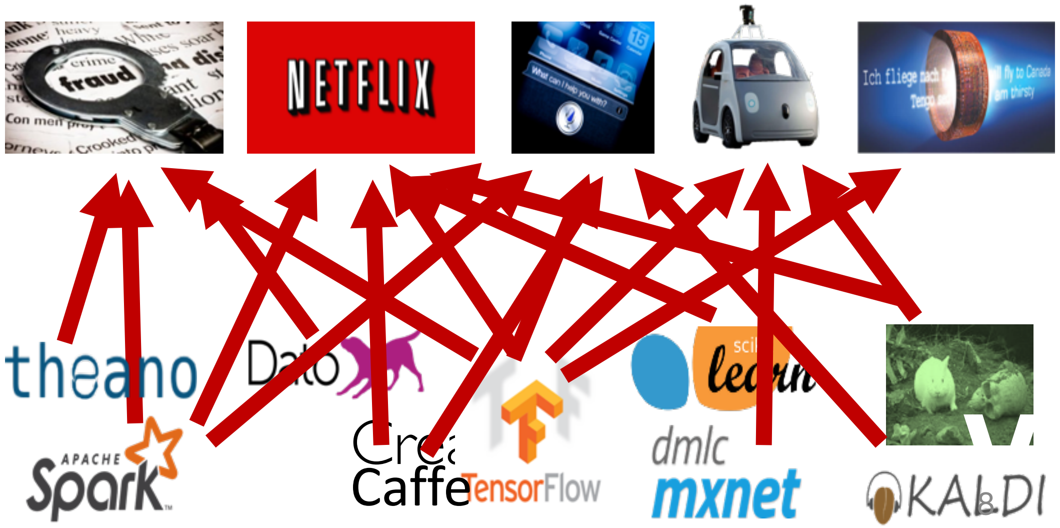
Timescale: ~10s of milliseconds

Prediction-Serving Raises New Challenges

Prediction-Serving Challenges

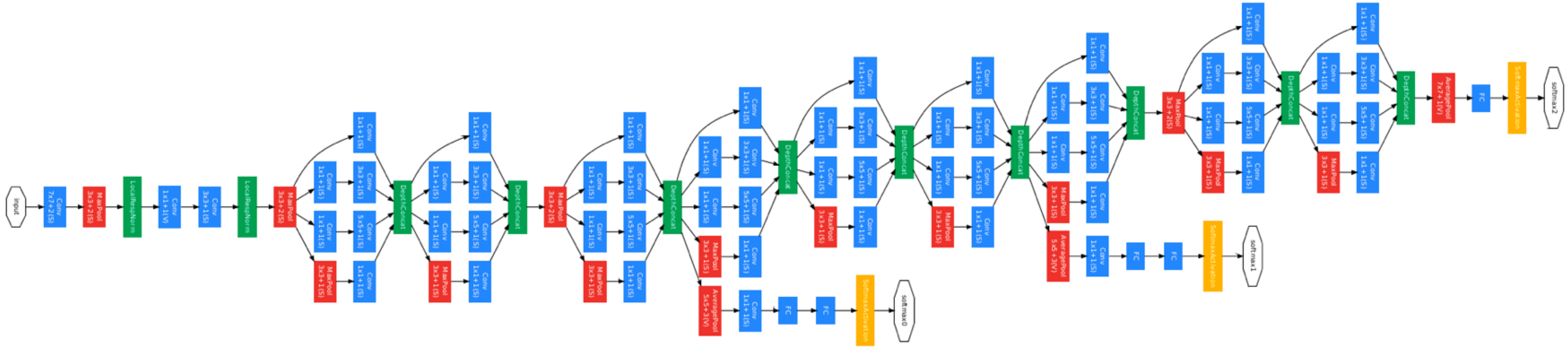


Support low-latency, high-throughput serving workloads



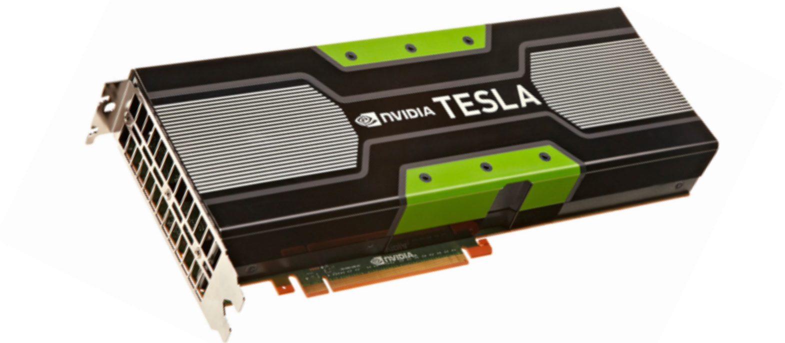
Large and growing ecosystem of ML models and frameworks

Support low-latency, high-throughput serving workloads



Models getting more complex

- *10s of GFLOPs [1]*



Deployed on critical path

- *Maintain SLOs under heavy load*

Using specialized hardware for predictions

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

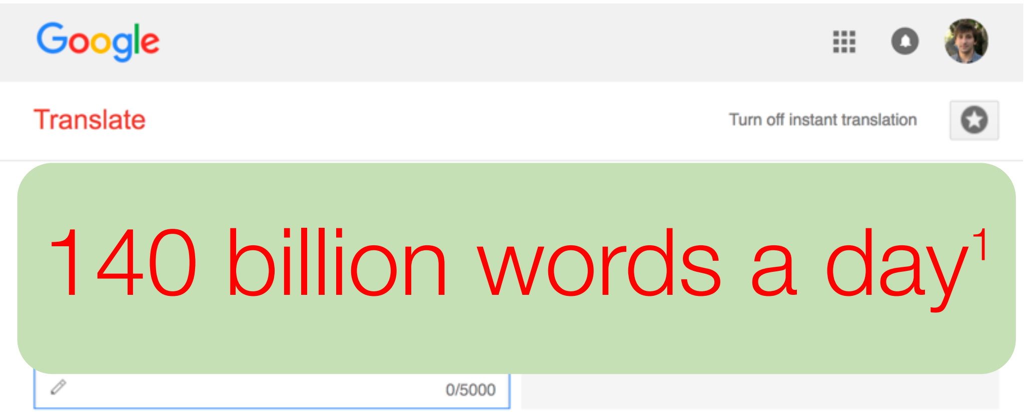
Google Translate

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

Serving



140 billion words a day[1]

82,000 GPUs
running 24/7

Invented New Hardware!
Tensor Processing Unit (TPU)

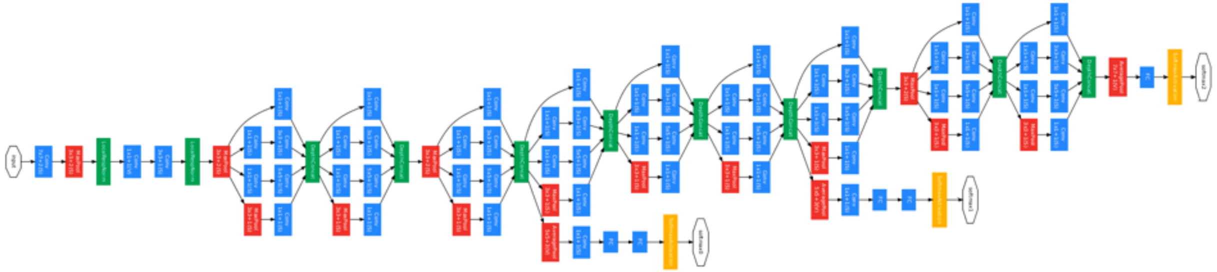
[1] https://www.nytimes.com/2016/12/14/magazine/the-great-ai-awakening.html

Big Companies Build One-Off Systems

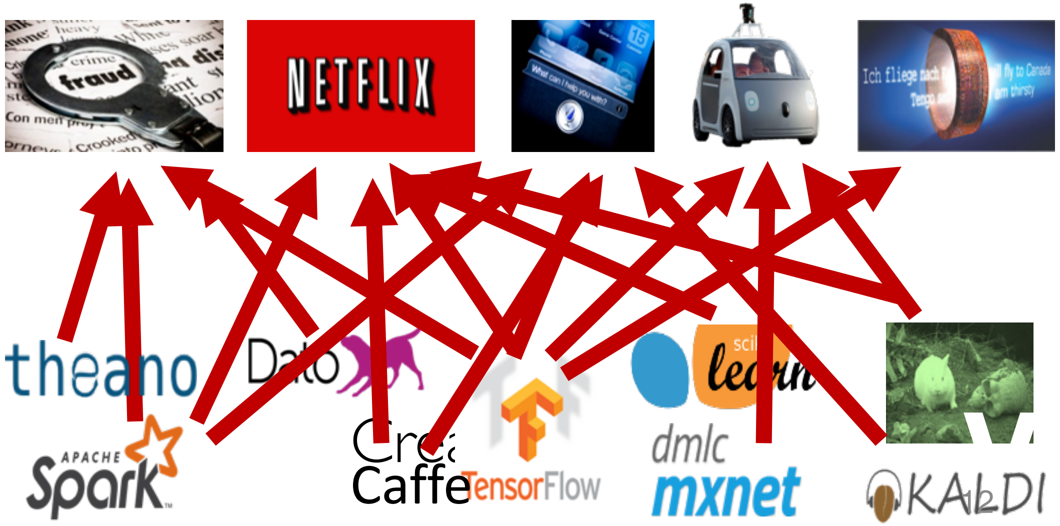
Problems:

- ➢ 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



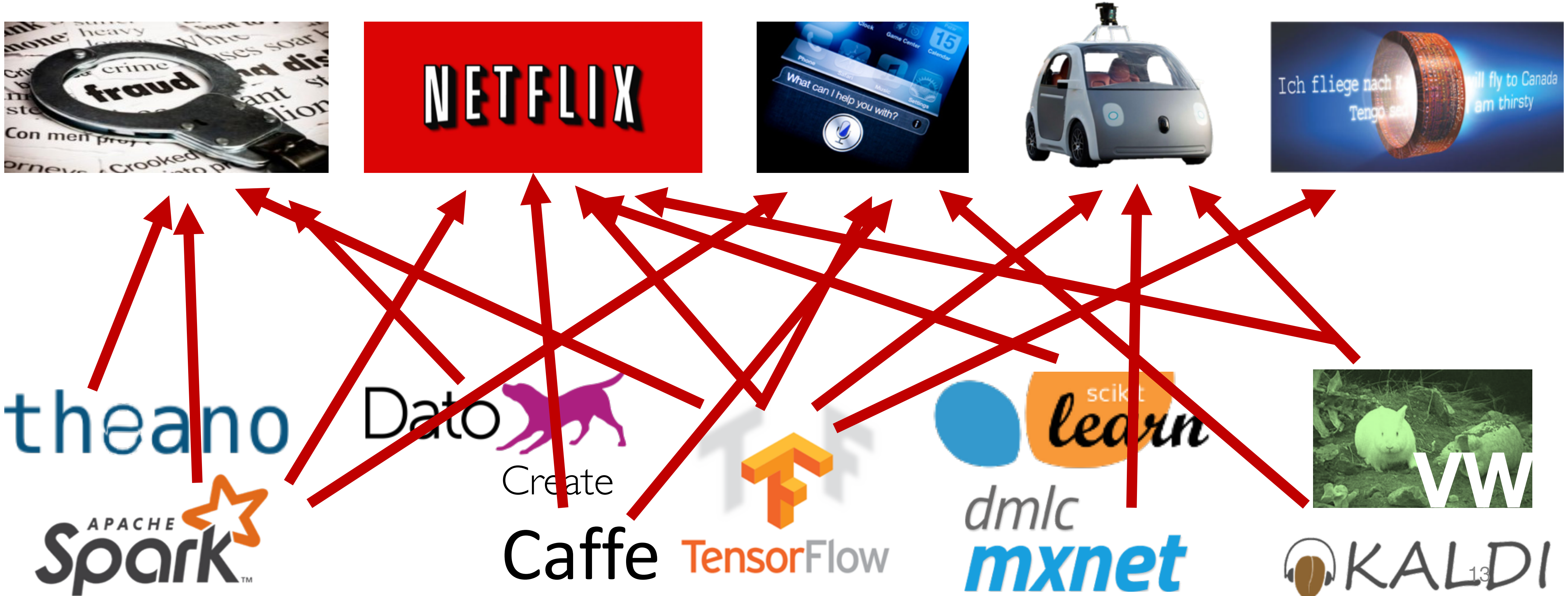
Support low-latency, high-throughput serving workloads



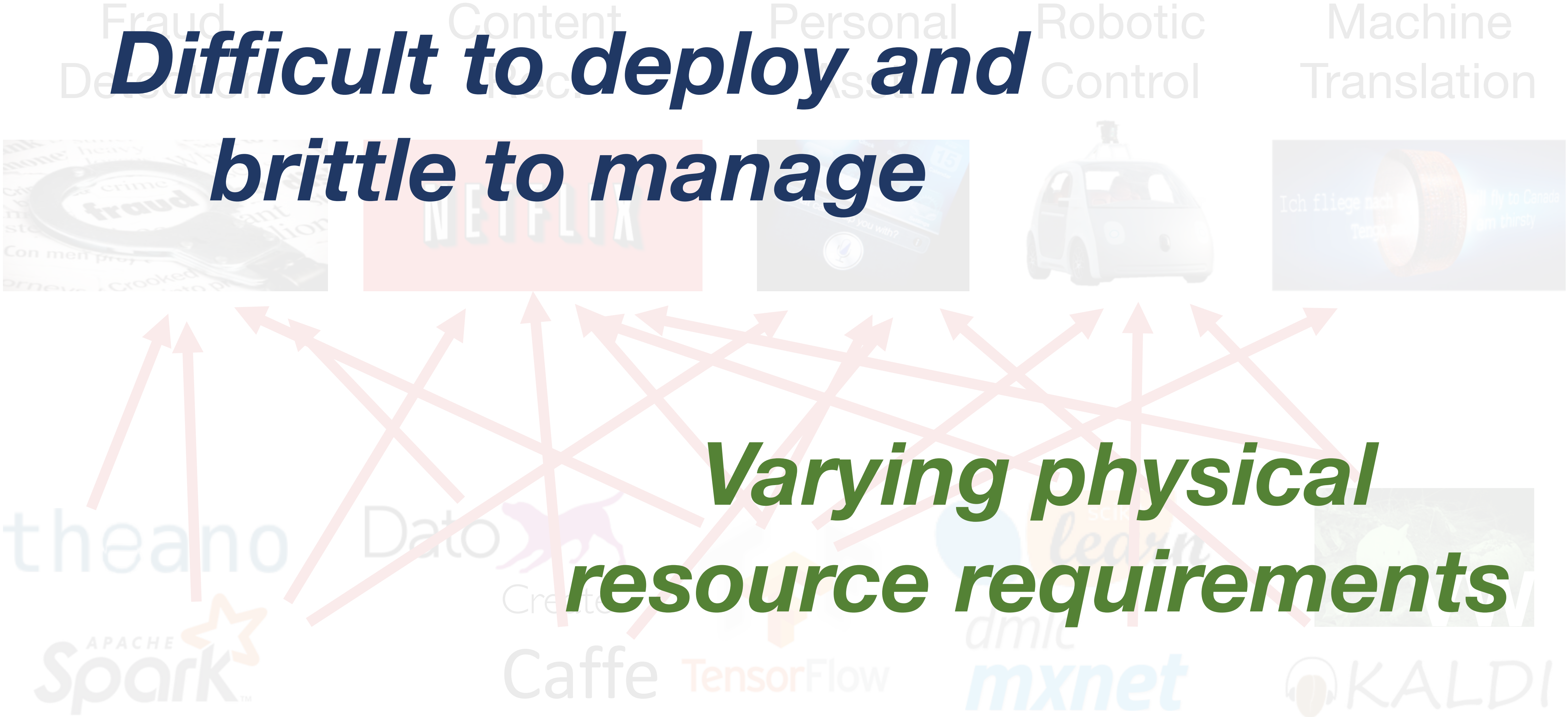
Large and growing ecosystem of ML models and frameworks

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

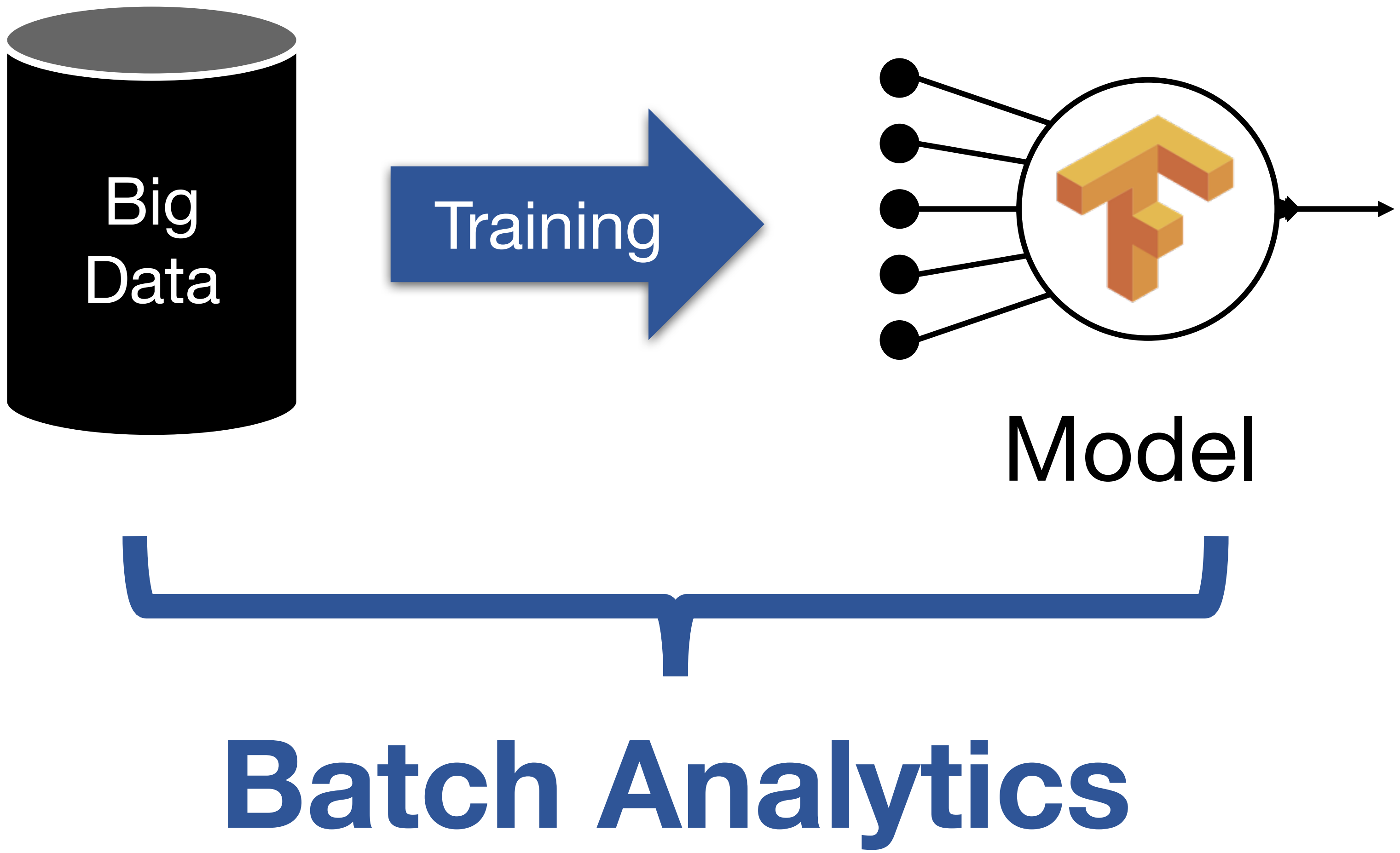


Difficult to deploy and brittle to manage

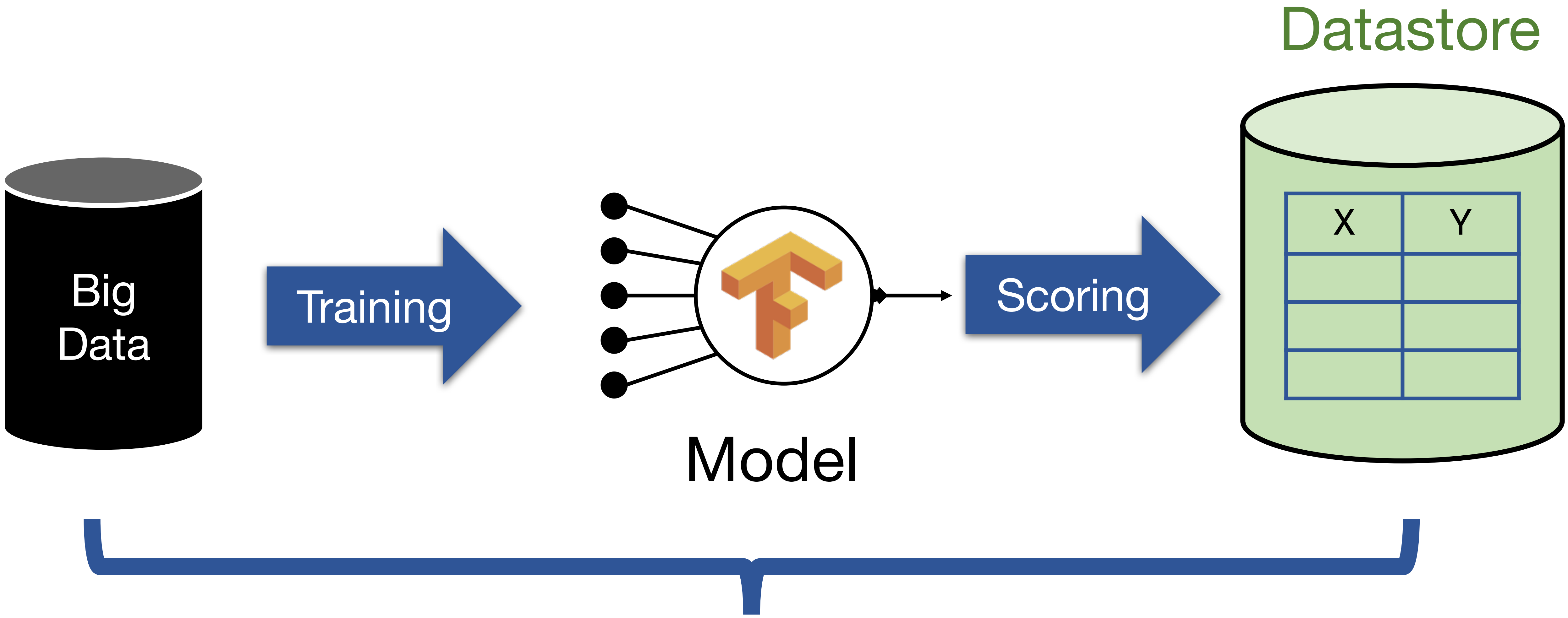
Varying physical resource requirements

But most companies can’t build new serving systems…

Use existing systems: Offline Scoring



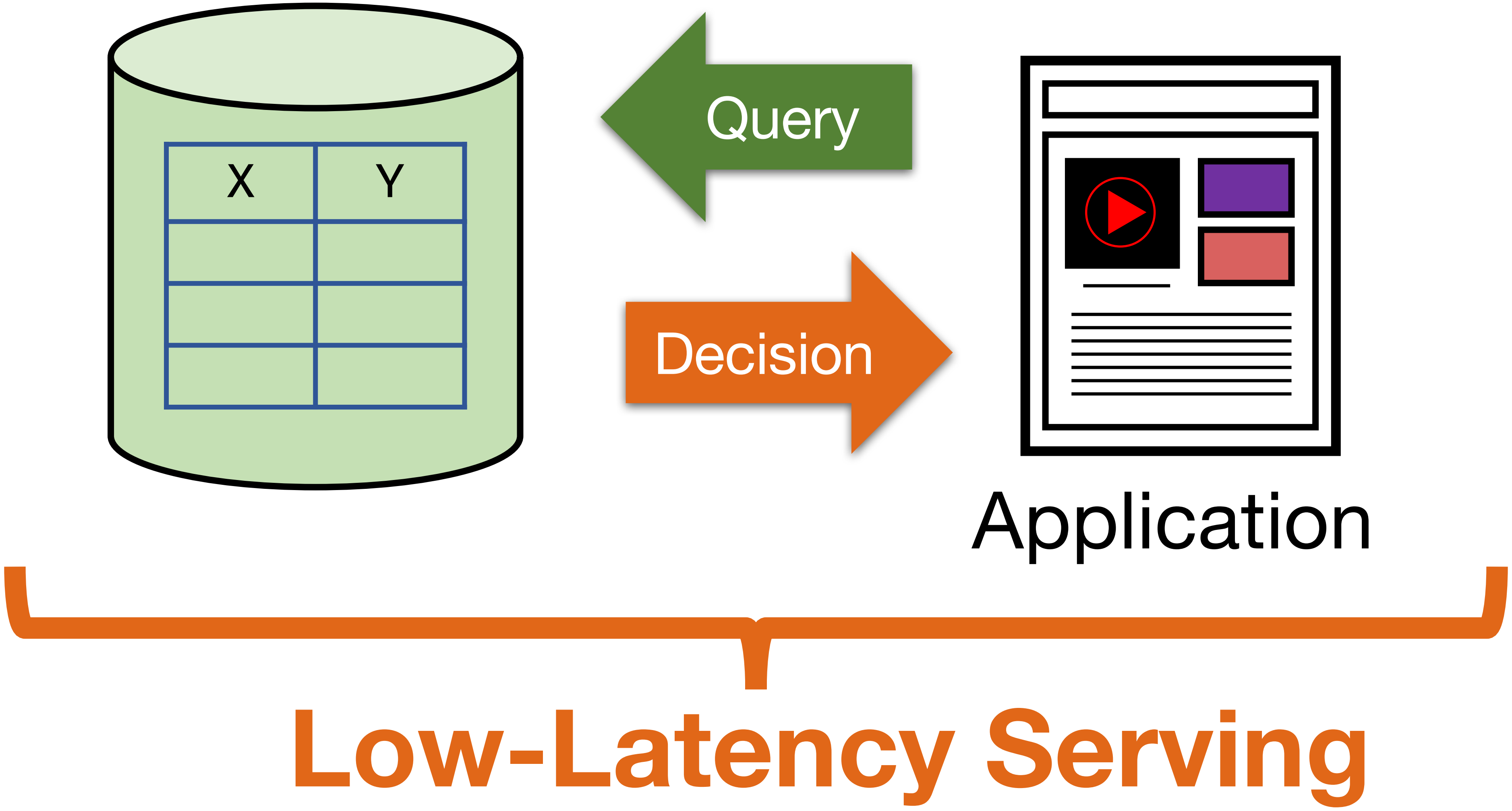
Use existing systems: Offline Scoring



Batch Analytics

Use existing systems: Offline Scoring

Look up decision in datastore

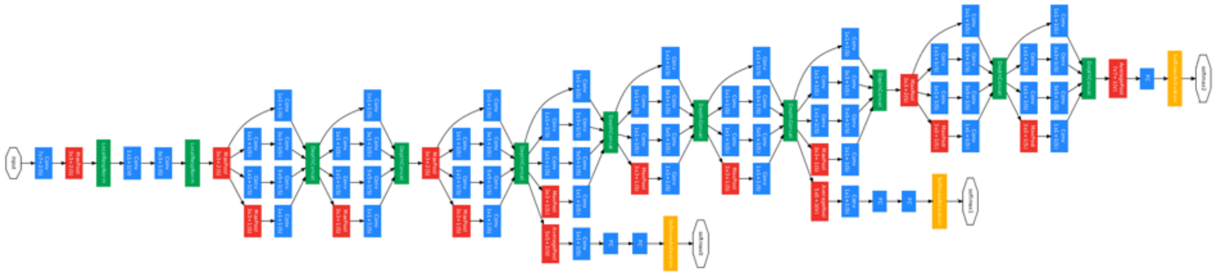


Use existing systems: Offline Scoring

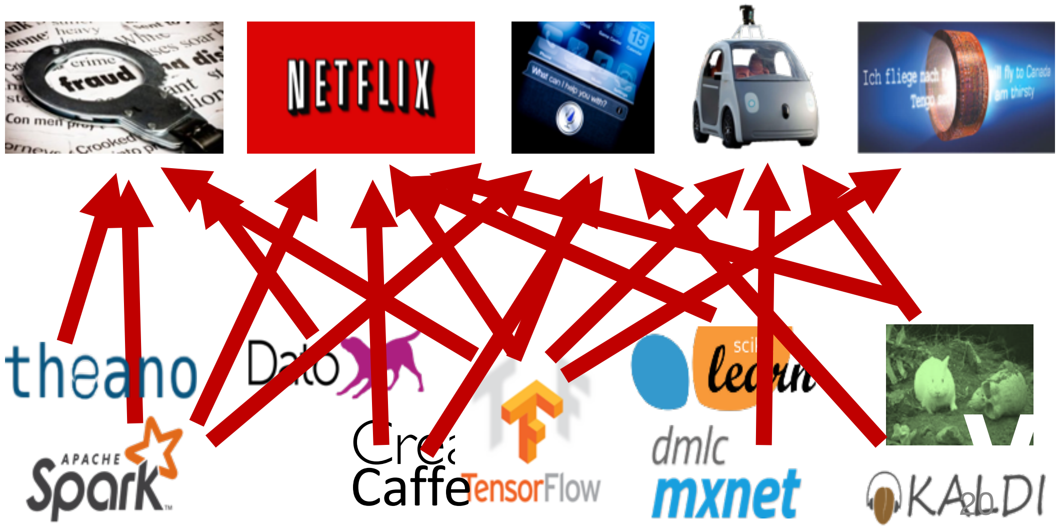
Problems:

- ➢ 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



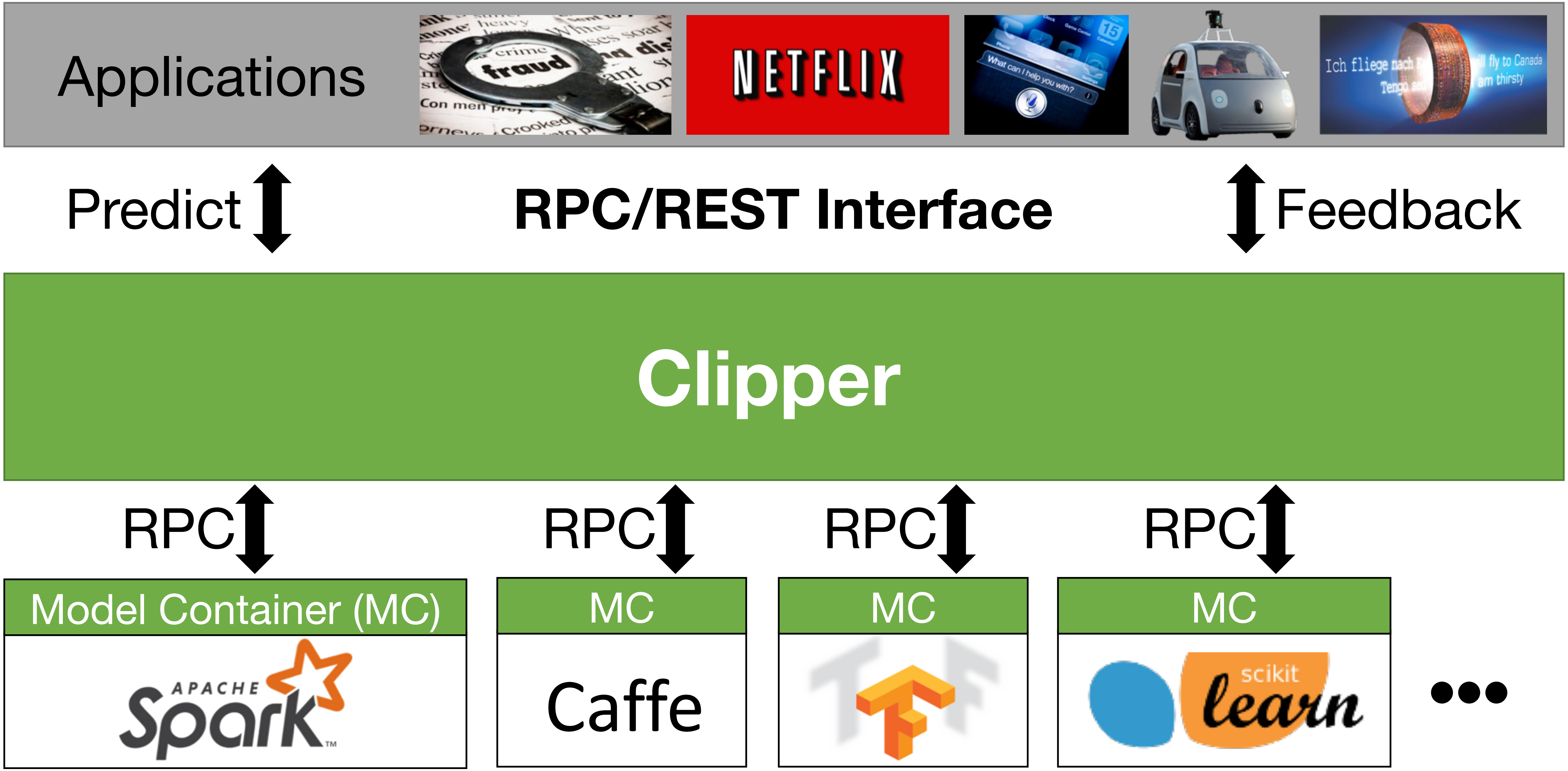
Large and growing ecosystem of ML models and frameworks

How does Clipper address these challenges?

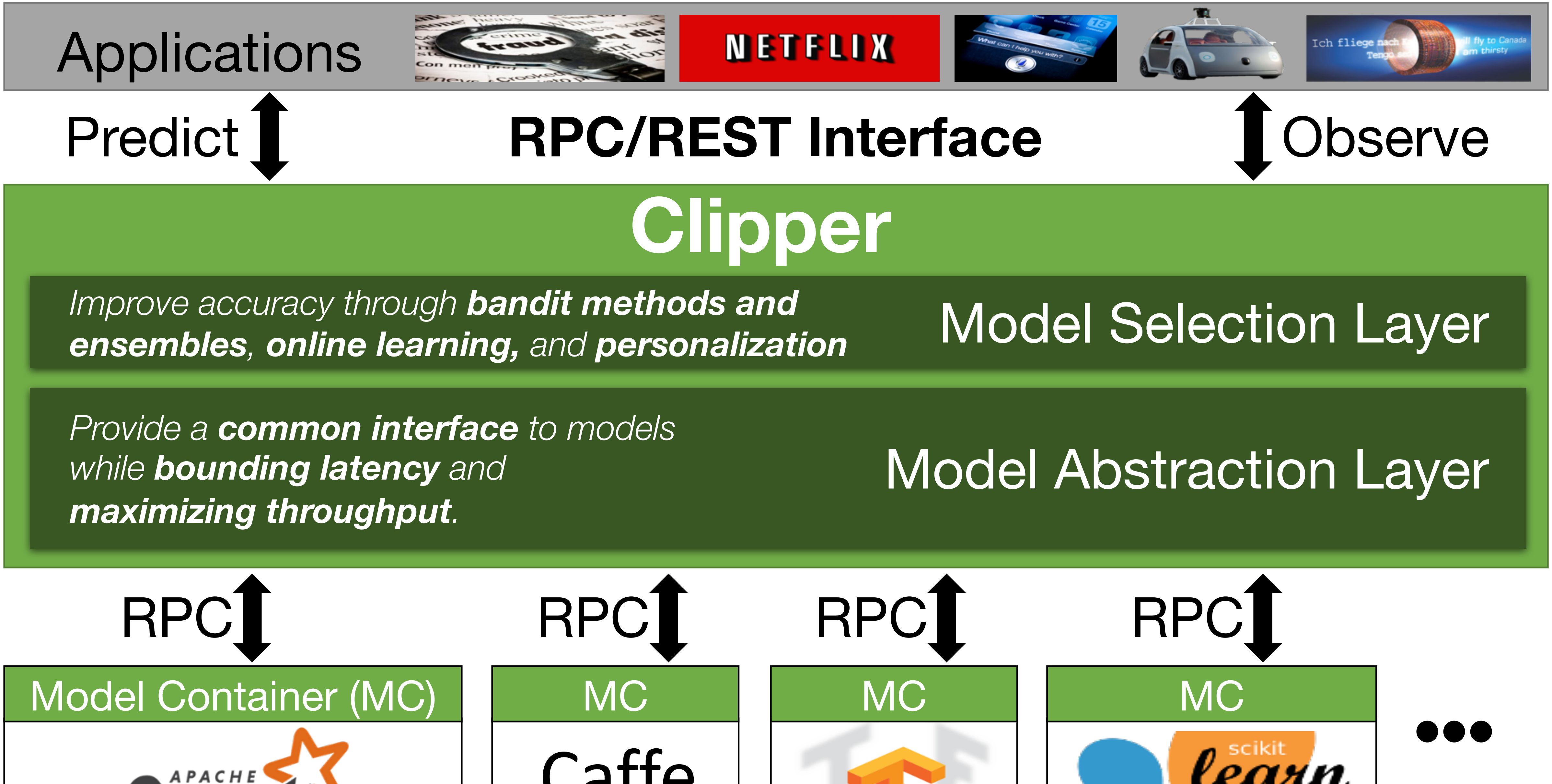
Clipper Solutions

- ❑ *Simplifies deployment through layered architecture*
- ❑ *Serves many models across ML frameworks concurrently*
- ❑ *Employs caching, batching, scale-out for high-performance serving*

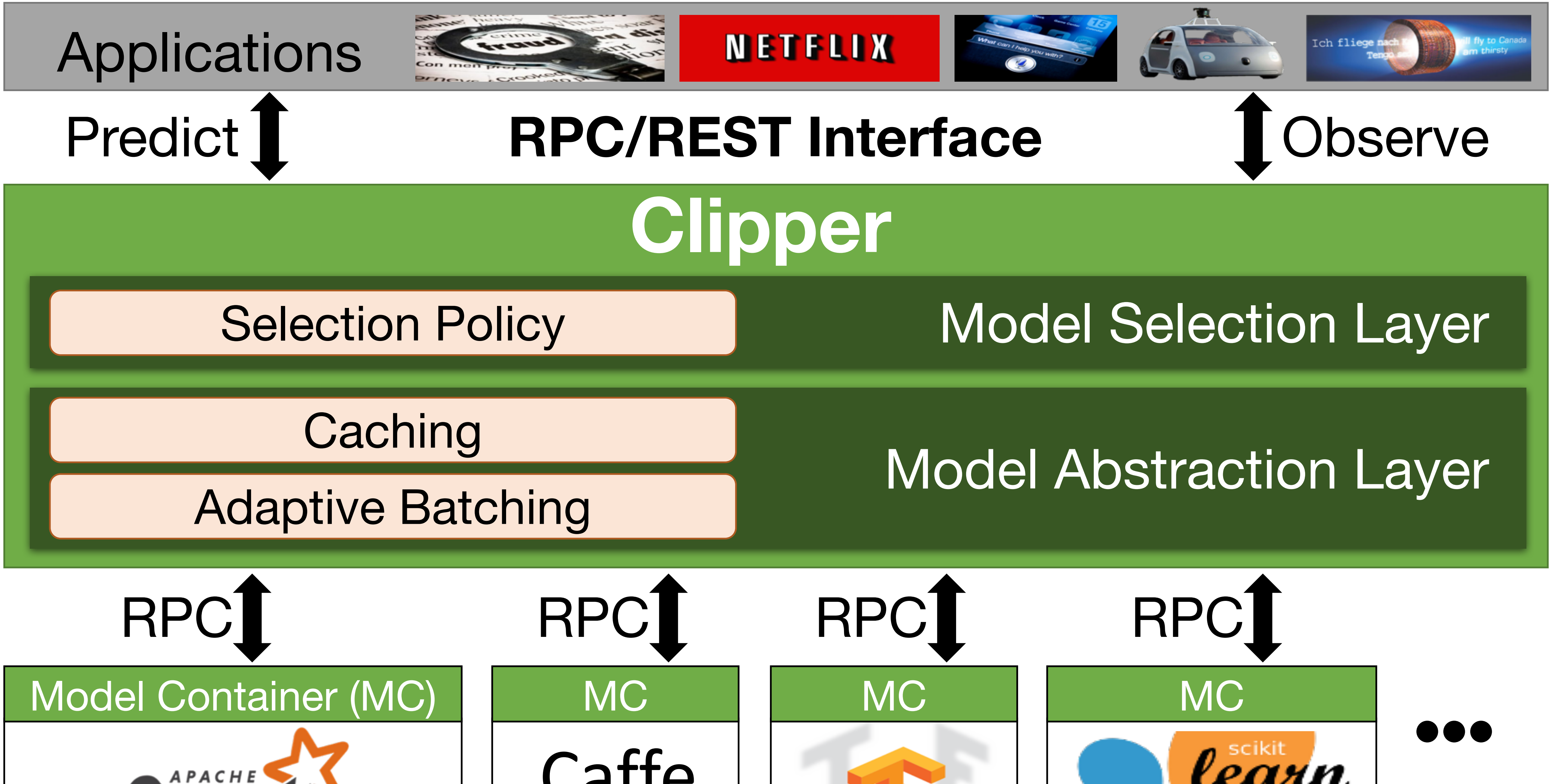
Clipper Decouples Applications and Models



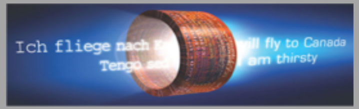
Clipper Architecture



Clipper Architecture



Clipper Implementation

Applications  NETFLIX   

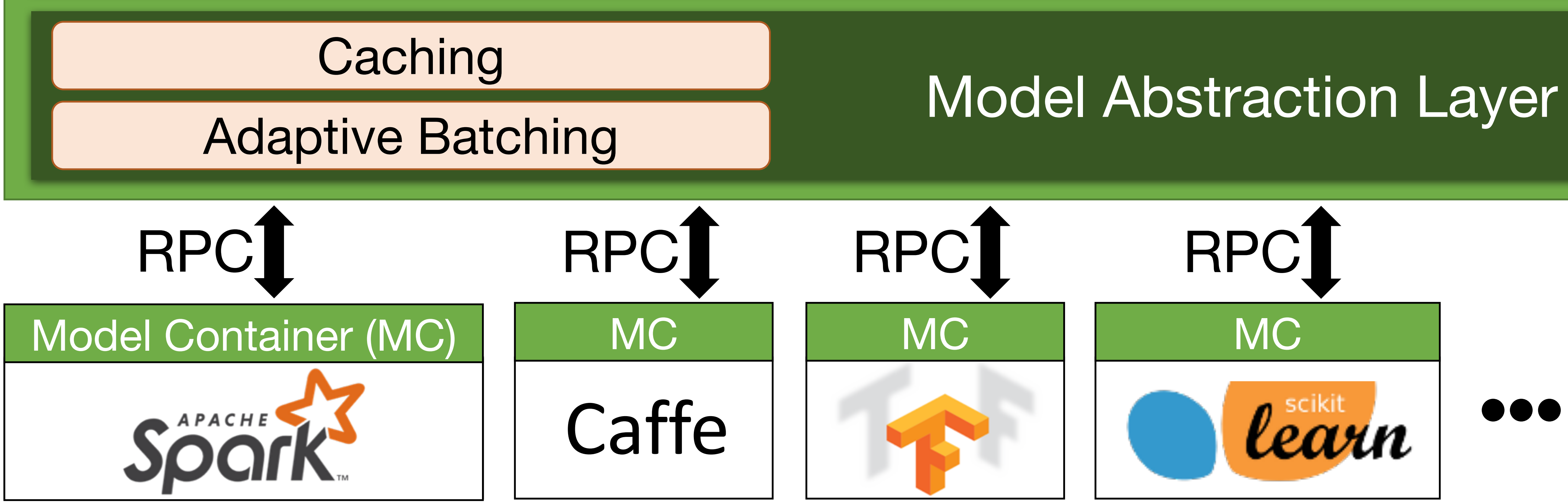
Predict ↕ **RPC/REST Interface** ↕ Observe

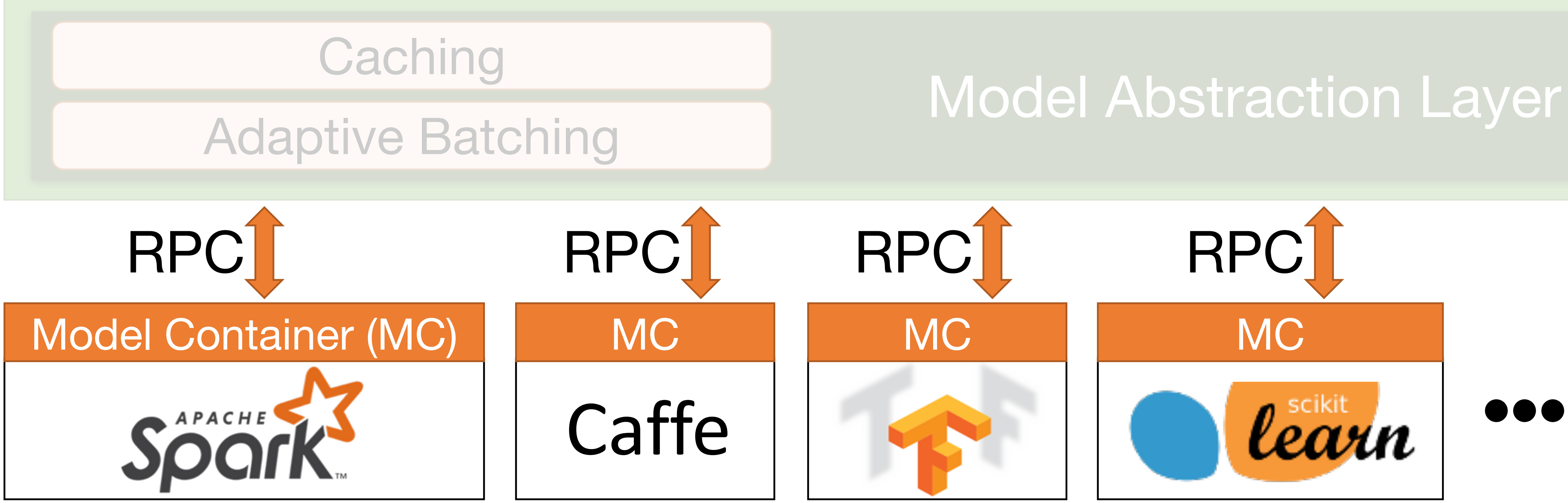
Clipper

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
RPC
RPC
RPC
RPC
Model Container (MC)
MC
MC
MC
APACHE
Spark
Caffe
scikit
learn



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++

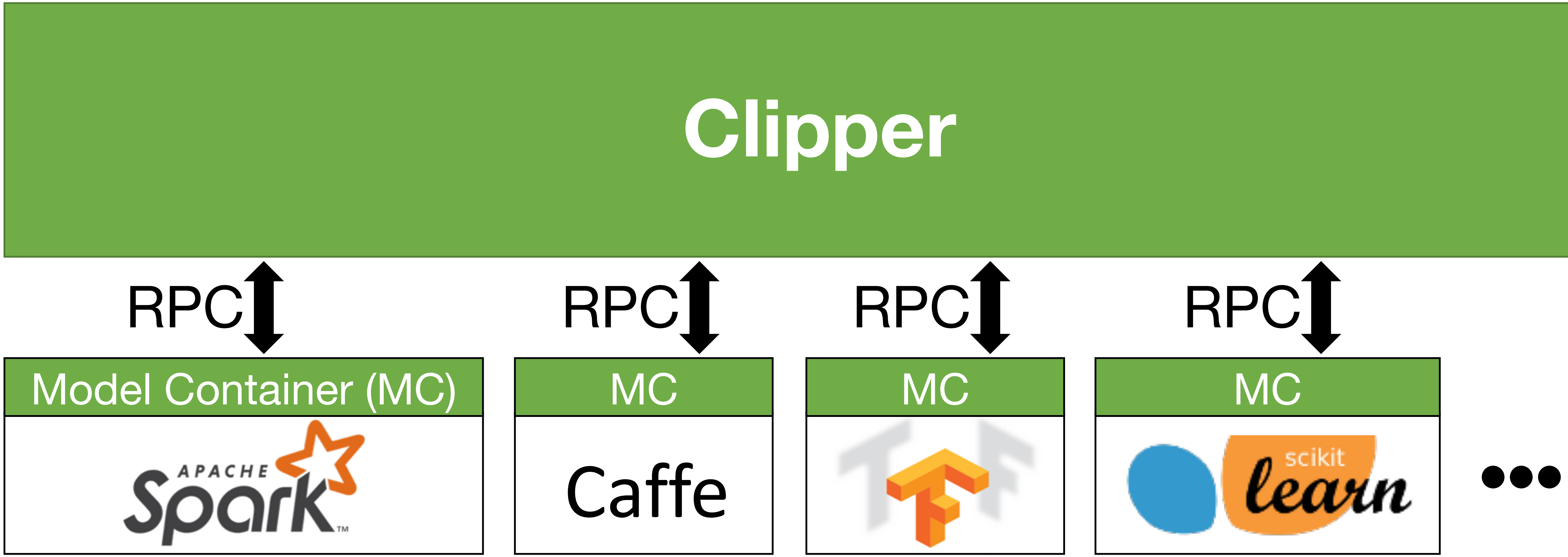
Container-based Model Deployment

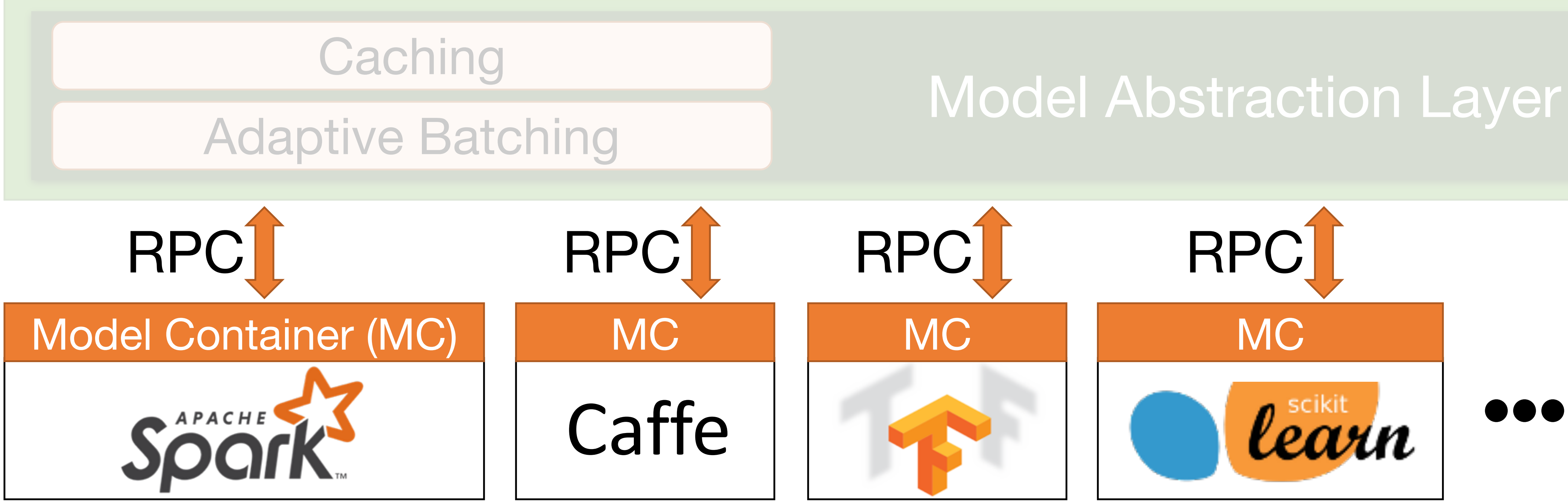
Model implementation packaged in container

Model Container (MC)

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

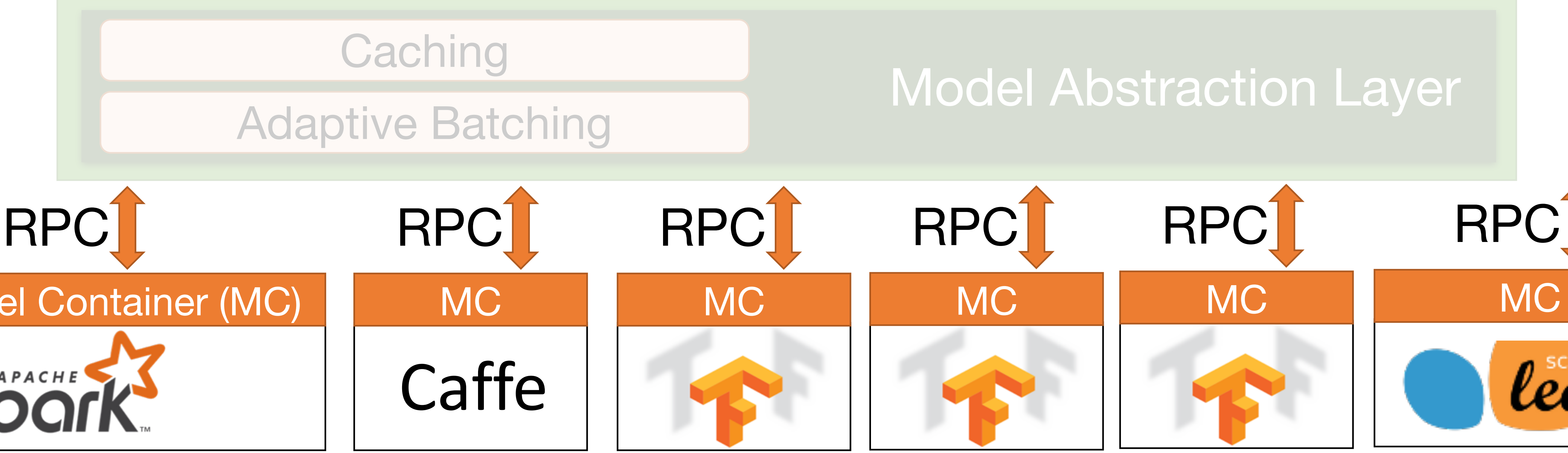
Container-based Model Deployment





Common Interface → Simplifies Deployment:

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



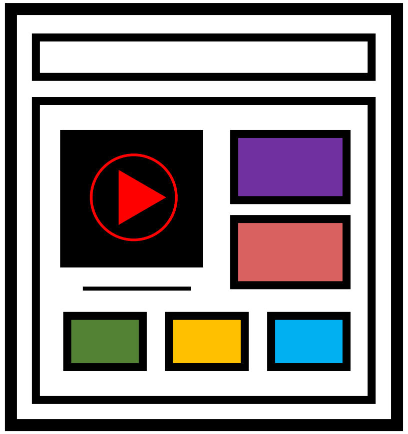
Common Interface → Simplifies Deployment:

- 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

Hardware Acceleration

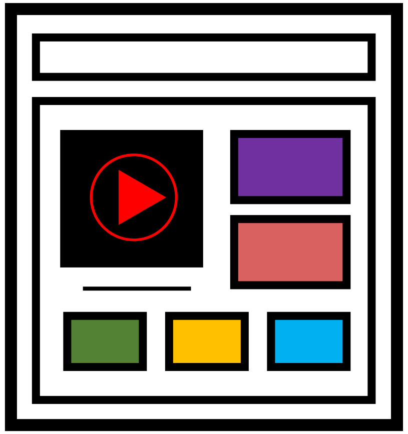
GRPC

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

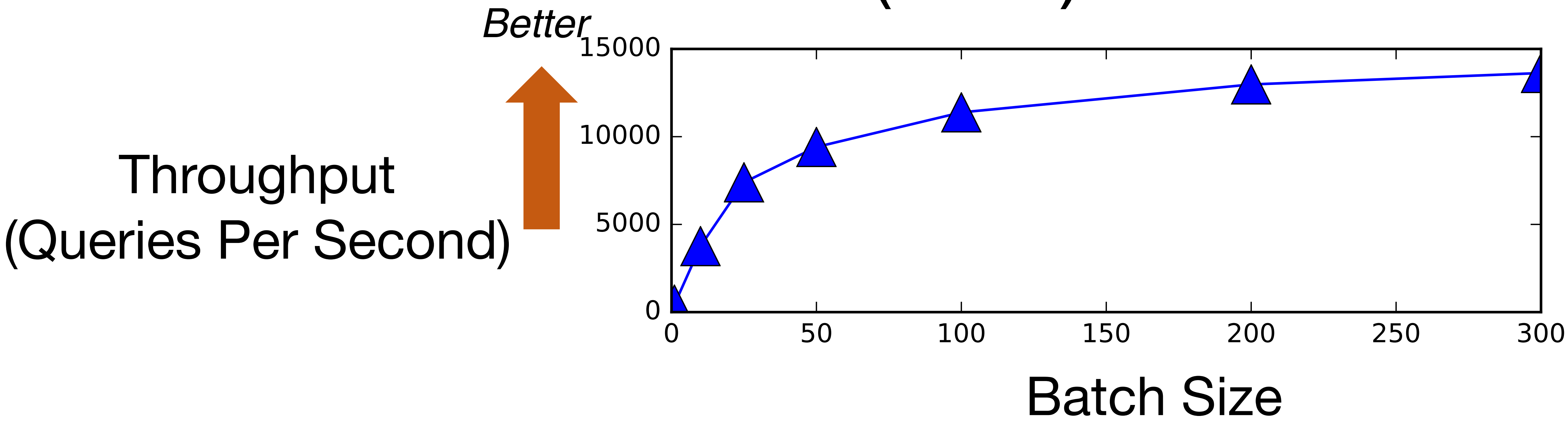
- Optimal batch depends on:
  - hardware configuration
  - model and framework
  - system load

Clipper Solution:

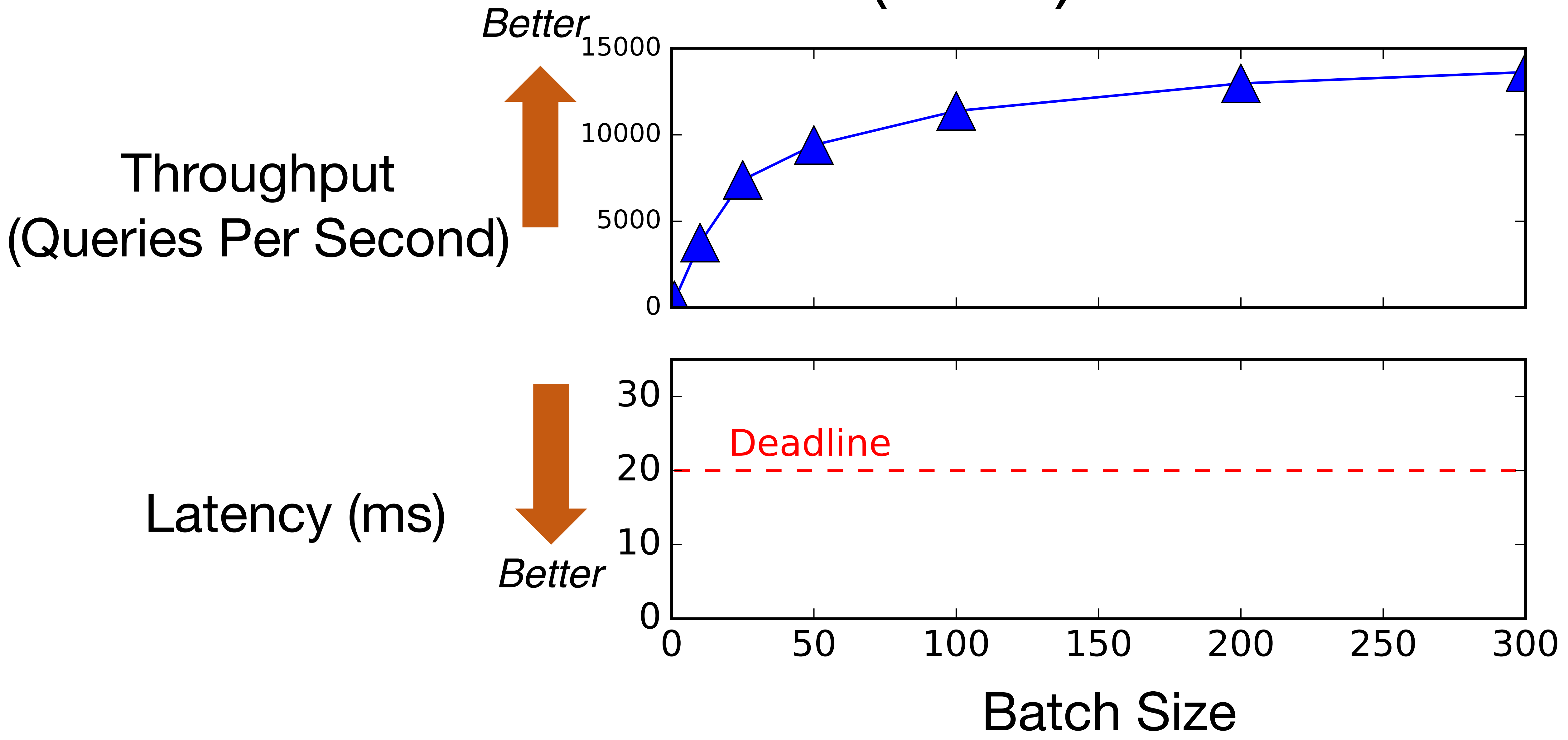
Adaptively tradeoff latency and throughput…

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

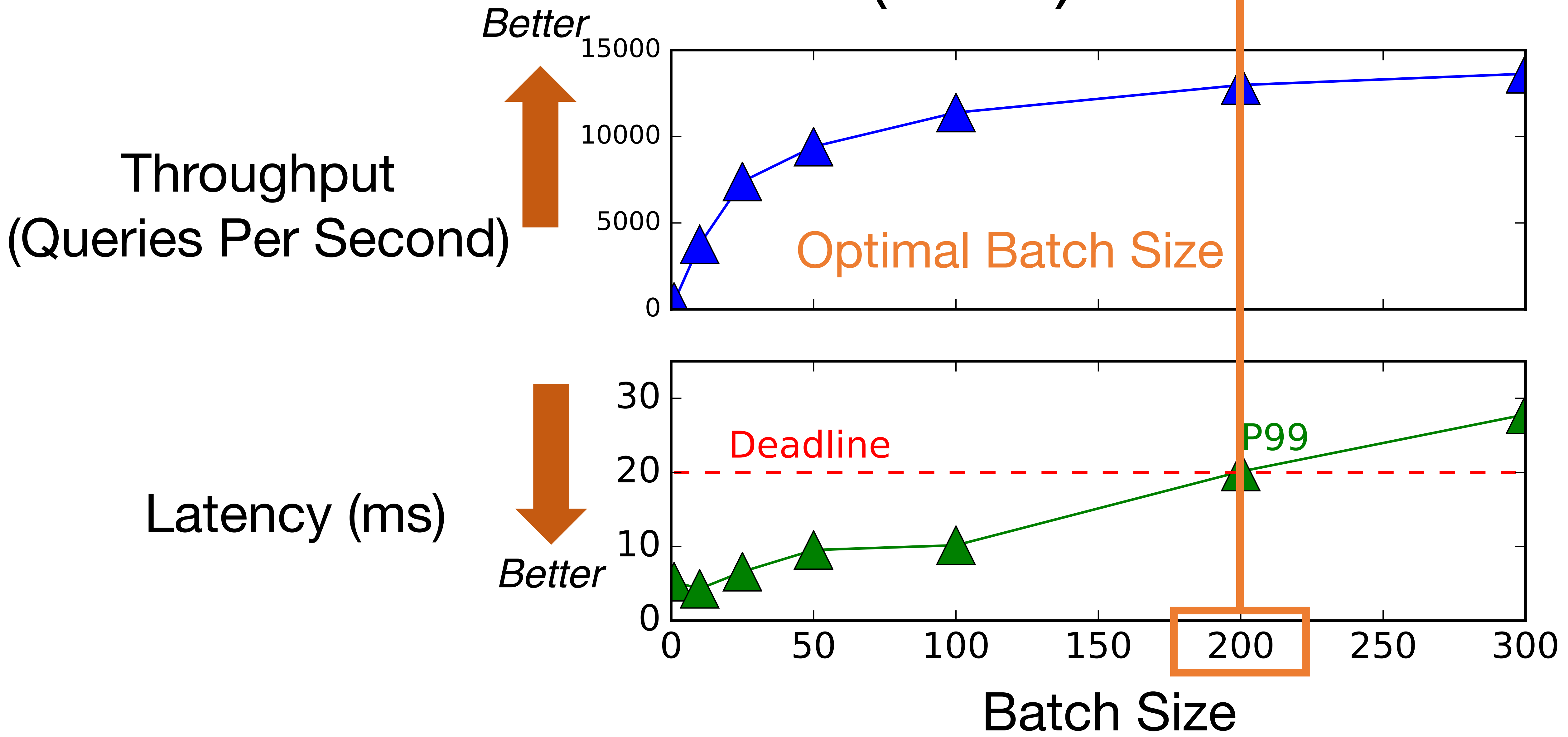
Tensor Flow Conv. Net (GPU)

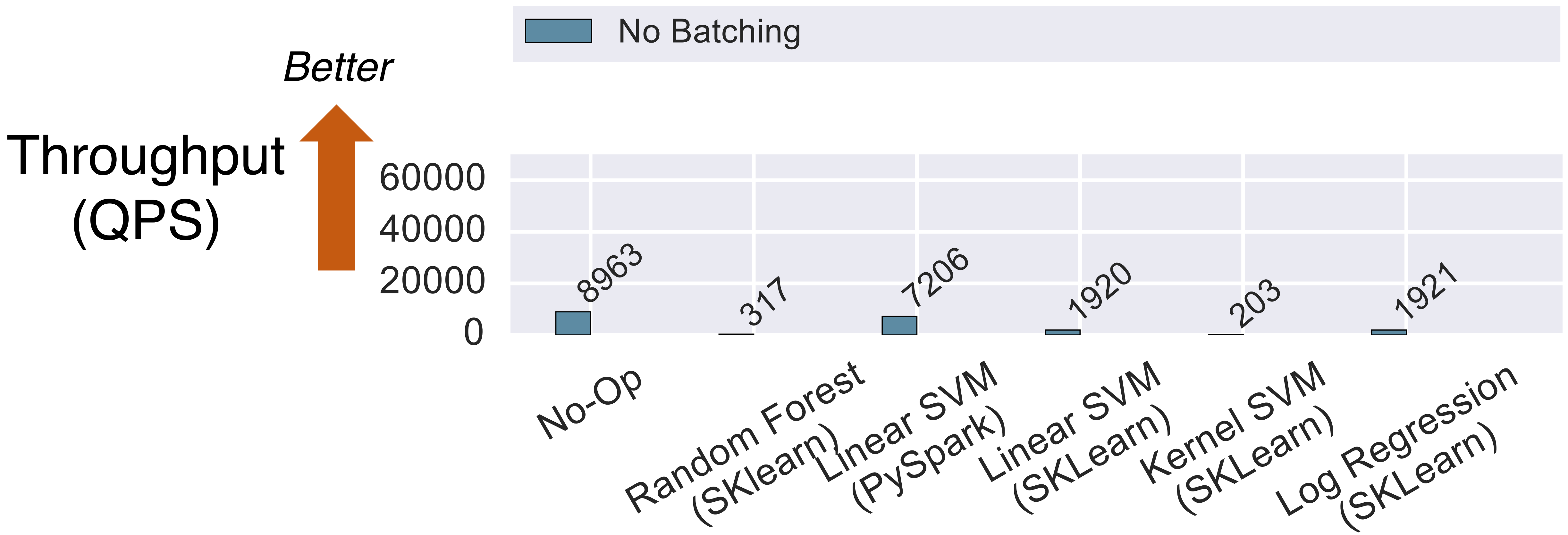


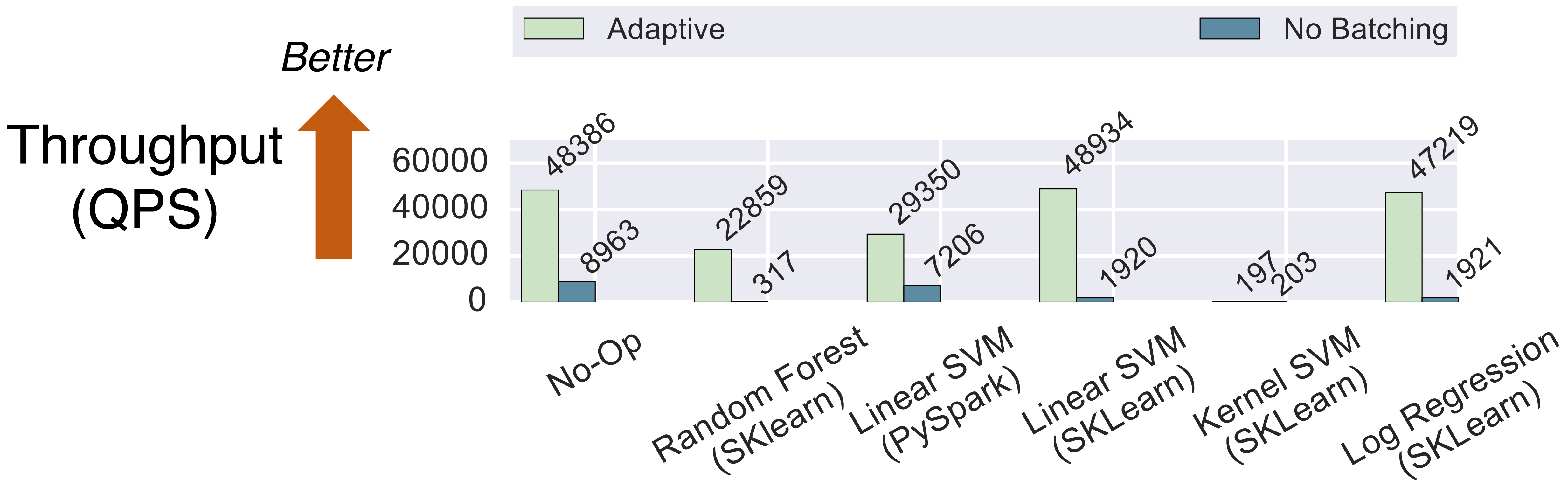
Tensor Flow Conv. Net (GPU)

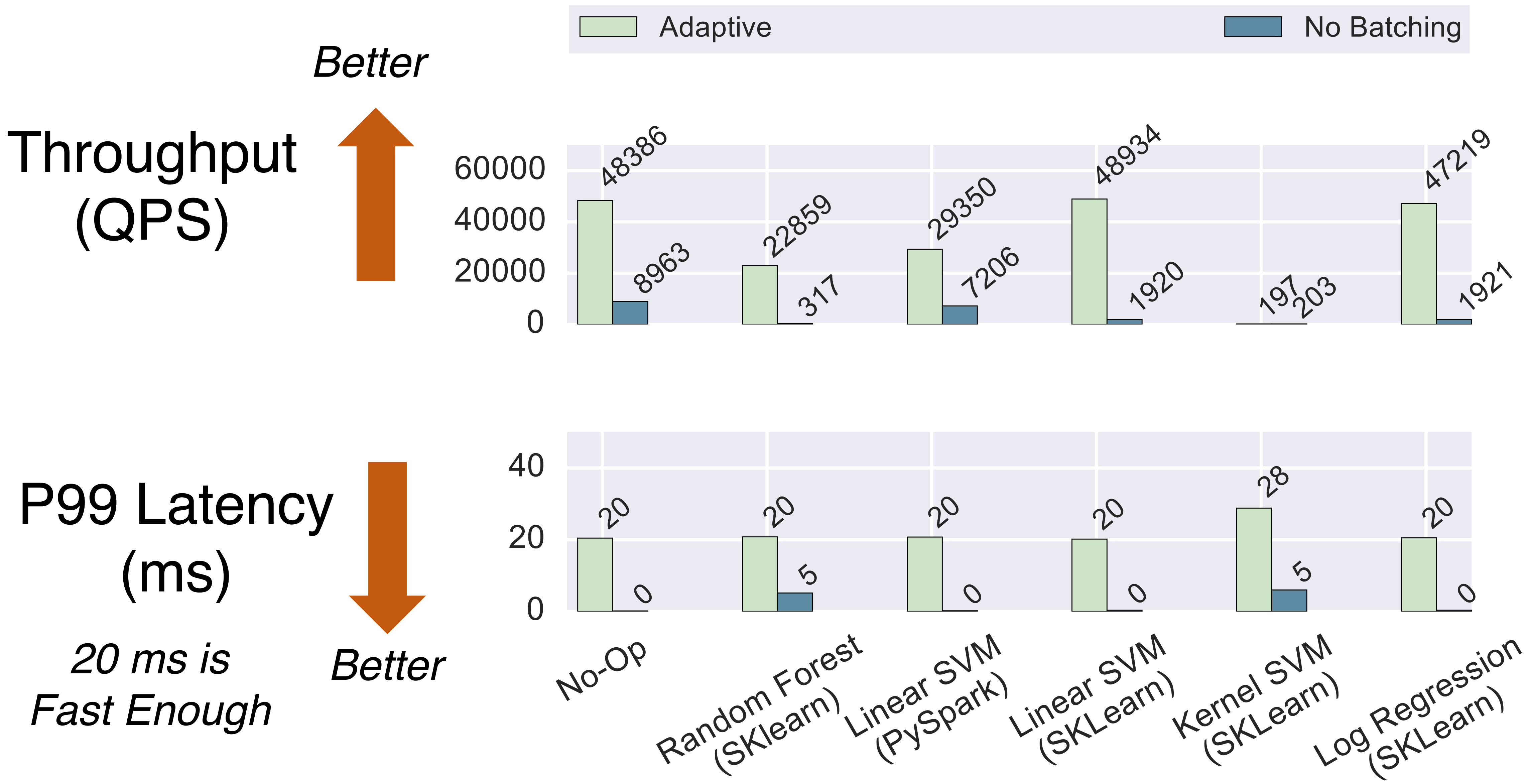


Tensor Flow Conv. Net (GPU)

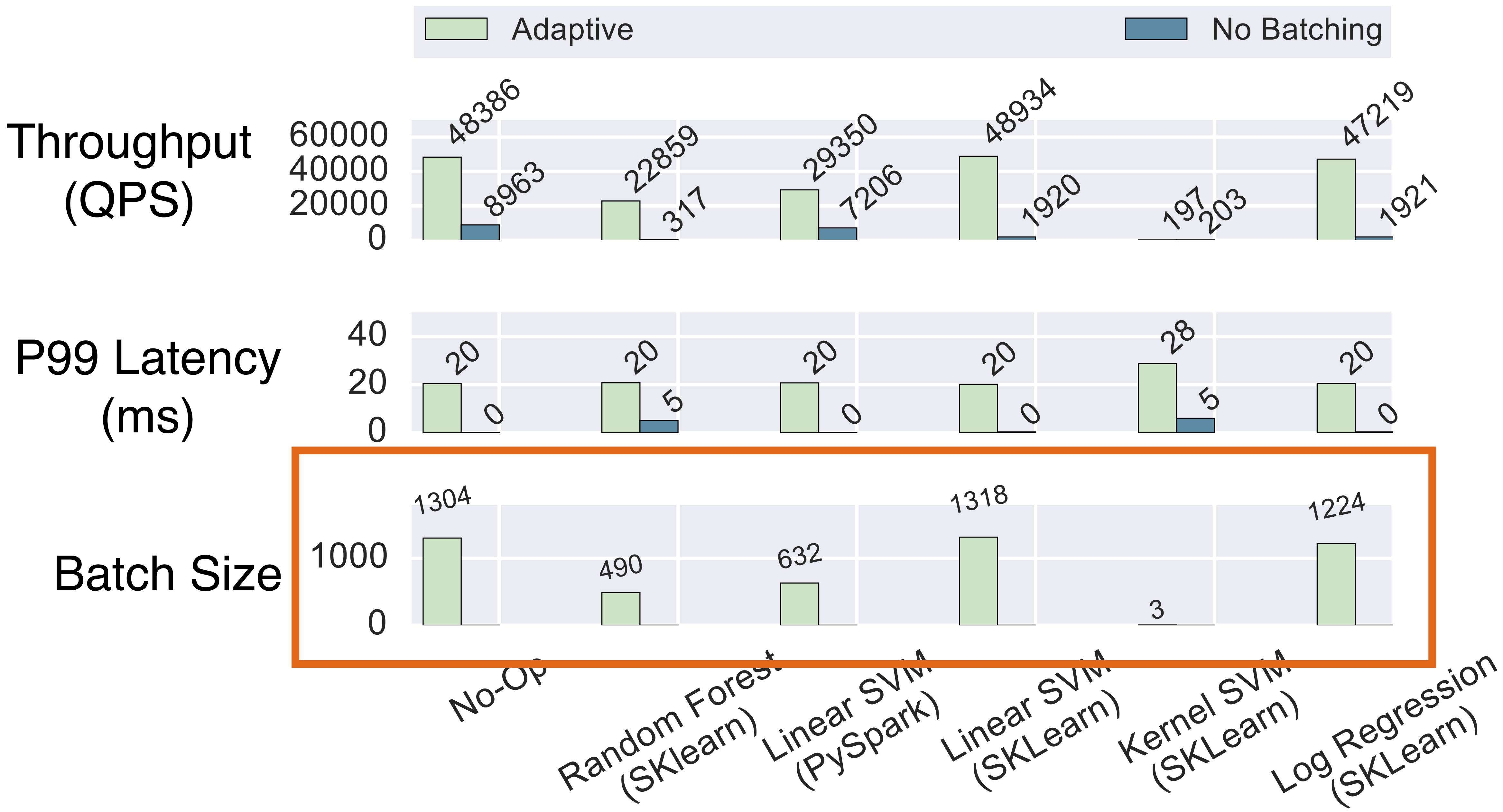



No Batching
Better
Throughput (QPS)
60000
40000
20000
0
8963
317
7206
1920
203
1921
No-Op
Random Forest (SKlearn)
Linear SVM (PySpark)
Linear SVM (SKLearn)
Kernel SVM (SKLearn)
Log Regression (SKLearn)


Adaptive
No Batching
Better
Throughput
(QPS)
60000
40000
20000
0
48386
8963
22859
317
29350
7206
48934
1920
197
203
47219
1921
No-Op
Random Forest
(SKlearn)
Linear SVM
(PySpark)
Linear SVM
(SKLearn)
Kernel SVM
(SKLearn)
Log Regression
(SKLearn)

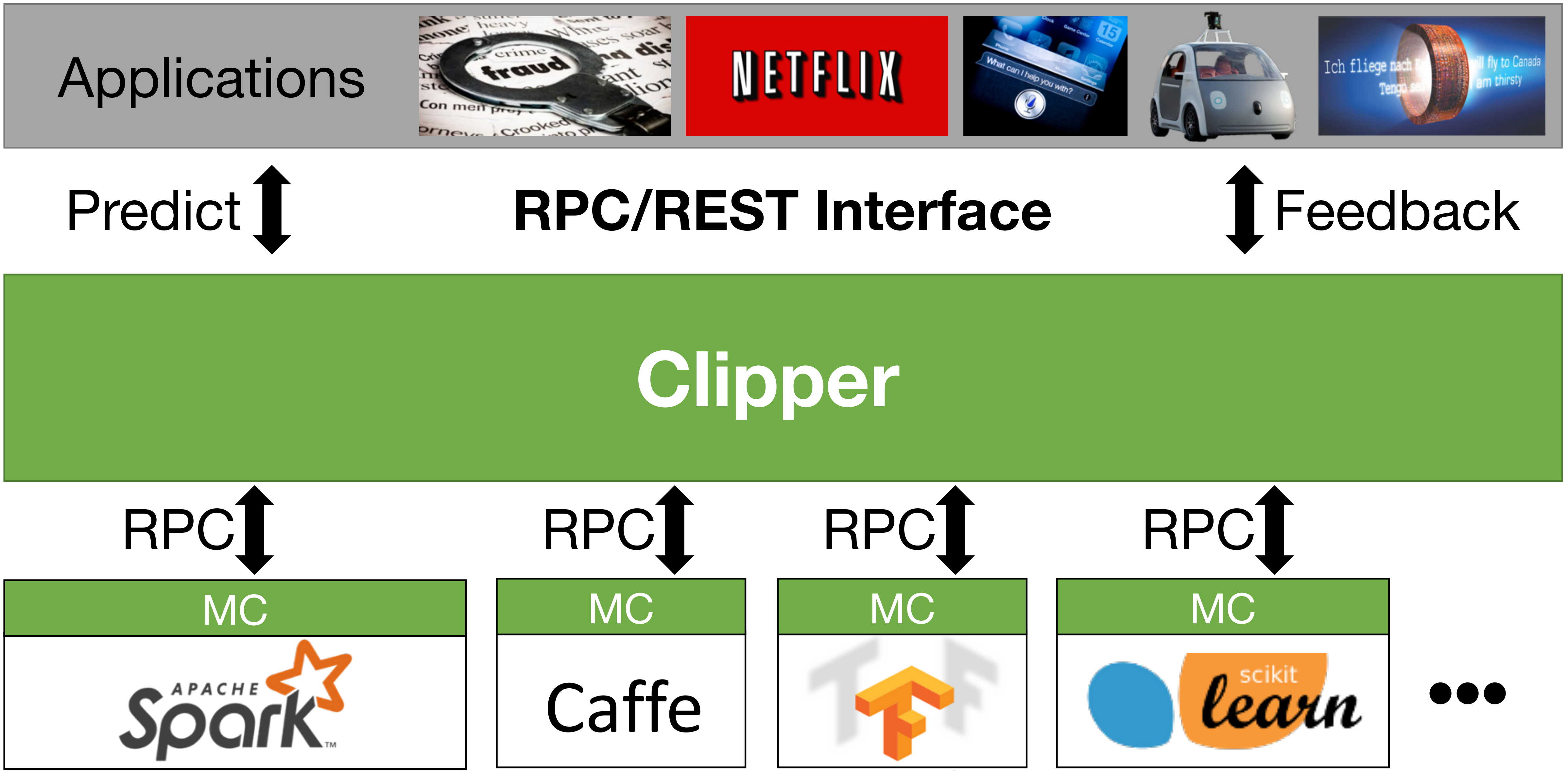


Adaptive
No Batching
Better
Throughput (QPS)
60000
40000
20000
0
48386
8963
22859
317
29350
7206
48934
1920
197
203
47219
1921
P99 Latency (ms)
40
20
0
20
0
20
5
20
0
20
0
28
5
20
0
20 ms is Fast Enough
Better
No-Op
Random Forest (SKlearn)
Linear SVM (PySpark)
Linear SVM (SKLearn)
Kernel SVM (SKLearn)
Log Regression (SKLearn)

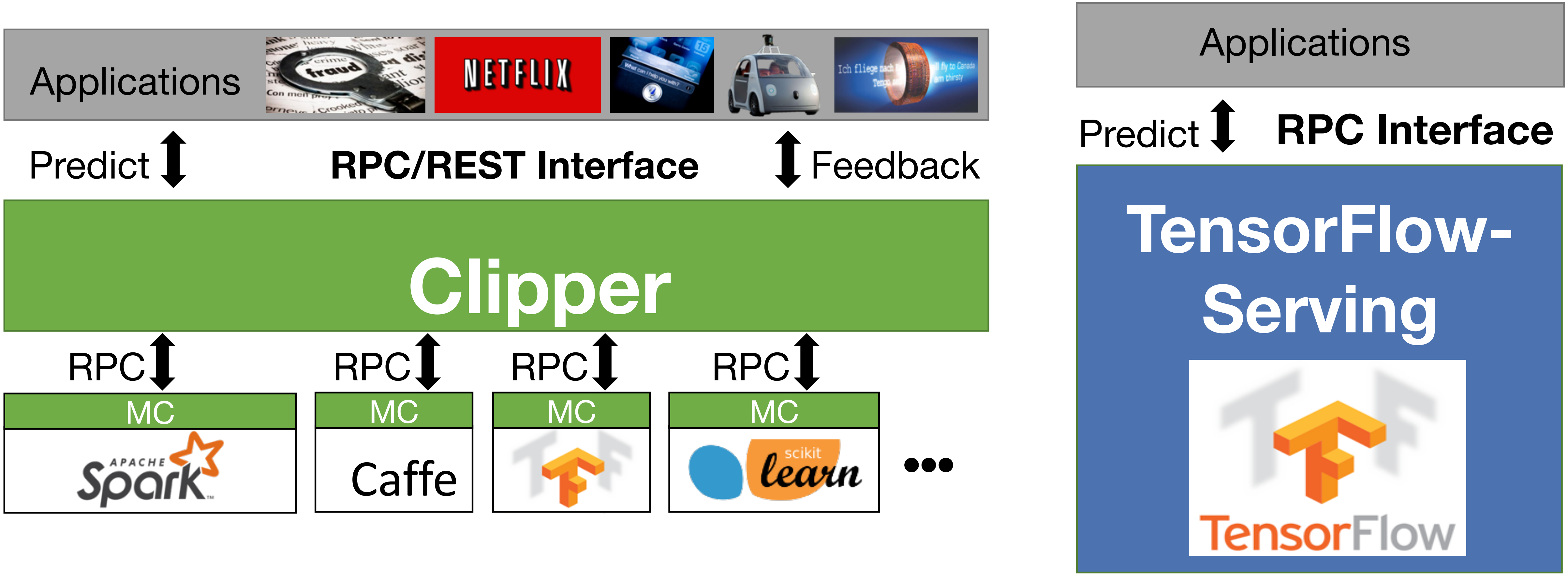


Adaptive
No Batching
Throughput (QPS)
60000
40000
20000
0
48386
8963
22859
317
29350
7206
48934
1920
197
203
47219
1921
P99 Latency (ms)
40
20
0
20
0
20
5
20
0
20
0
28
5
20
0
Batch Size
1000
0
1304
490
632
1318
3
1224
No-Op
Random Forest (SKlearn)
Linear SVM (PySpark)
Linear SVM (SKLearn)
Kernel SVM (SKLearn)
Log Regression (SKLearn)

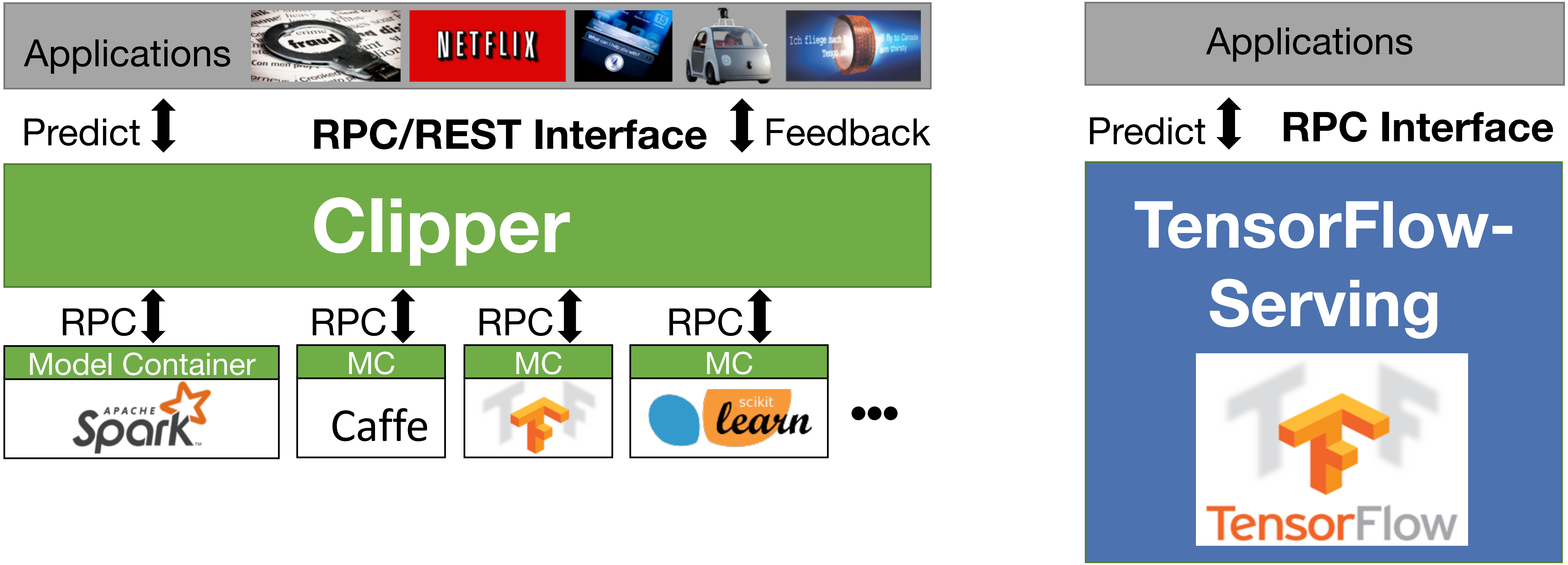
Overhead of decoupled architecture



Overhead of decoupled architecture

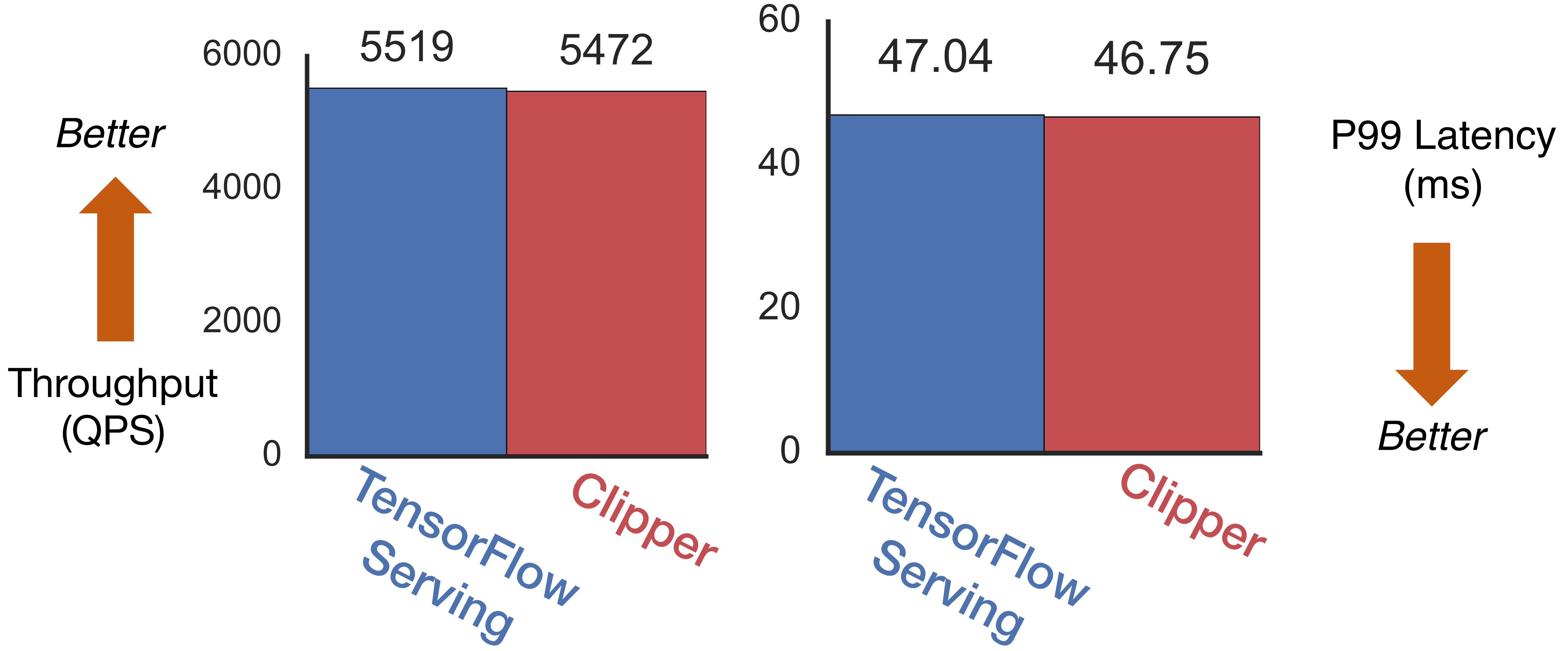


Overhead of decoupled architecture



Overhead of decoupled architecture

Model: AlexNet trained on CIFAR-10



Clipper Architecture

Applications

Predict

RPC/REST Interface

Observe

Clipper

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

Model Selection Layer

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

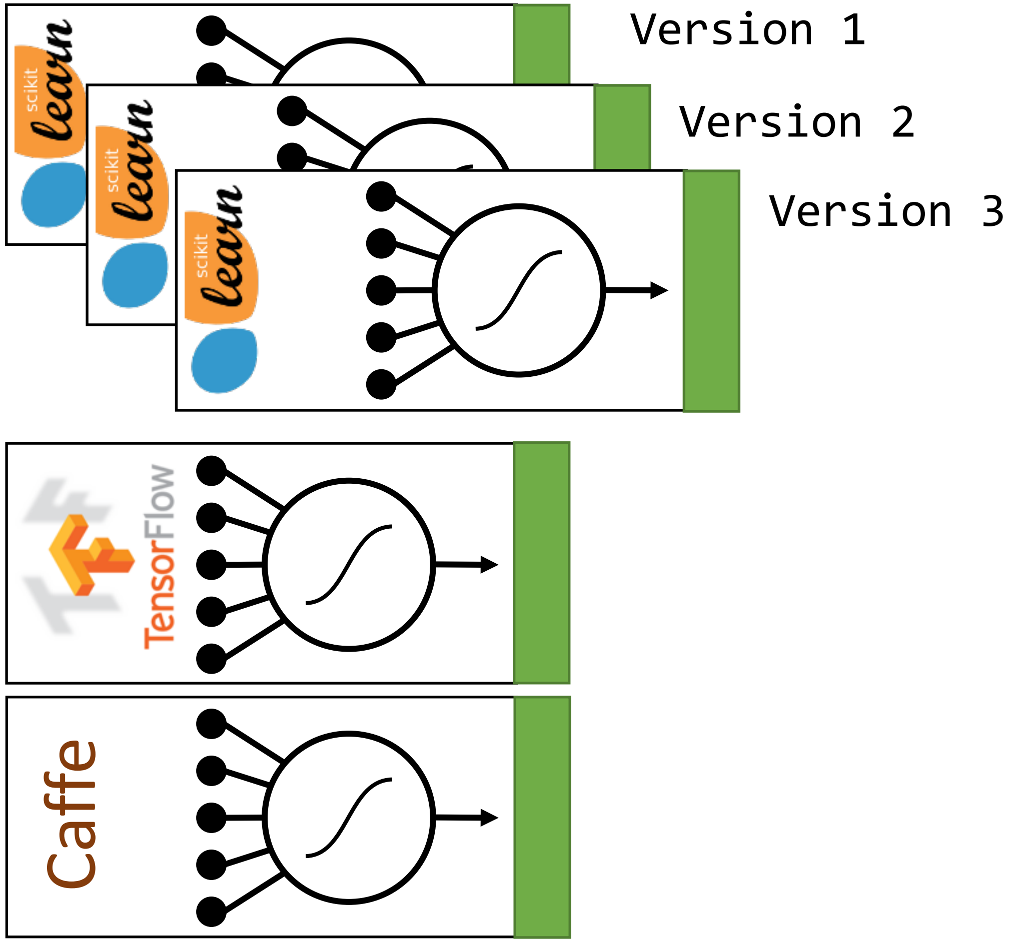
Model Abstraction Layer

RPC RPC RPC RPC

Model Container (MC) | MC | MC | MC | •••

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

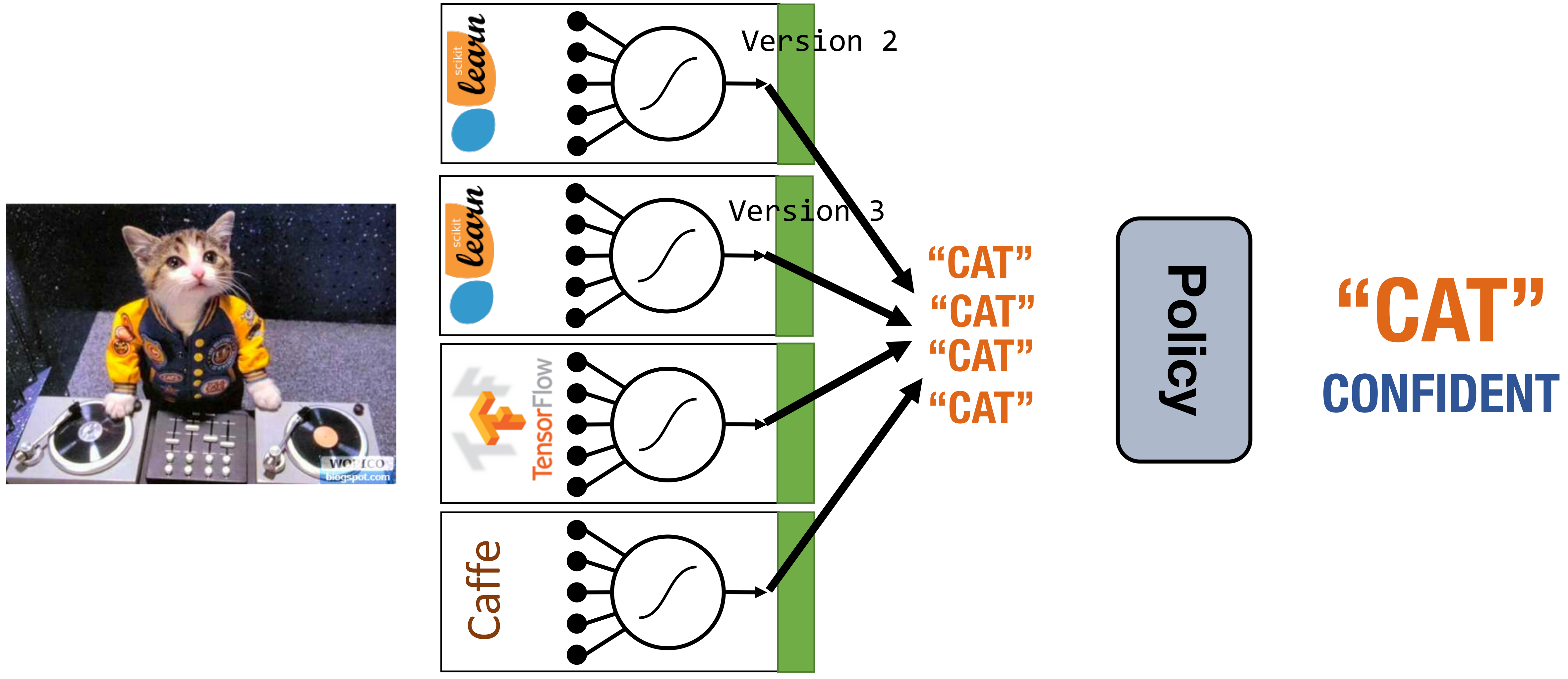
Model Selection Layer



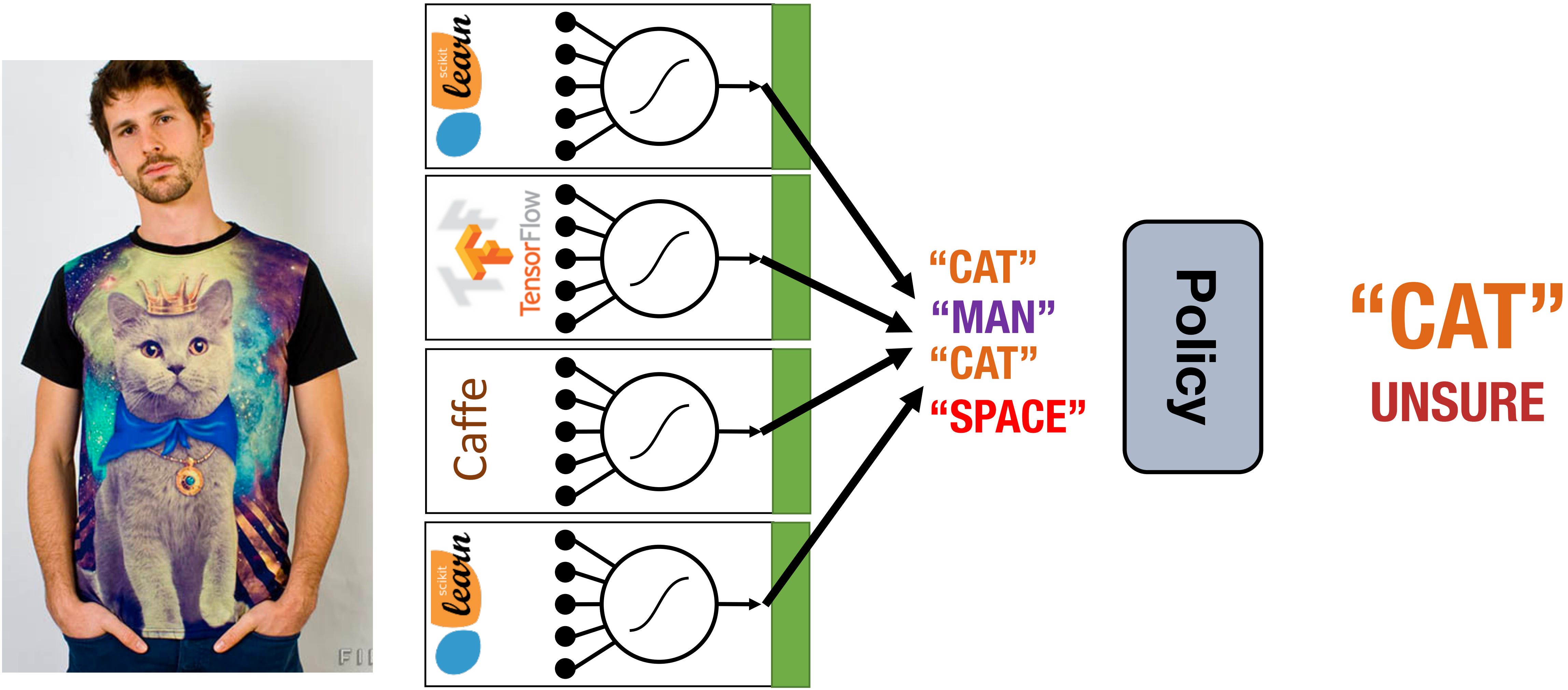
Periodic retraining

Experiment with new models and frameworks

Selection Policy: Estimate confidence

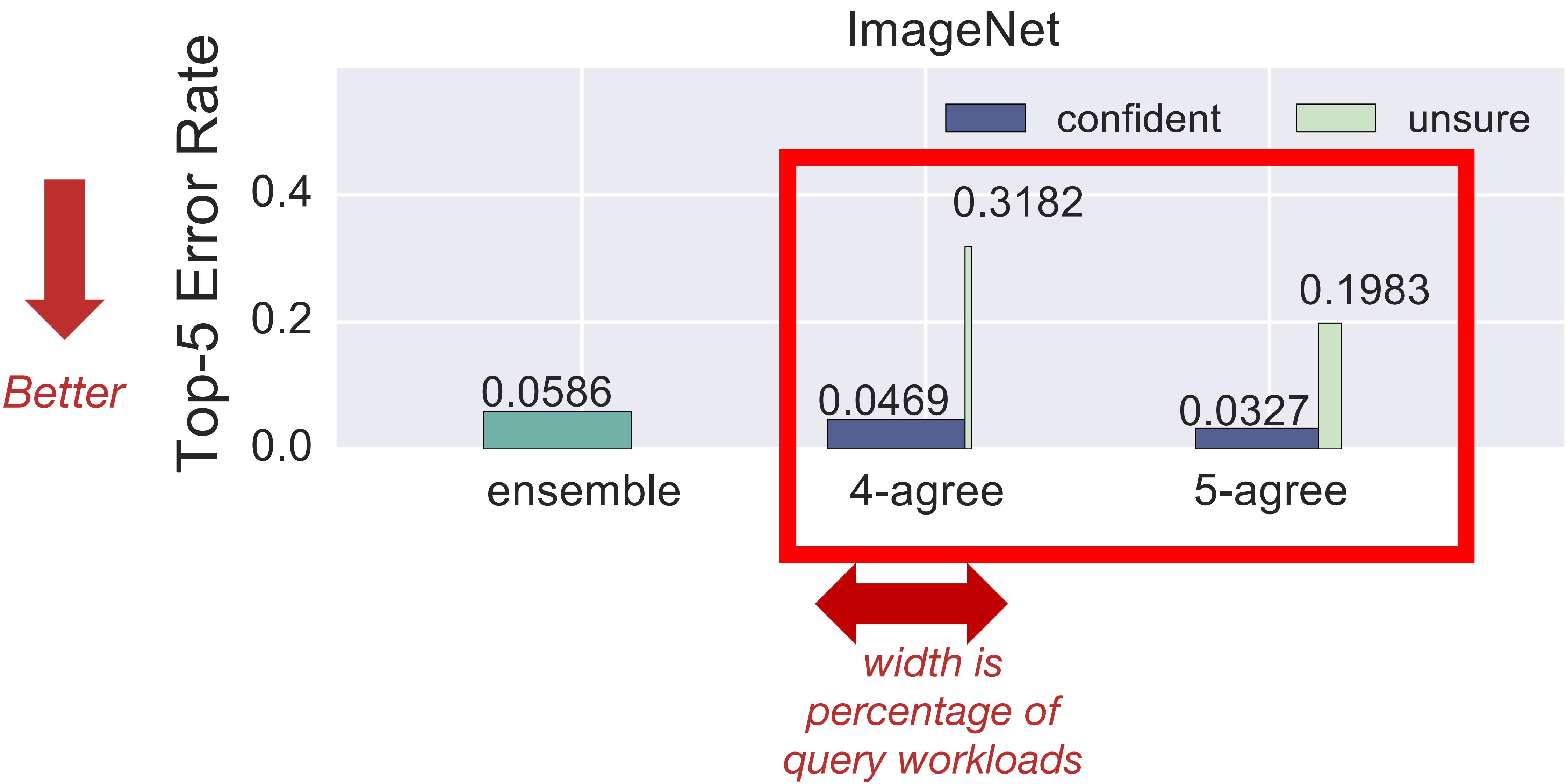


Selection Policy: Estimate confidence



Selection Policy: Estimate confidence

Selection Policy: Estimate confidence



Selection Policy

Model Selection Layer

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