RISE to the Challenges of AI Systems

Joseph E. Gonzalez
Assistant Professor, UC Berkeley
jegonzal@cs.berkeley.edu

Large-Scale parallel and distributed systems

GraphLab

GraphX

DMLC TensorFlow Caffe VW GraphLab scikit learn

Apache Spark MLbase KeystoneML GraphX Splash CoCoA

How to do Research in AI Systems

- Manage **Complexity**
  - seek parsimony in system design
  - great systems research is often about what features are taken away
  - Do a few things well and be composable

- Identify **Tradeoffs**
  - With each design decision what do you gain and lose?
  - What trade-offs are fundamental?

- **Evaluate** your System
  - **Positive:** How fast and scalable is it and *why*?
  - **Negative:** When does it fail and what are it's *limitations*?

Hemingway*

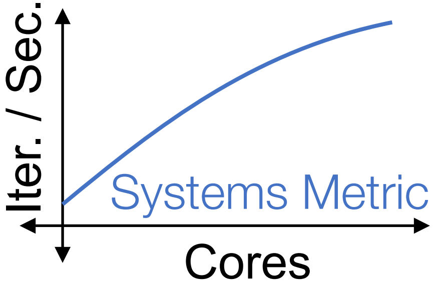
Modeling Throughput and Convergence for ML Workloads

Shivaram Venkataraman

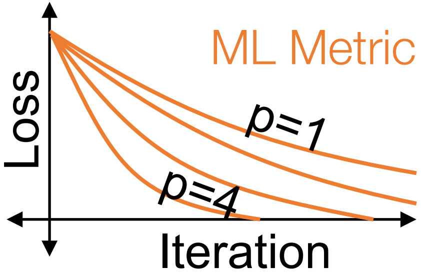
Xinghao Pan

Zi Zheng

- What is the best algorithm and level of parallelism for an ML task?
  - **Trade-off:** Parallelism, Coordination, & Convergence
- **Research challenge:** Can we model this trade-off explicitly?



$I(p)$ Iterations per second as a function of cores ***p***



$L(i, p)$ Loss as a function of iterations ***i*** and cores ***p***

We can estimate I from data on many systems

We can estimate L from data for our problem

*follow-up work to Shivaram's Ernest System in NSDI'16

Hemingway*

Modeling Throughput and Convergence for ML Workloads

Shivaram Venkataraman

Xinghao Pan

Zi Zheng

- What is the best algorithm and level of parallelism for an ML task?
  - **Trade-off:** Parallelism, Coordination, & Convergence
- **Research challenge:** Can we model this trade-off explicitly?

$L(i,p)$ Loss as a function of iterations ***i*** and cores ***p***

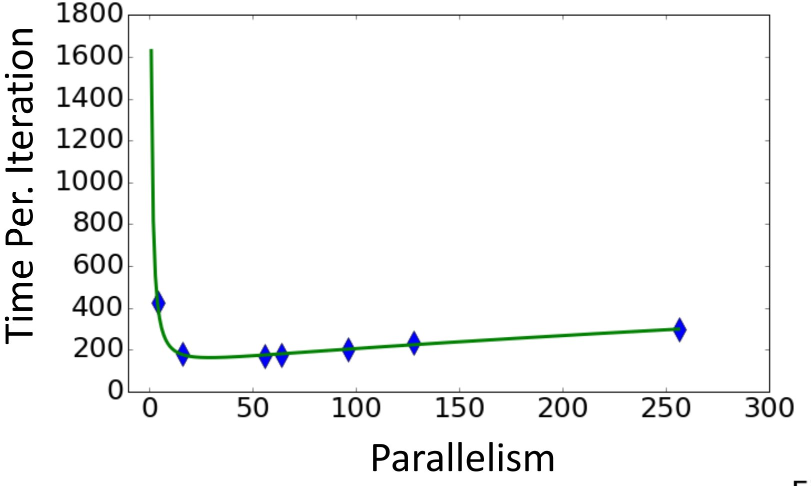
$I(p)$ Iterations per second as a function of cores ***p***

$$\mathbf{loss}(t,p) = L\left(t{*}I\left(p\right),p\right)$$

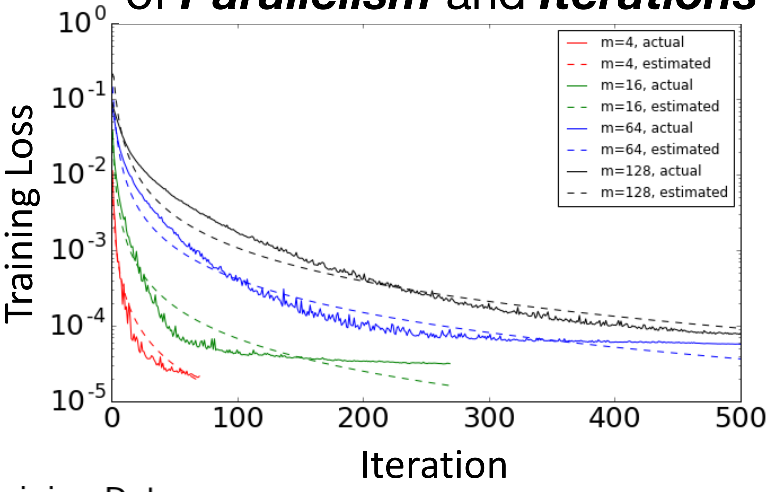
- How long does it take to get to a given loss?
- Given a time budget and number of cores which algorithm will give the best result?

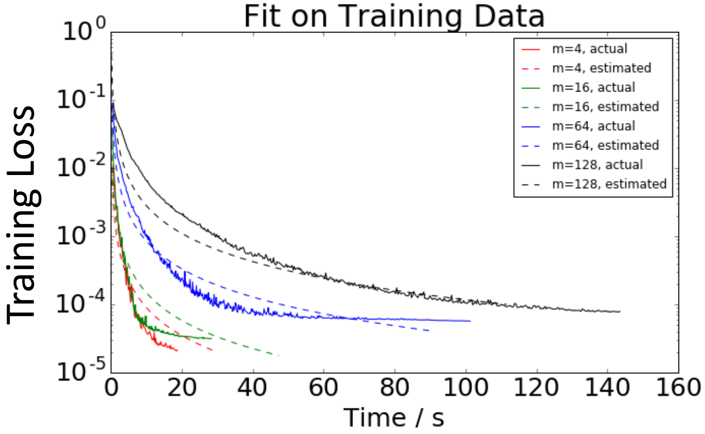
*follow-up work to Shivaram's Ernest System in NSDI'16

System Performance as a function of ***Parallelism***



Convergence as a function of ***Parallelism*** and ***Iterations***





Convergence as a fn. of ***Time*** and ***Parallelism***

Hemingway: Modeling Distributed Optimization Algorithms.
Xinghao Pan, Shivaram Venkataraman, Zizheng Tai, Joseph Gonzalez.
NIPS’16 ML-Sys Workshop.

Take away …

try to decouple

System Improvements

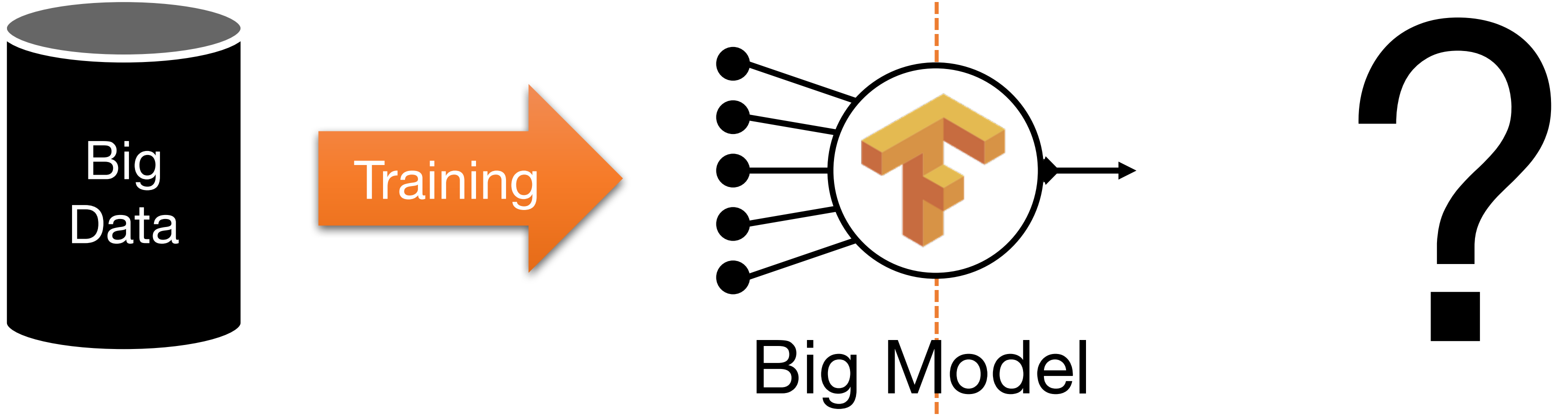
Algorithm Improvements

use data collection + sparse modeling
to understand your system

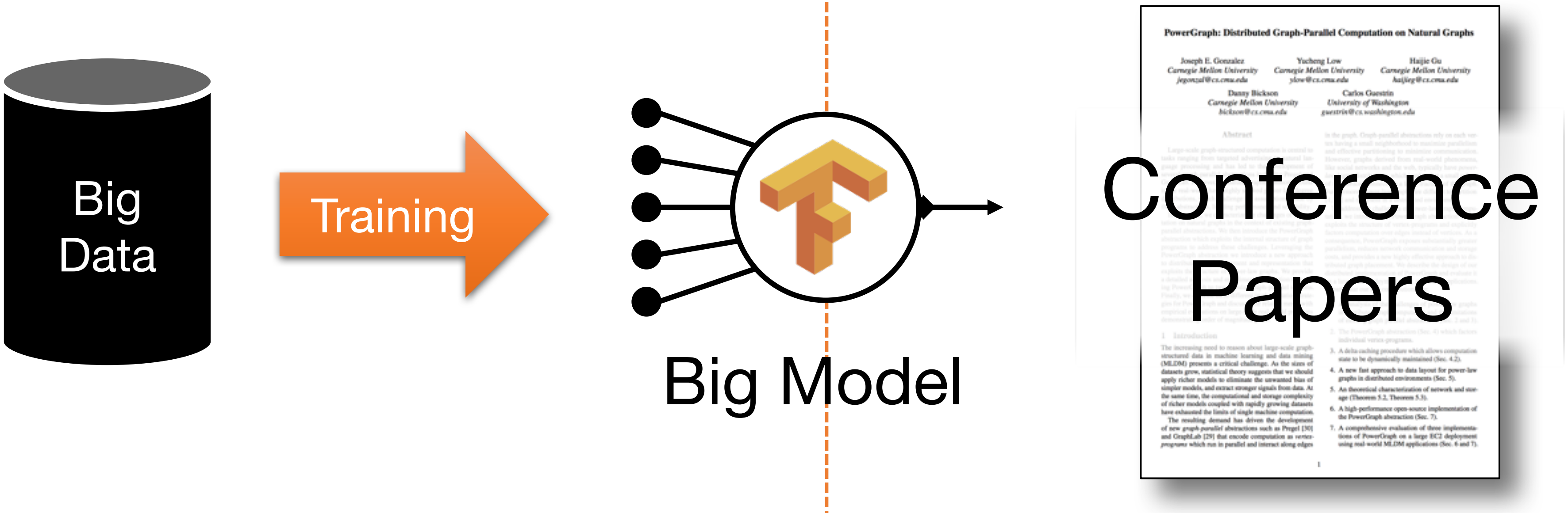
DMLC
TensorFlow
Caffe
VW
GraphLab
scikit learn
APACHE Spark
MLbase
KeystoneML
GraphX
Splash
CoCoA

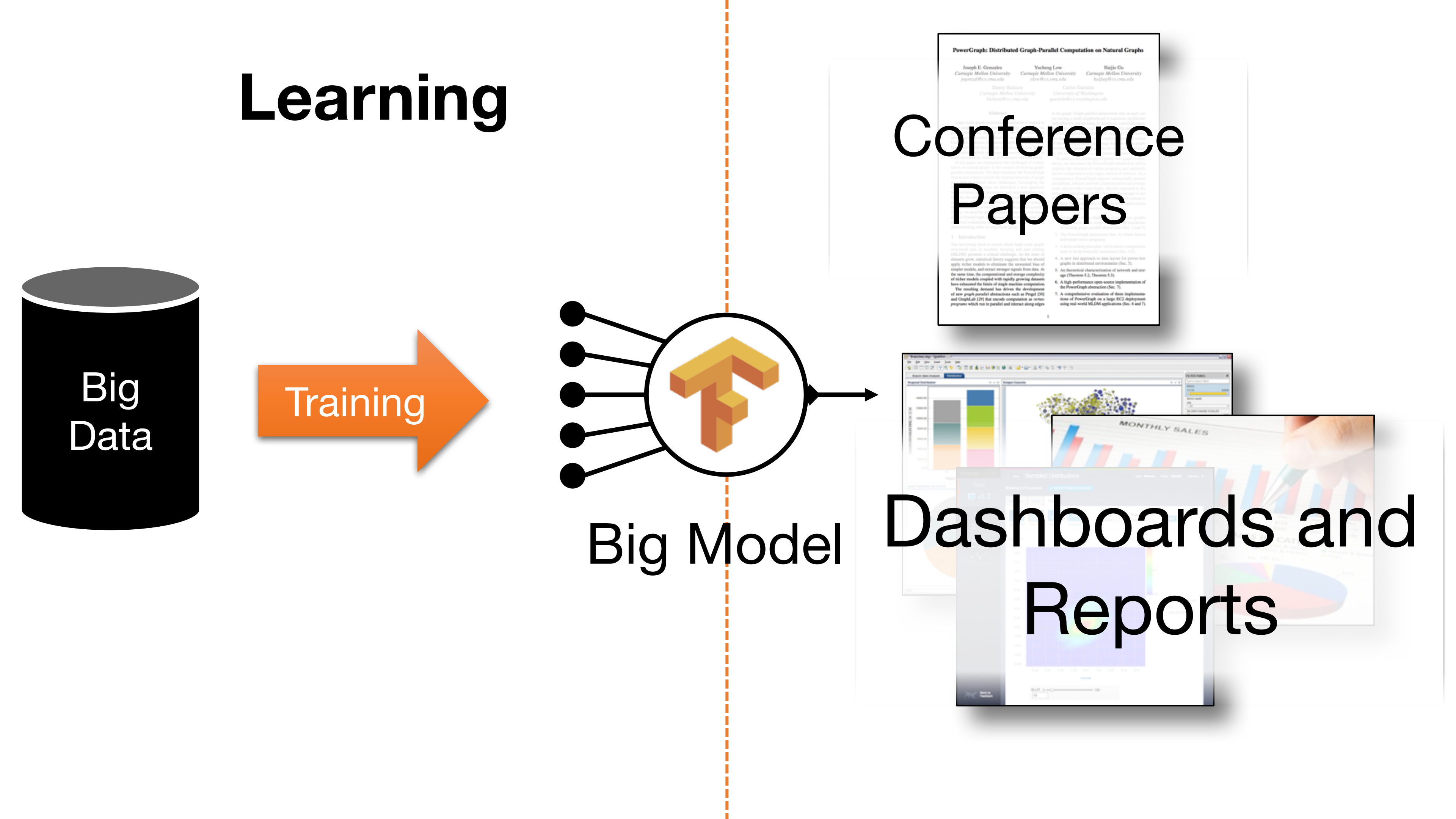
amplab

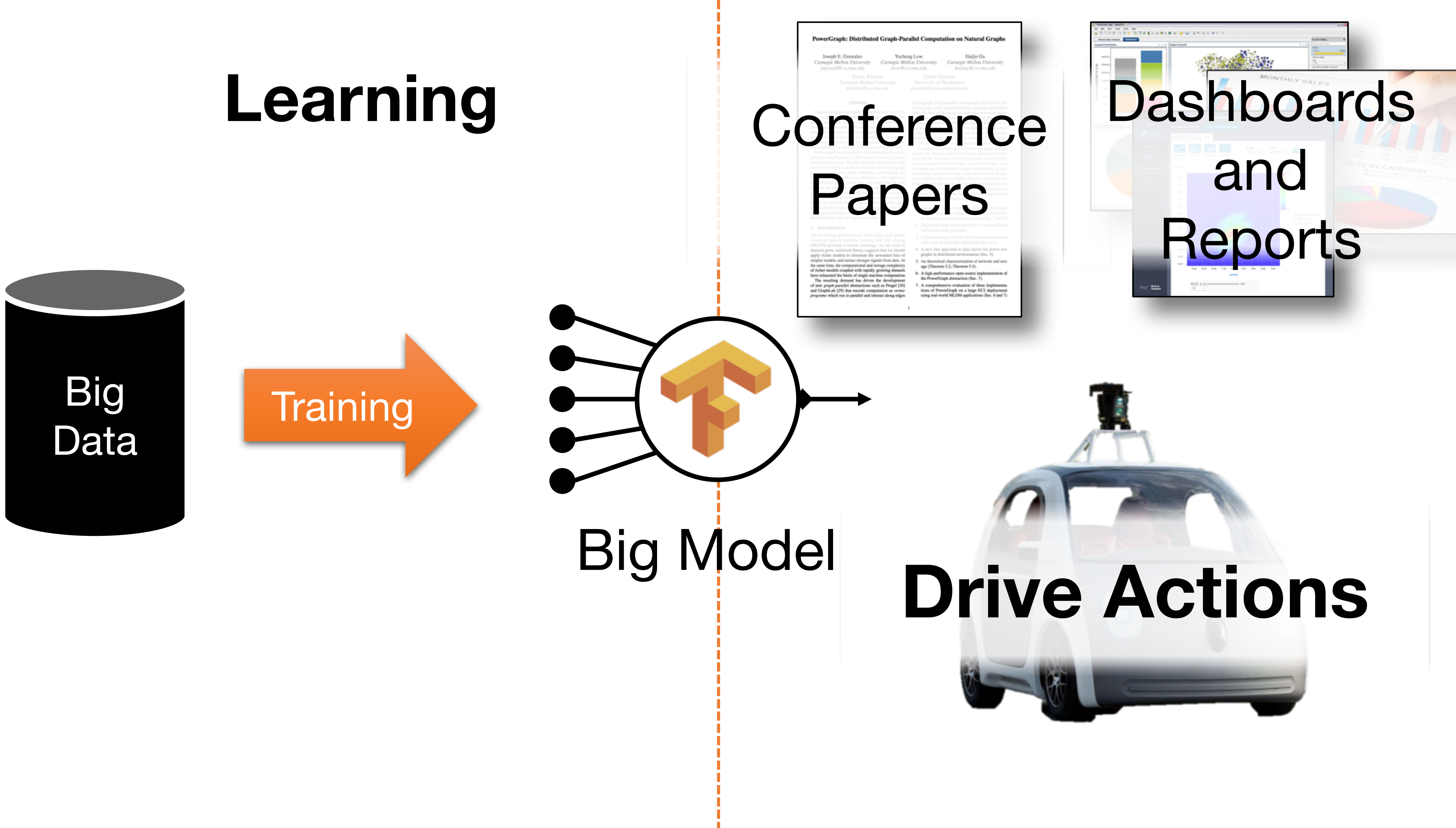
Learning



Learning

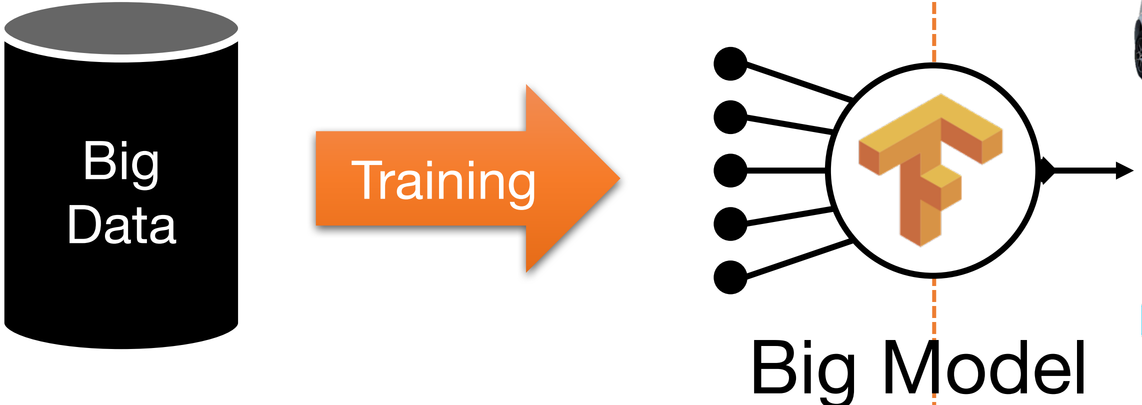

Big Data
Training
Big Model
Conference Papers
PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs


Learning
Big
Data
Training
Big Model
PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs
Conference
Papers
MONTHLY SALES
Dashboards and
Reports


Learning
Big Data
Training
Big Model
PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs
Conference Papers
Dashboards and Reports
Drive Actions

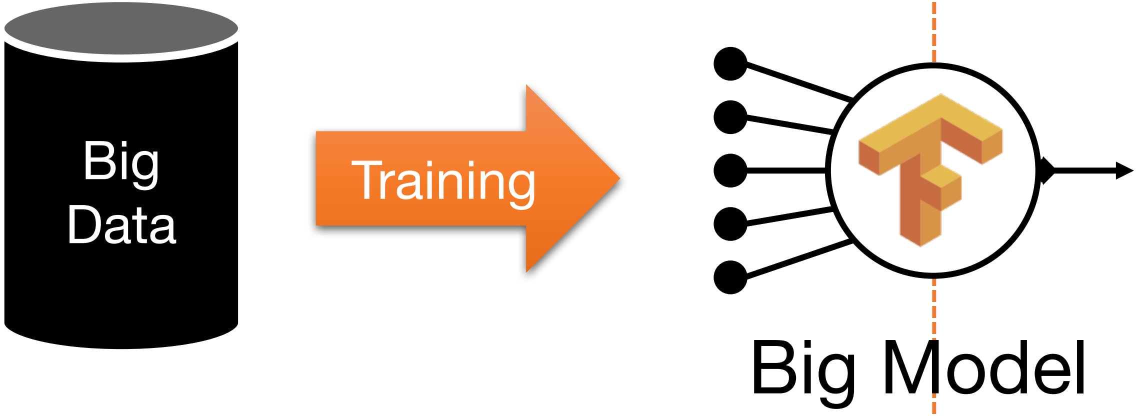
Learning

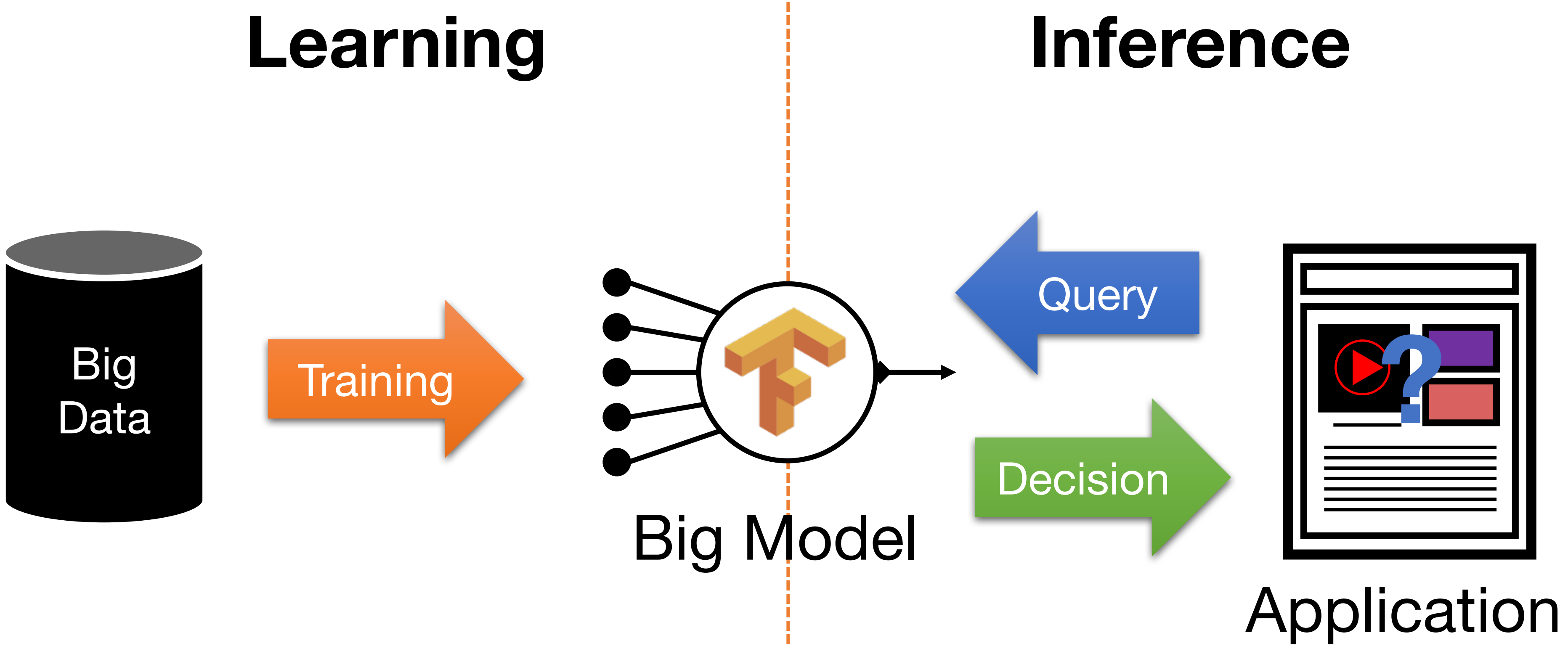
Drive Actions



Learning

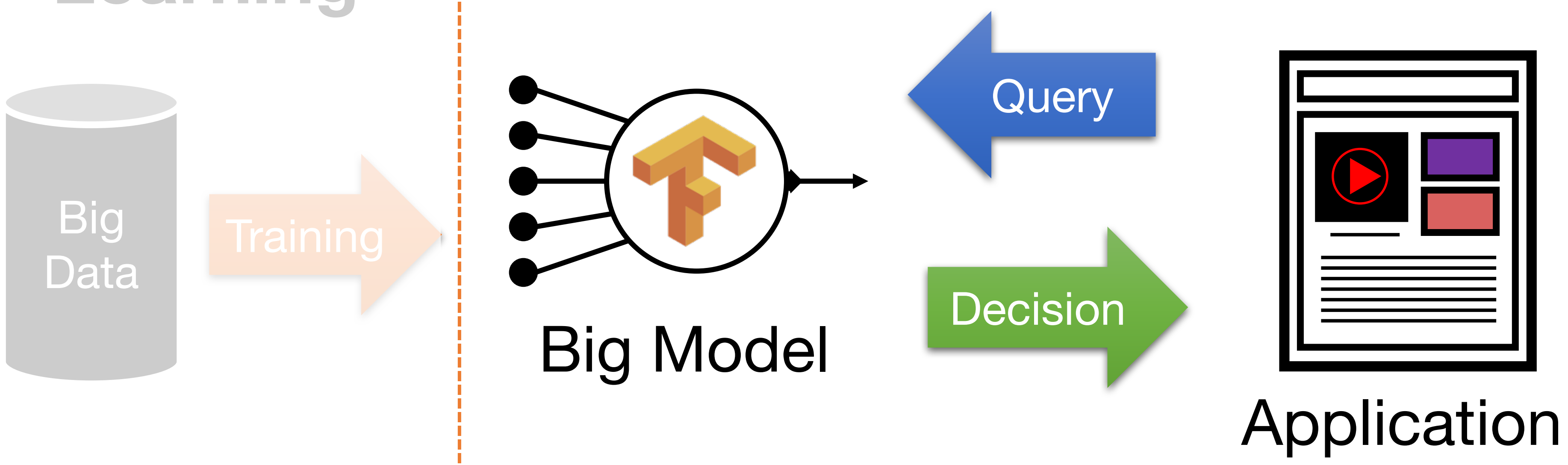
Inference




Learning
Inference
Big
Data
Training
Big Model
Query
Decision
Application

Learning

Inference



Often **overlooked**

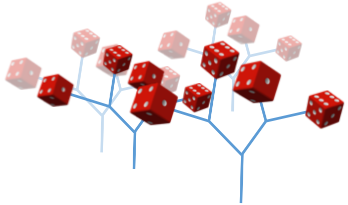
Timescale: ~10 milliseconds

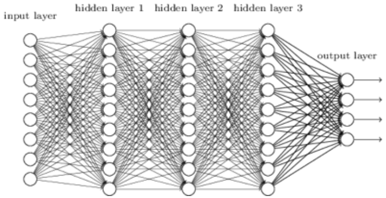
Billions of Queries a Day → Costly

why is **Inference** challenging?

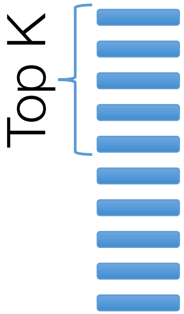
Need to render **low latency** (< 10ms) predictions for **complex**

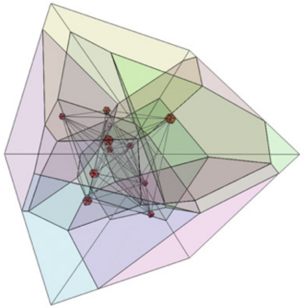
Models





Queries



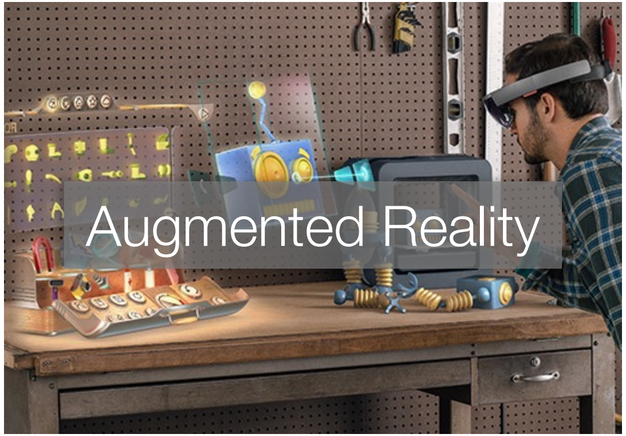


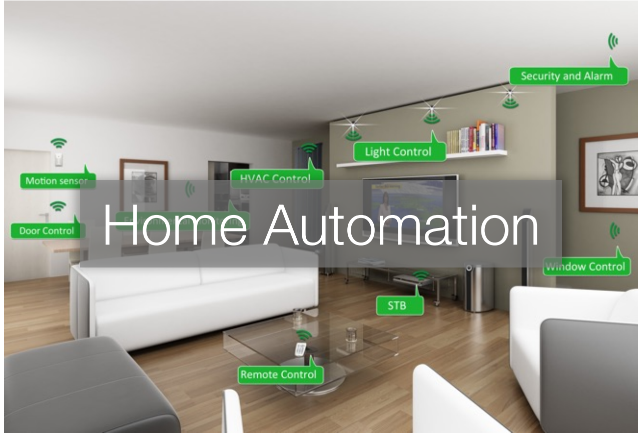
Features

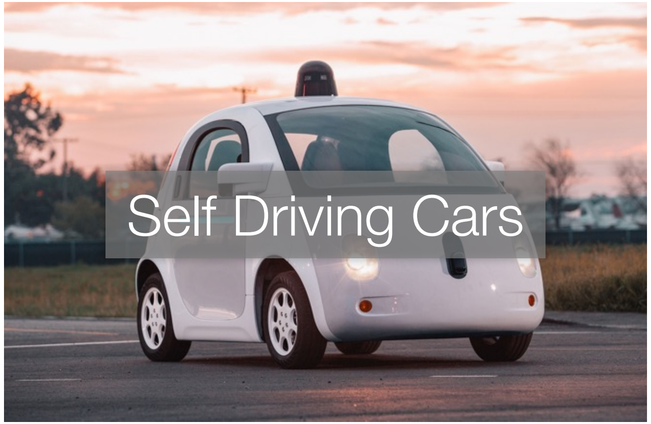
SELECT * FROM users JOIN items, click_logs, pages WHERE …

under **heavy load** with system **failures**.

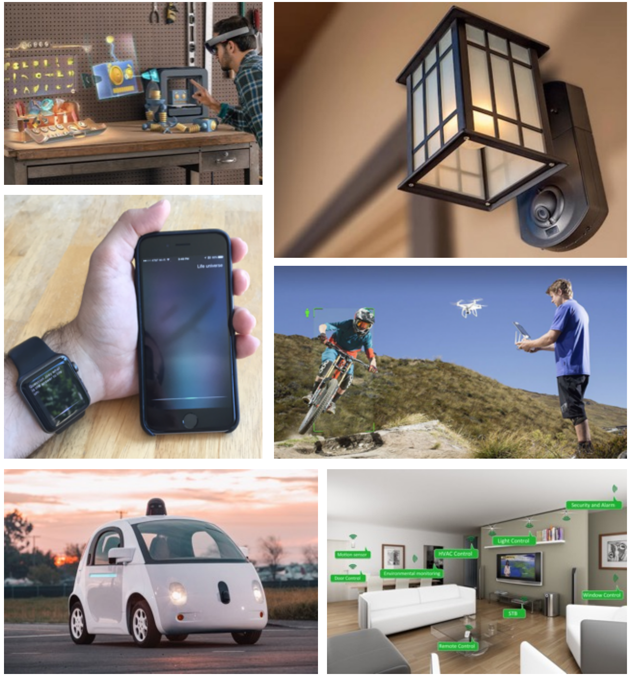
Inference is moving beyond the cloud







Inference is moving beyond the cloud



Opportunities

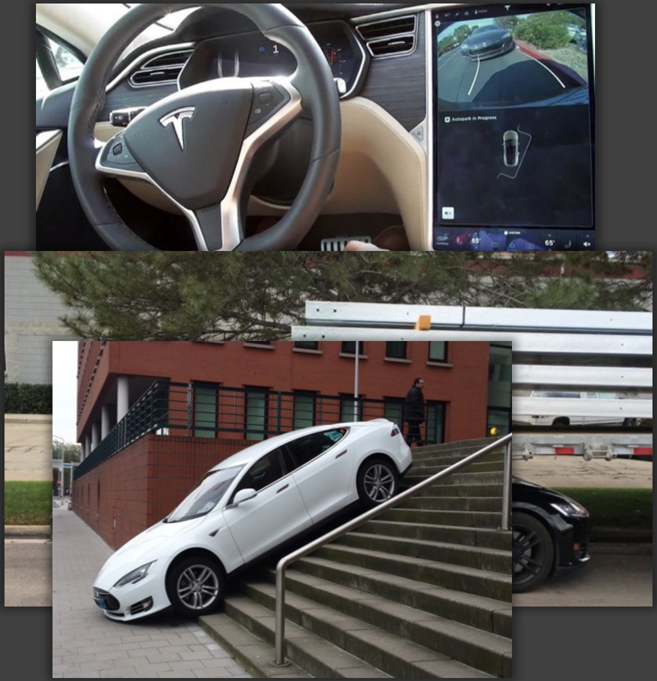
- ➢ Reduce latency and improve privacy
- ➢ Address network partitions

Research Challenges

- ➢ Minimize **power consumption**
- ➢ **Limited hardware** & long life-cycles
- ➢ Develop new **hybrid models** to leverage the cloud and edge devices

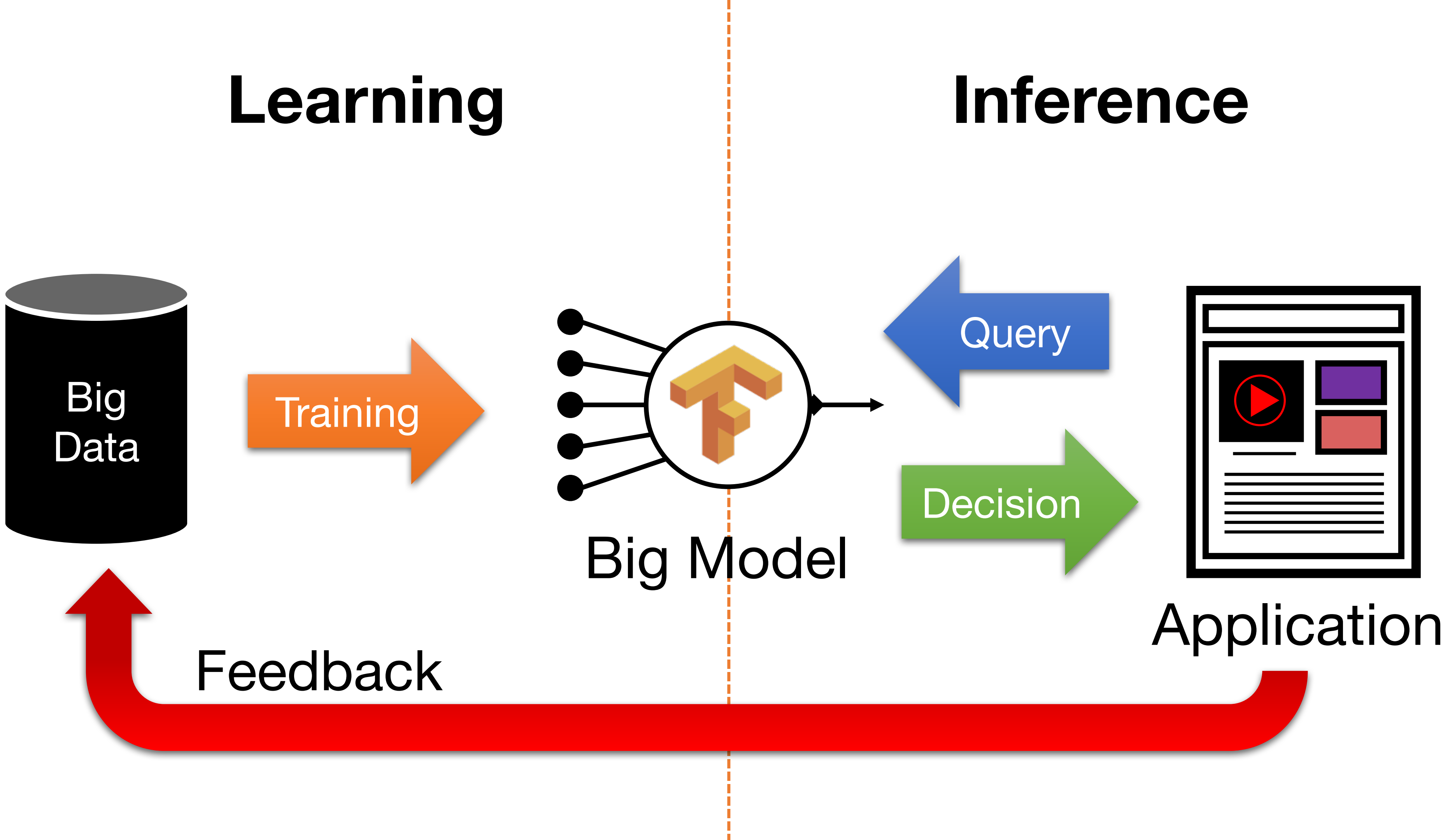
Robust **Inference** is critical

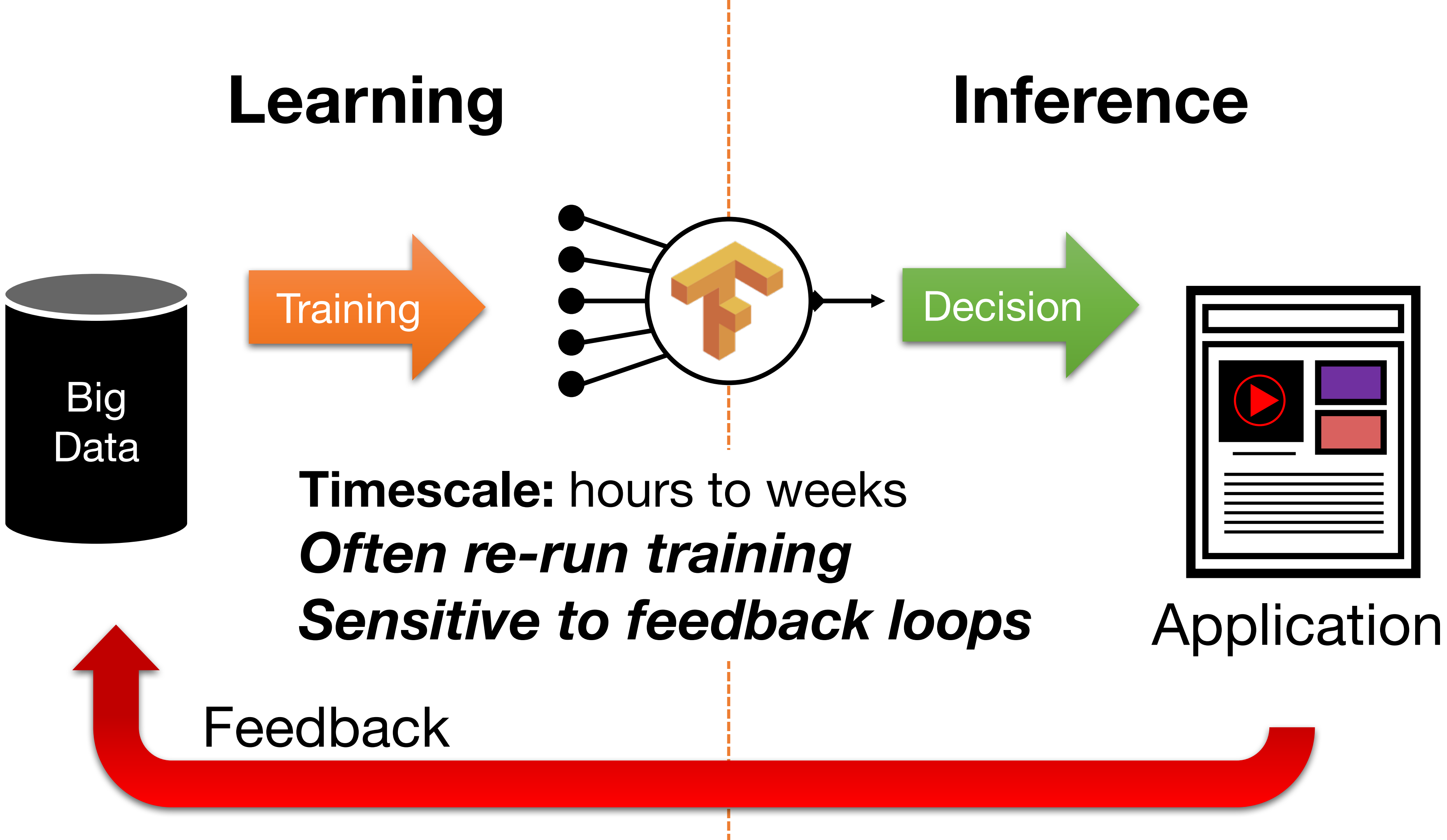
Self “*Parking*” Cars



Self “*Driving*” Cars

Chat AIs


Learning
Inference
Big
Data
Training
Big Model
Query
Decision
Application
Feedback


Learning
Inference
Big Data
Training
Decision
Application
Timescale: hours to weeks
Often re-run training
Sensitive to feedback loops
Feedback

Why is **Closing the Loop** challenging?

Implicit and **Delayed** Feedback

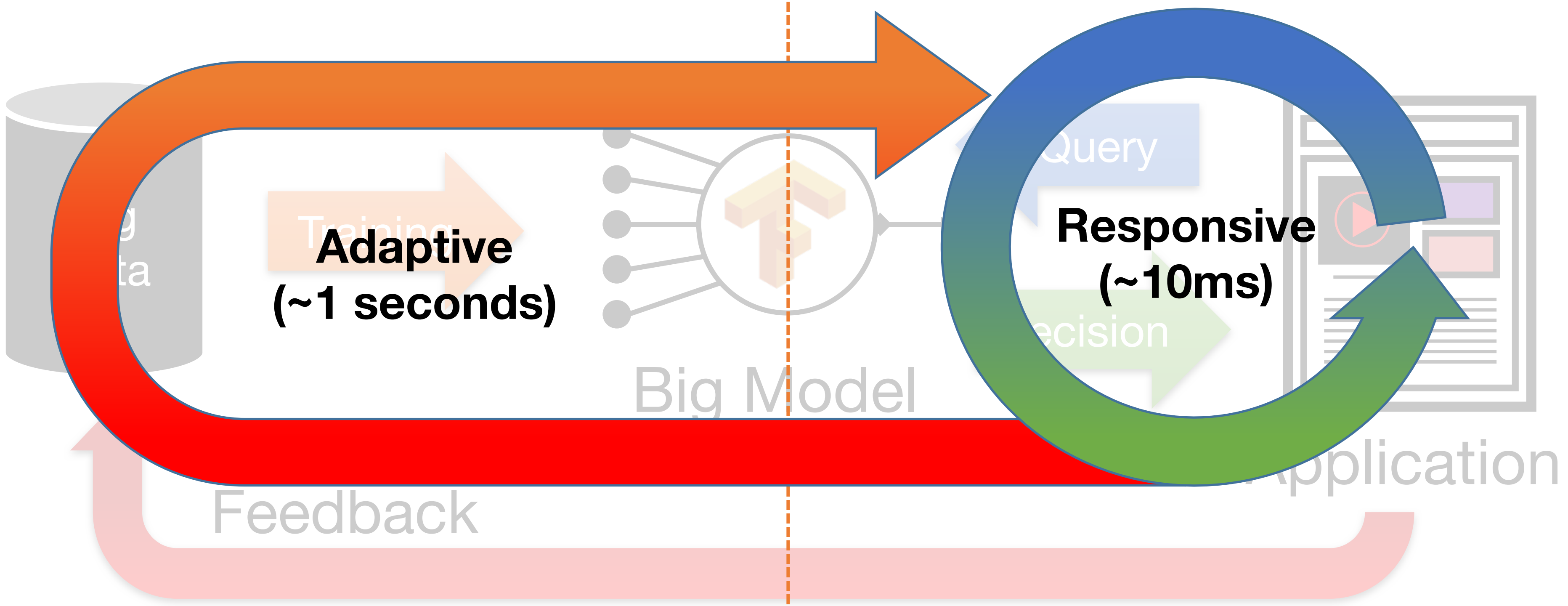
Self Reinforcing **Feedback Loops**

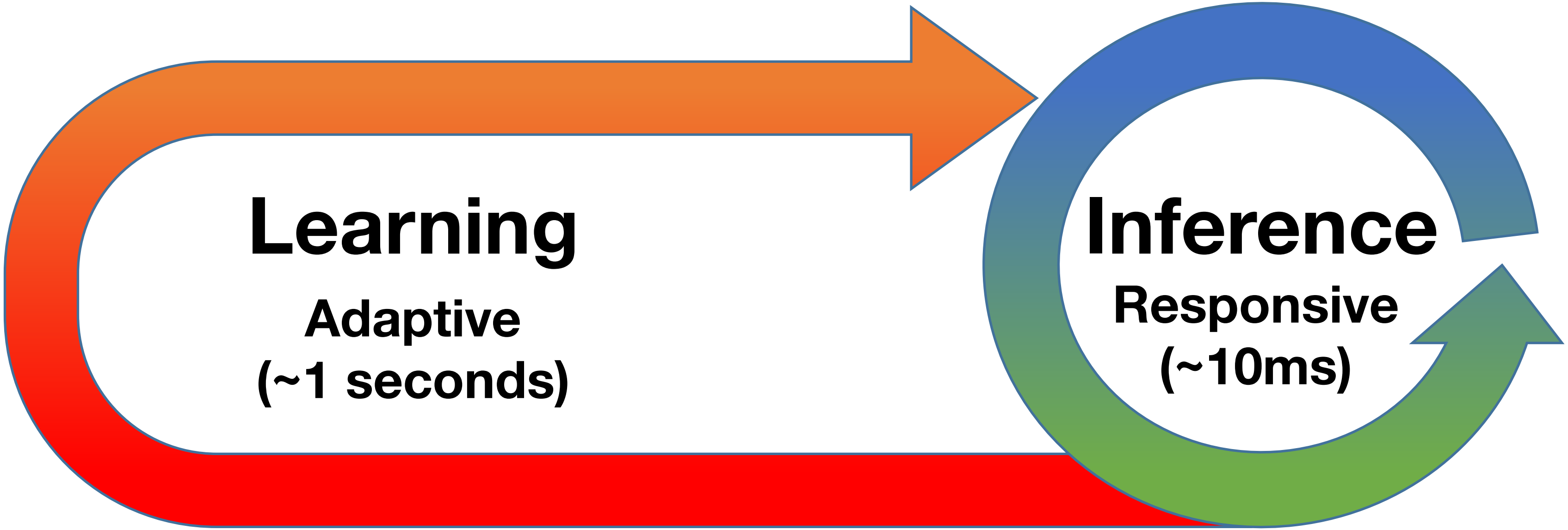
$$\frac{d}{dt}$$

World Changes **at varying rates**

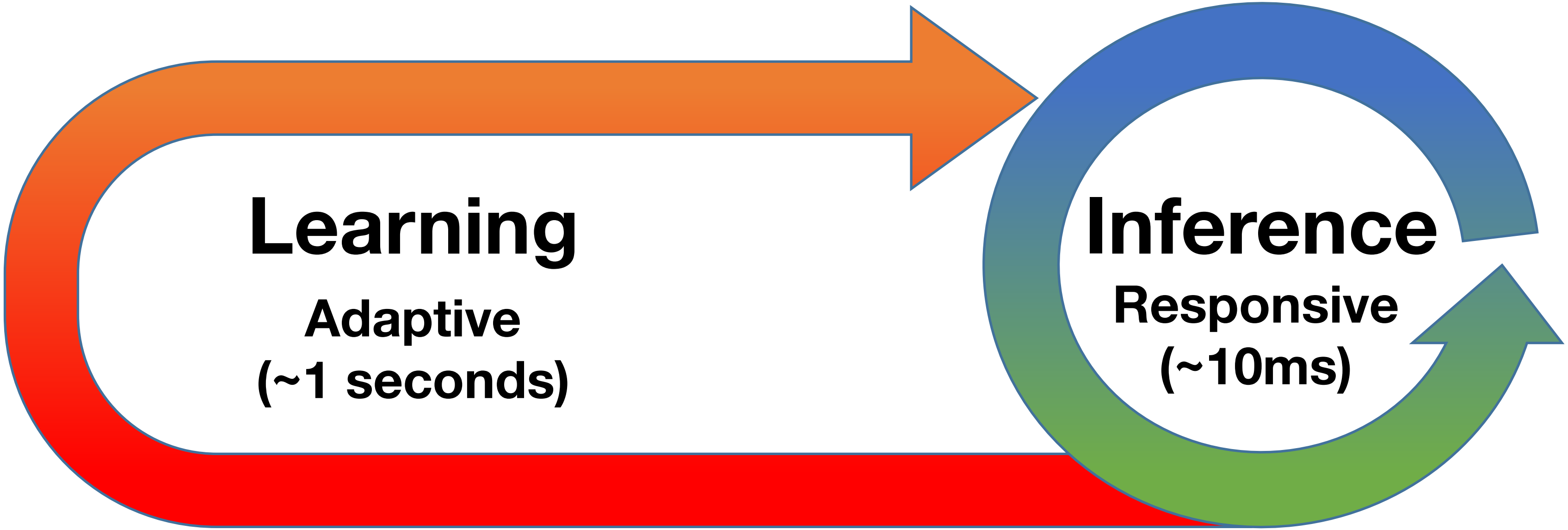
Learning

Inference




Learning
Adaptive
(~1 seconds)
Inference
Responsive
(~10ms)

?



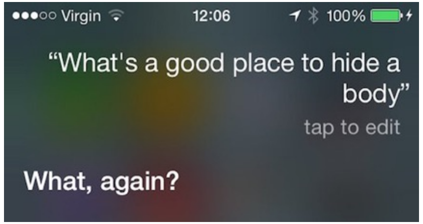
Secure

Intelligence in **Sensitive Contexts**

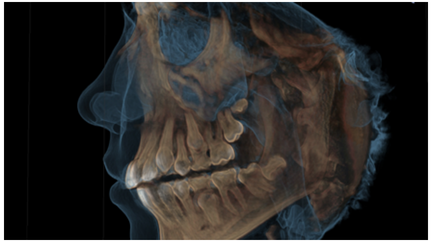
Augmented Reality

Home Monitoring

Voice Technologies



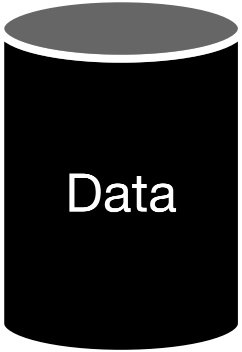
Medical Imaging



Protect the **data**, the **model**, and the **query**

Protect the **data**, the **model**, and the **query**

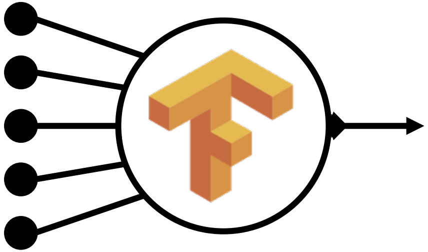
High-Value **Data** *is* ***Sensitive***



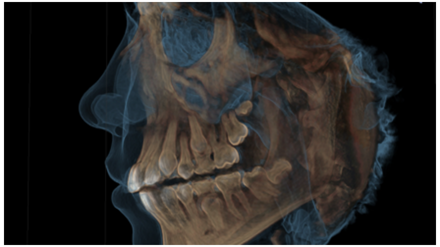
- Medical Info.
- Home video
- Finance

Models capture **value** in data

- Core Asset
- Sensitive



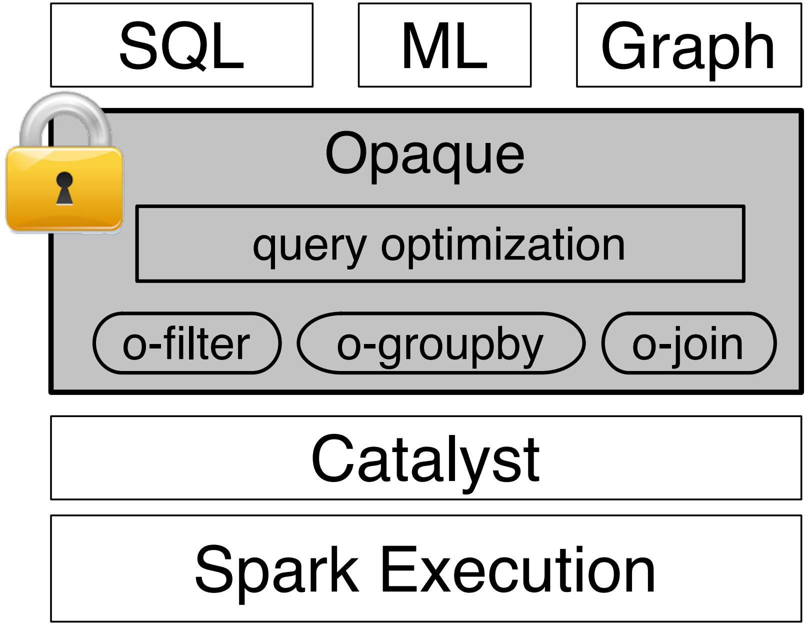
Queries can be as sensitive as the data



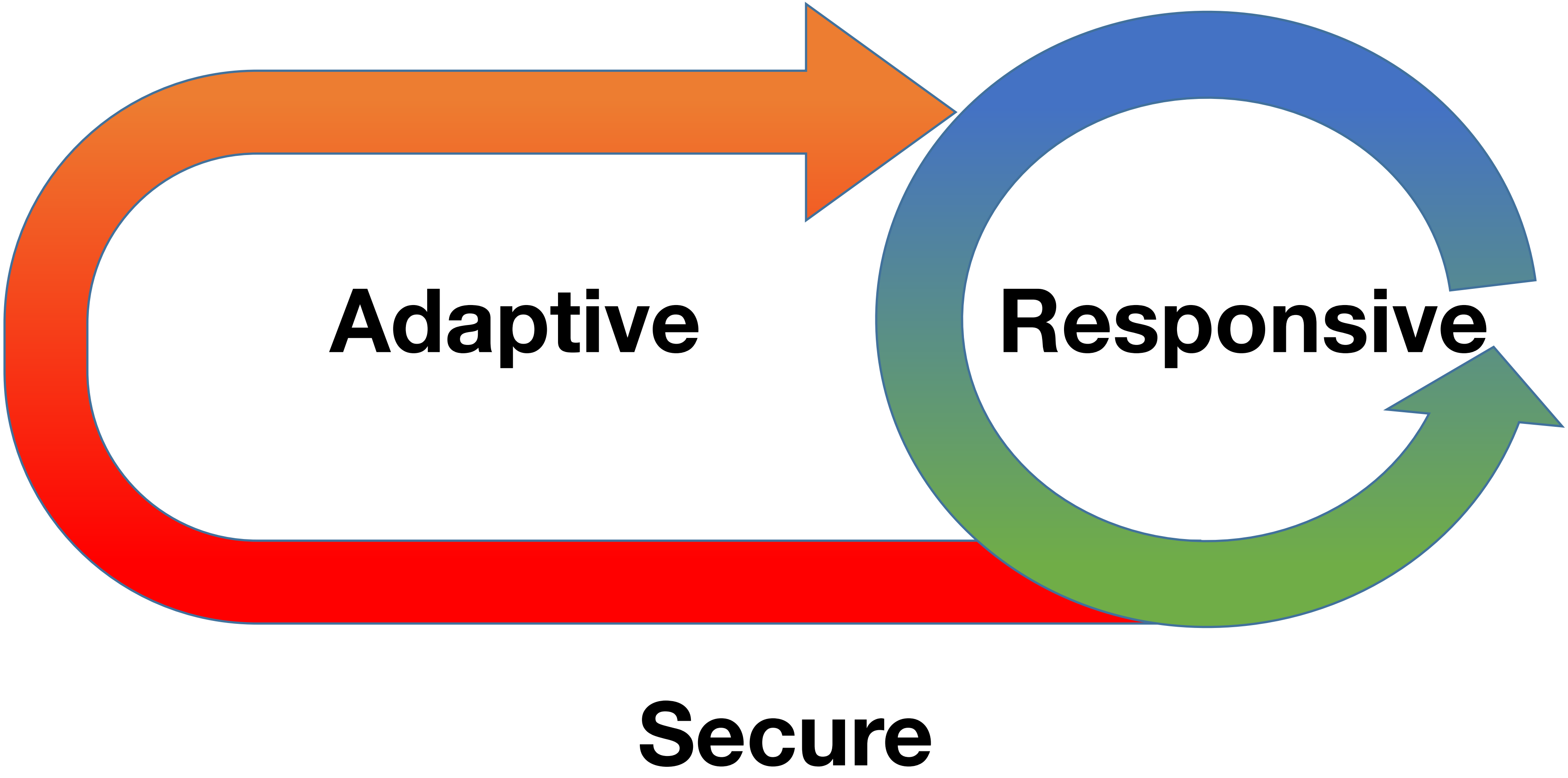
Opaque: Analytics on Secure Enclaves

Exploit hardware support to enable computing on encrypted data

- **Today:** prototype system running in Apache Spark
  - support SQL queries in untrusted cloud
  - ~50% reduction in perf.
- **Future:** enable prediction serving on enc. queries



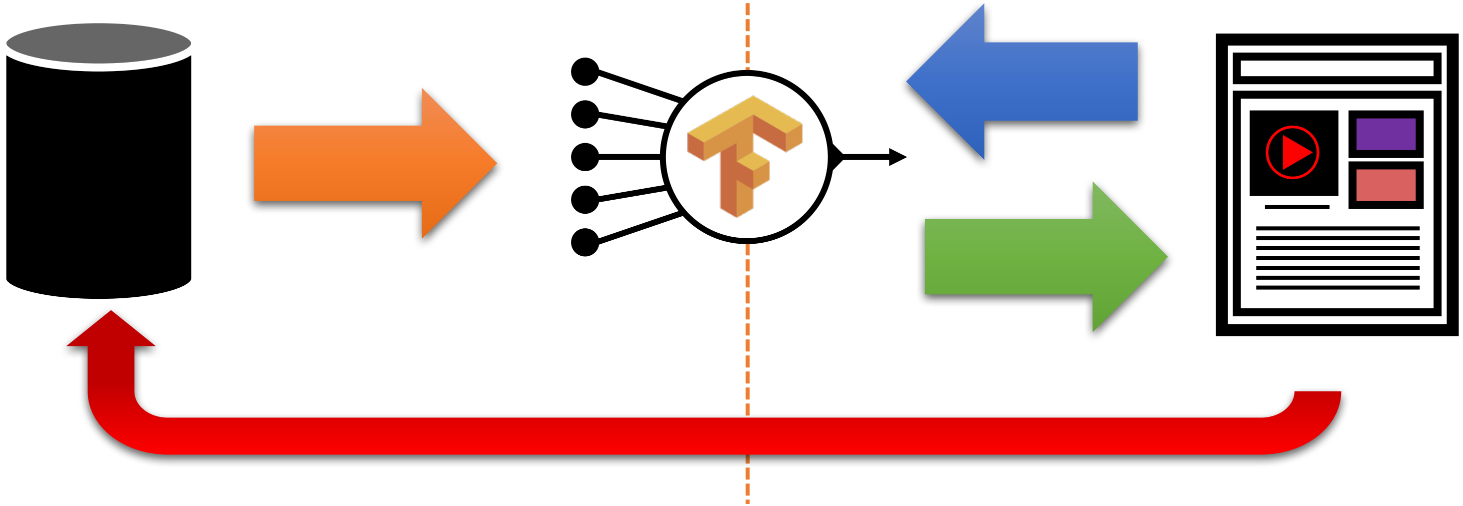
Wenting et al. (NSDI'17)


Adaptive
Responsive
Secure

amplab

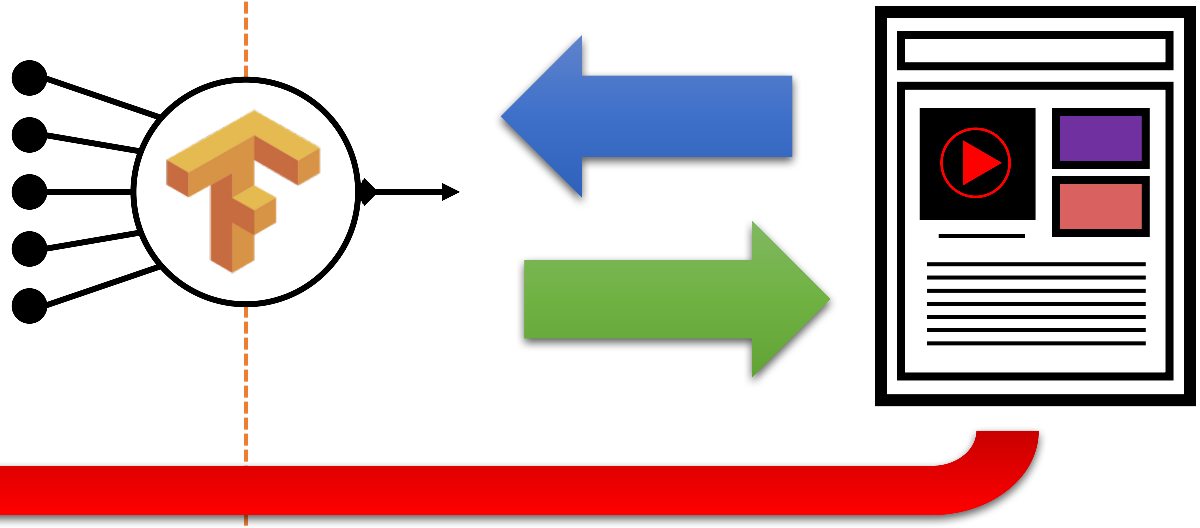
riselab
UC Berkeley

amplab



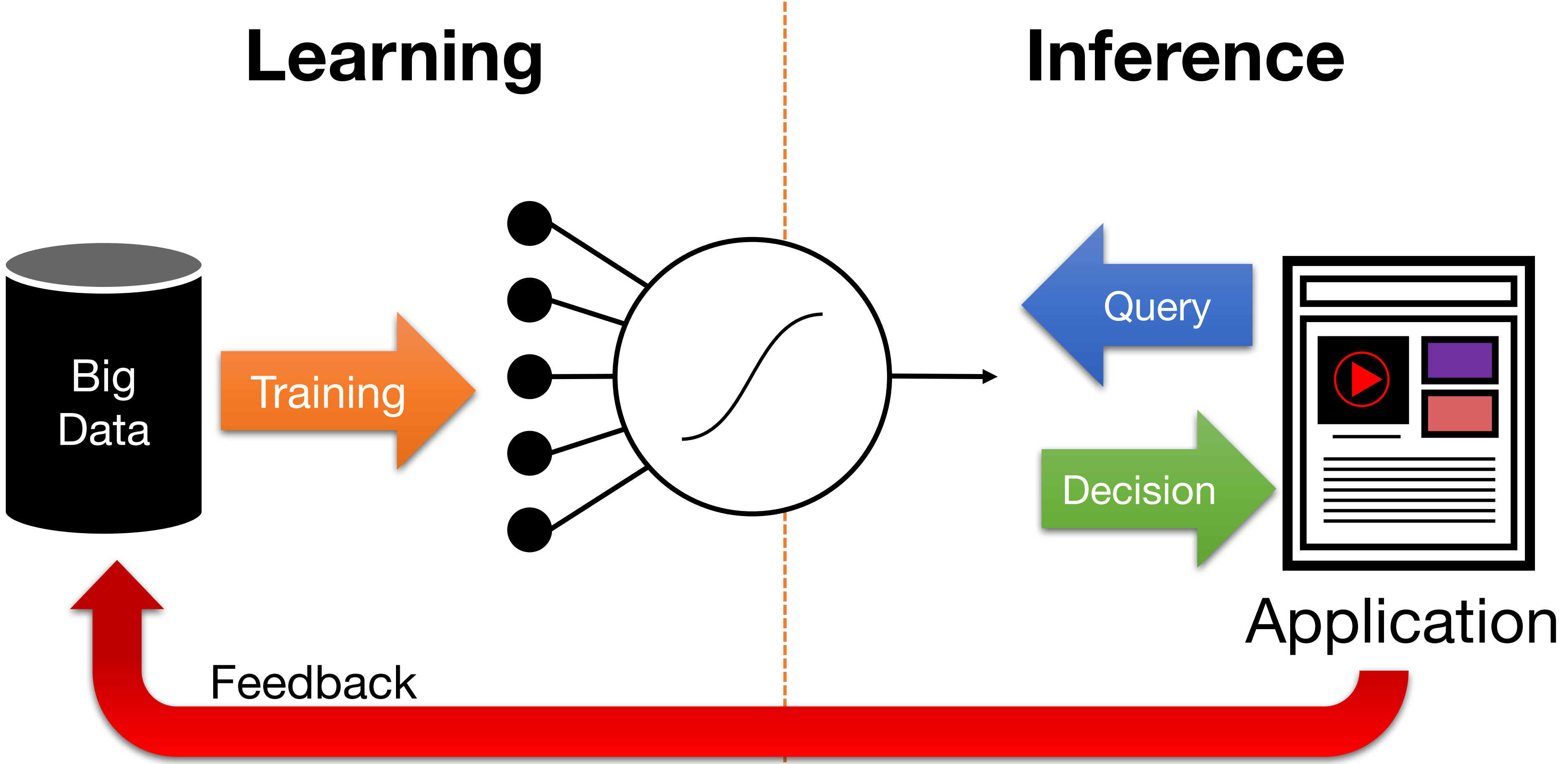
Clipper

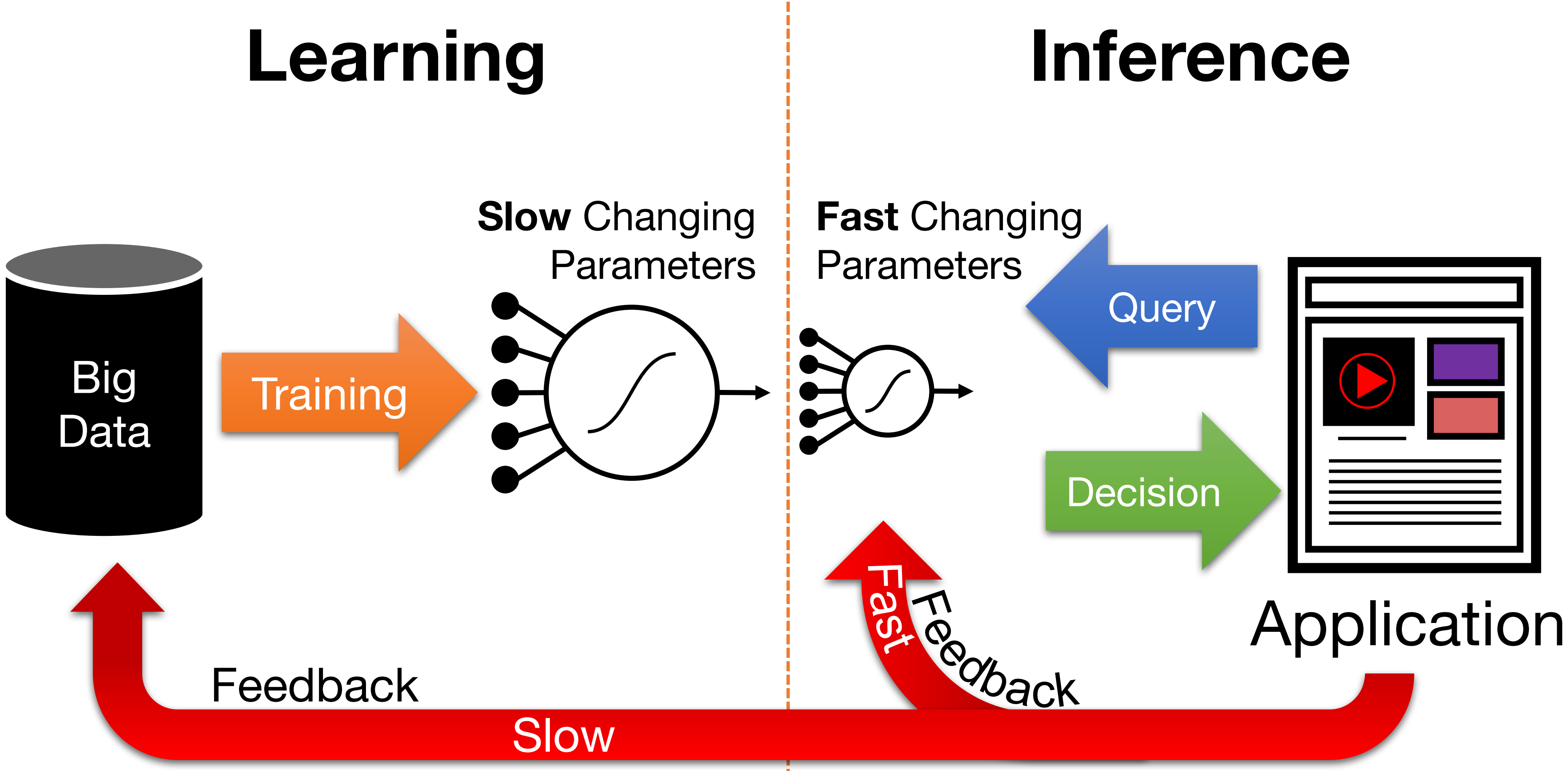
A Low-Latency Online Prediction Serving System



NSDI'17

Daniel Crankshaw
Xin Wang
Giulio Zhou
Michael J. Franklin
Joseph E. Gonzalez
Ion Stoica


Learning
Inference
Big Data
Training
Query
Decision
Application
Feedback


Learning
Inference
Slow Changing Parameters
Fast Changing Parameters
Big Data
Training
Query
Decision
Application
Feedback
Slow
Fast
Feedback

Hybrid Offline + Online Learning

Update "feature" functions **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 **online**:

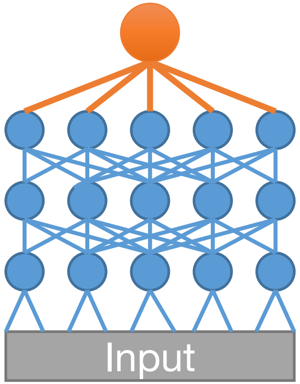
- Simple to train + more robust model
- Address rapidly changing user statistics

Common modeling structure

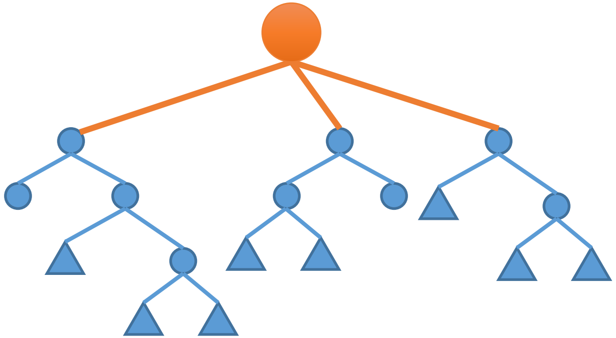
$$f(x;\theta)^T w_u$$

Matrix Factorization

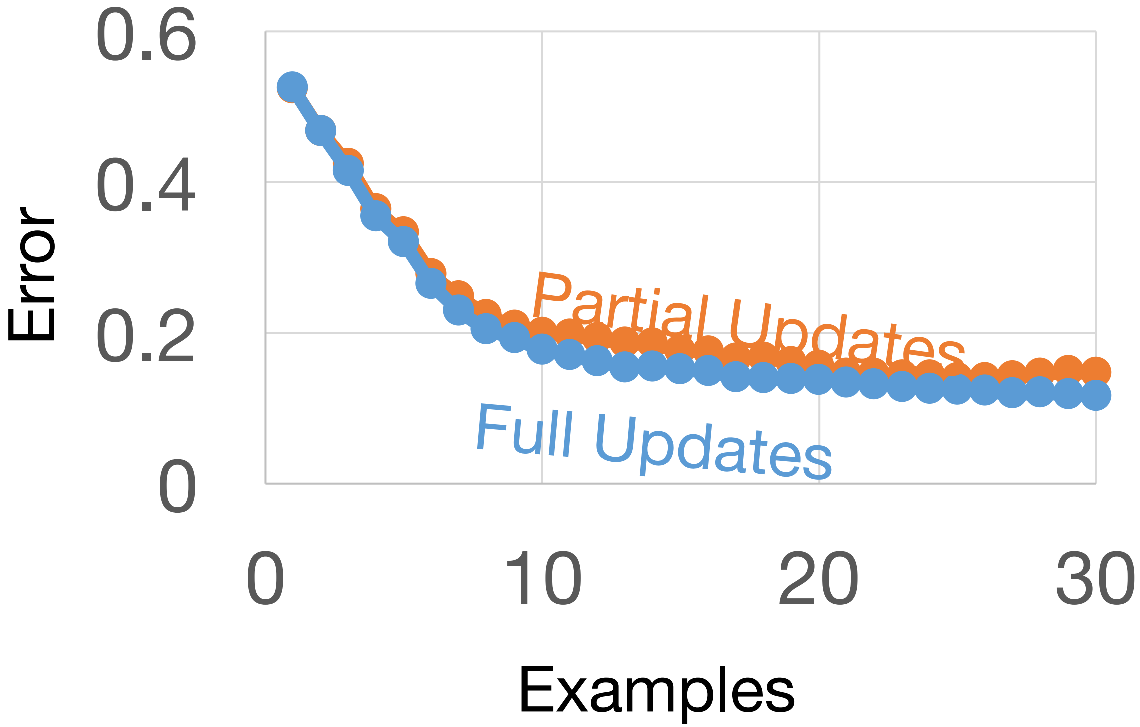
Deep Learning



Ensemble Methods

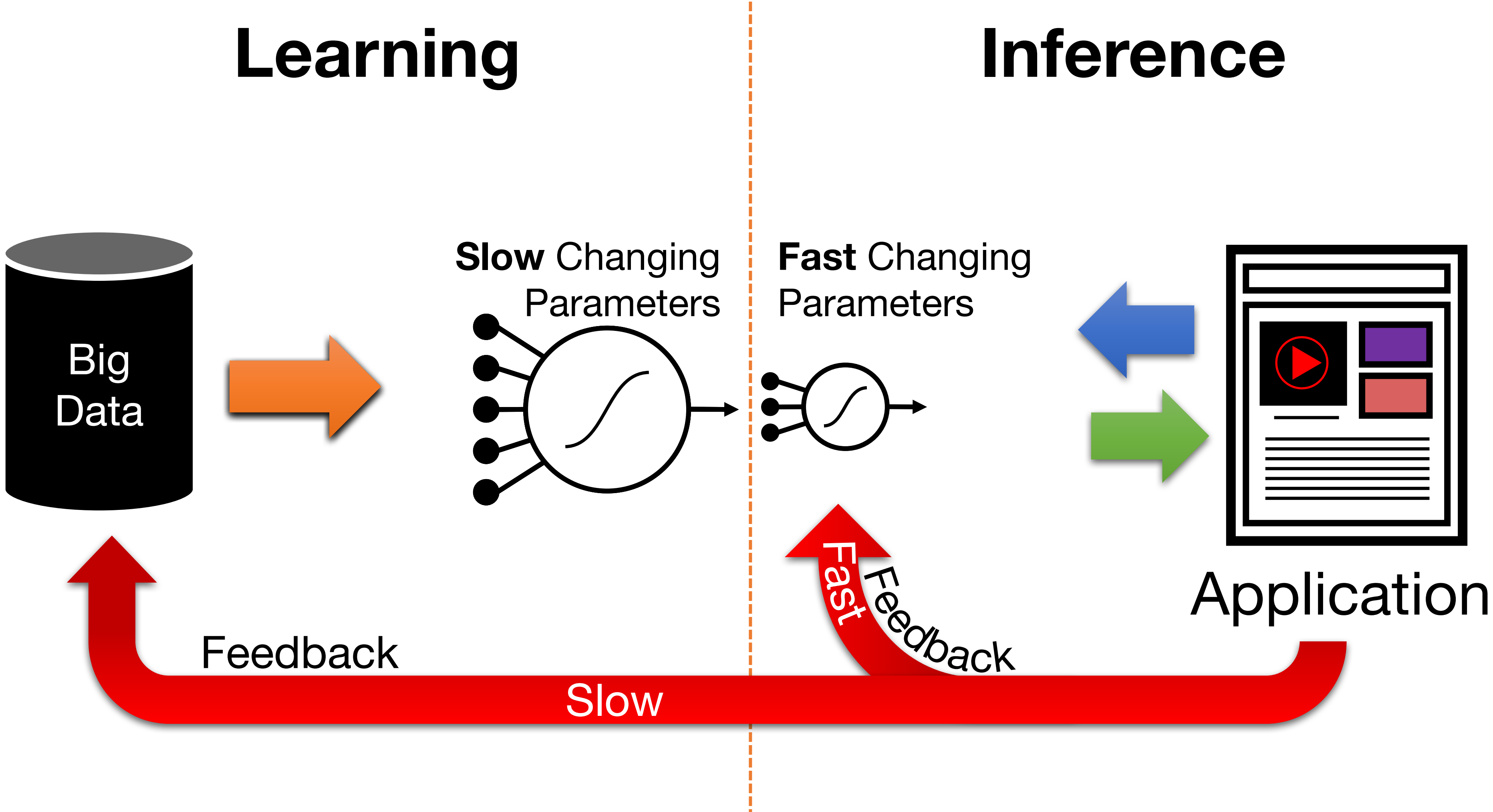


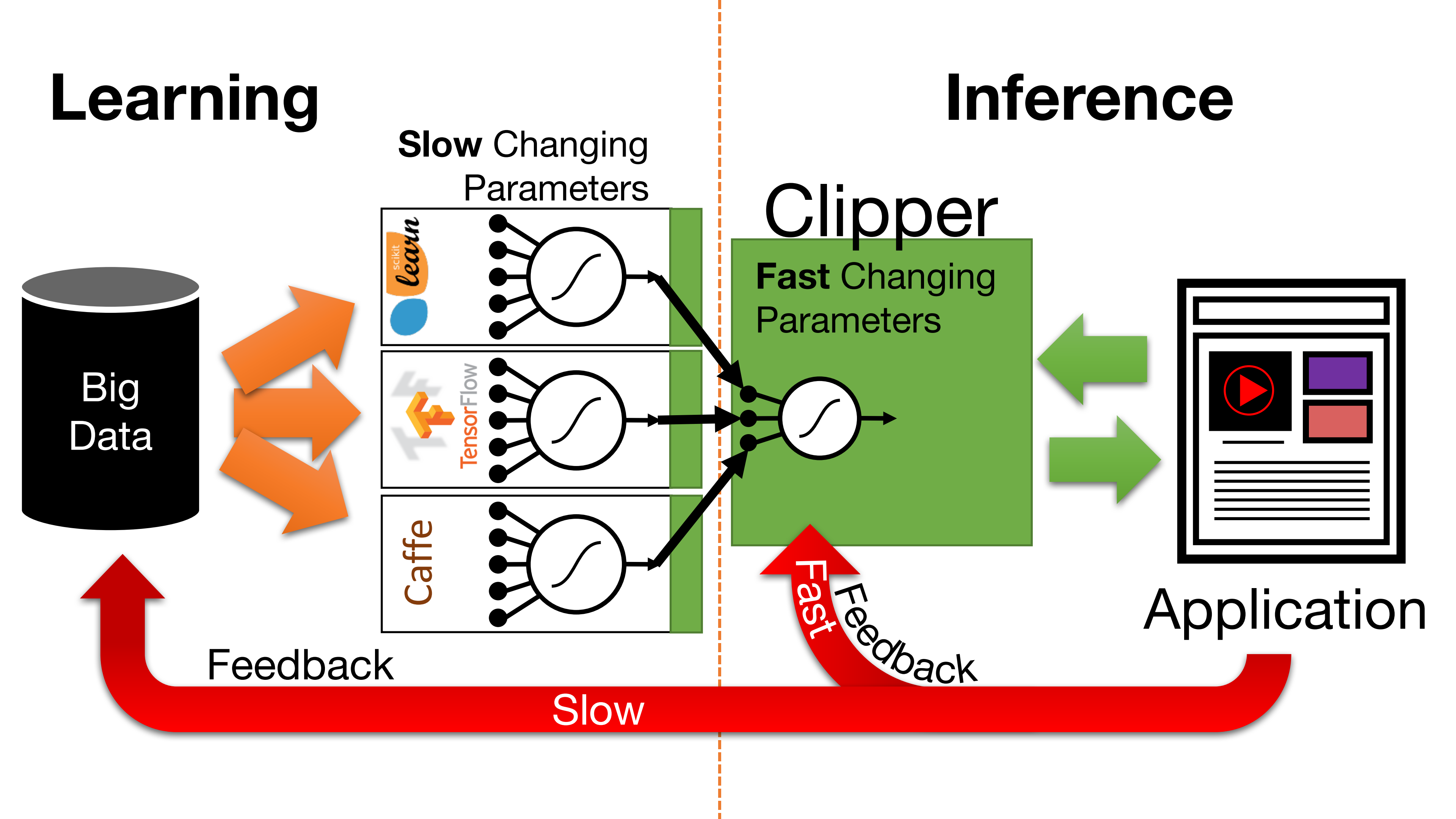
Clipper Online Learning for Recommendations
(Simulated News Rec.)



Partial Updates: *0.4 ms*
Retraining: *7.1 seconds*

>4 orders-of-magnitude ***faster adaptation***


Learning
Inference
Big Data
Slow Changing Parameters
Fast Changing Parameters
Application
Feedback
Slow
Fast
Feedback



Learning
Inference
Slow Changing Parameters
Clipper
Fast Changing Parameters
Big Data
scikit learn
TensorFlow
Caffe
Application
Feedback
Slow
Fast
Feedback

Clipper Serves Predictions across ML Frameworks

Fraud Detection

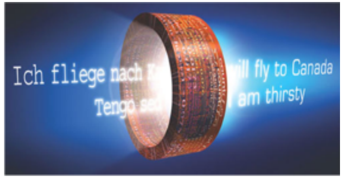
Content Rec.

NETFLIX

Personal Asst.

Robotic Control

Machine Translation



Clipper

theano

Dato Create

KeystoneML

Caffe

TensorFlow

scikit learn

dmlc mxnet

VW

KALDI

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 interface to applications
- **bound latency** and **maximize throughput**
  - through *caching, adaptive batching, model replication*
- enable *robust* **online learning** and **personalization**
  - through *model selection and ensemble algorithms*

without modifying *machine learning* ***frameworks*** *or front-end* ***applications***

Clipper Architecture

Fraud Detection

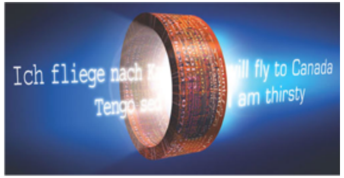
Content Rec.

NETFLIX

Personal Asst.

Robotic Control

Machine Translation

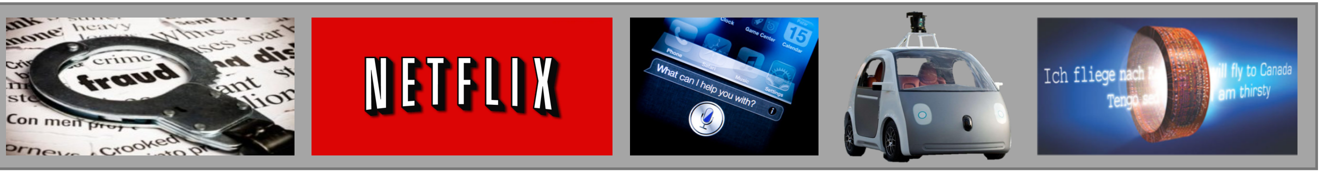


Clipper

theano
Dato Create
scikit learn
VW
KeystoneML
Caffe
TensorFlow
dmlc mxnet
KALDI

Clipper Architecture

Applications

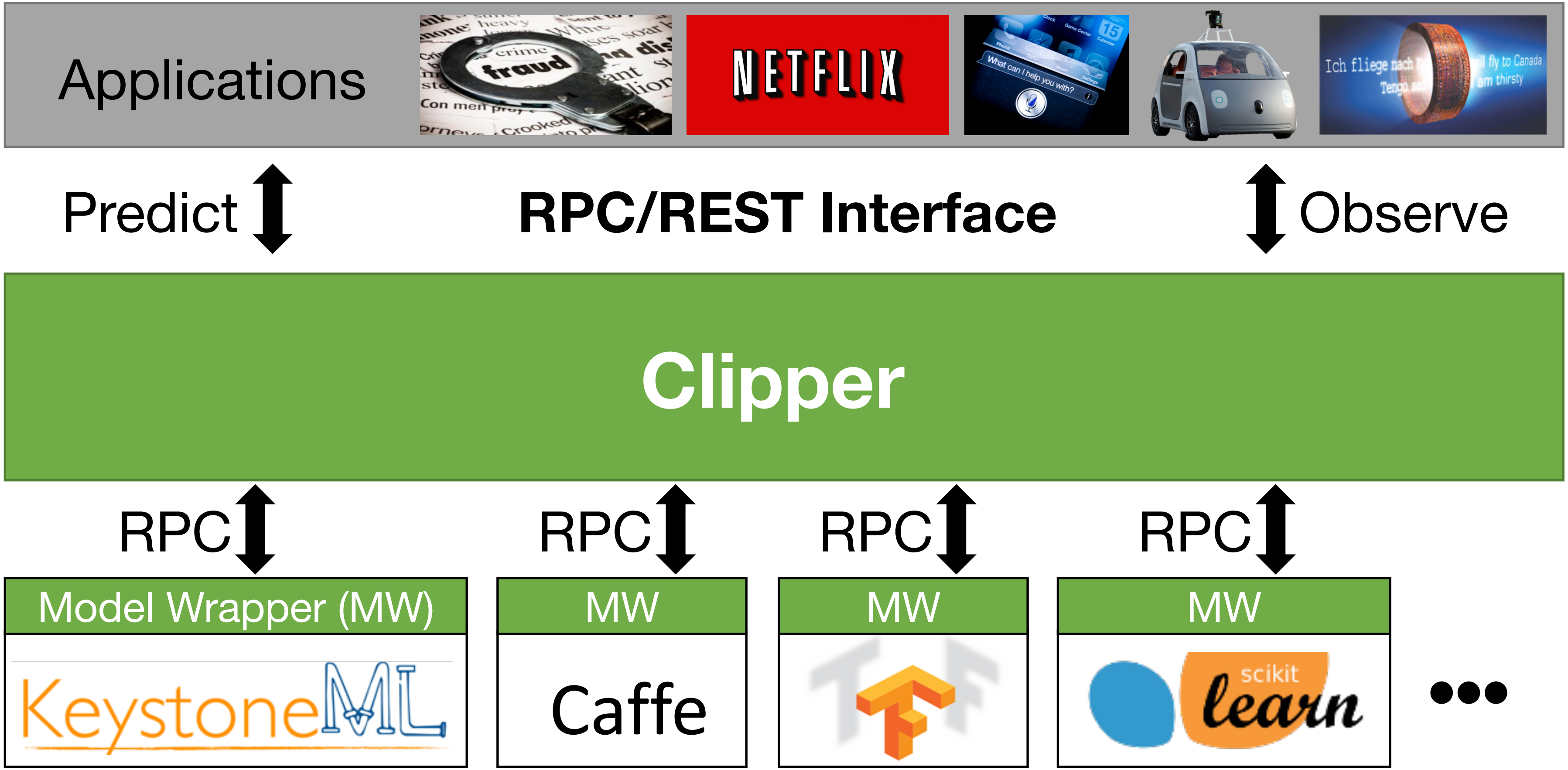


Predict ↕ **RPC/REST Interface** ↕ Observe

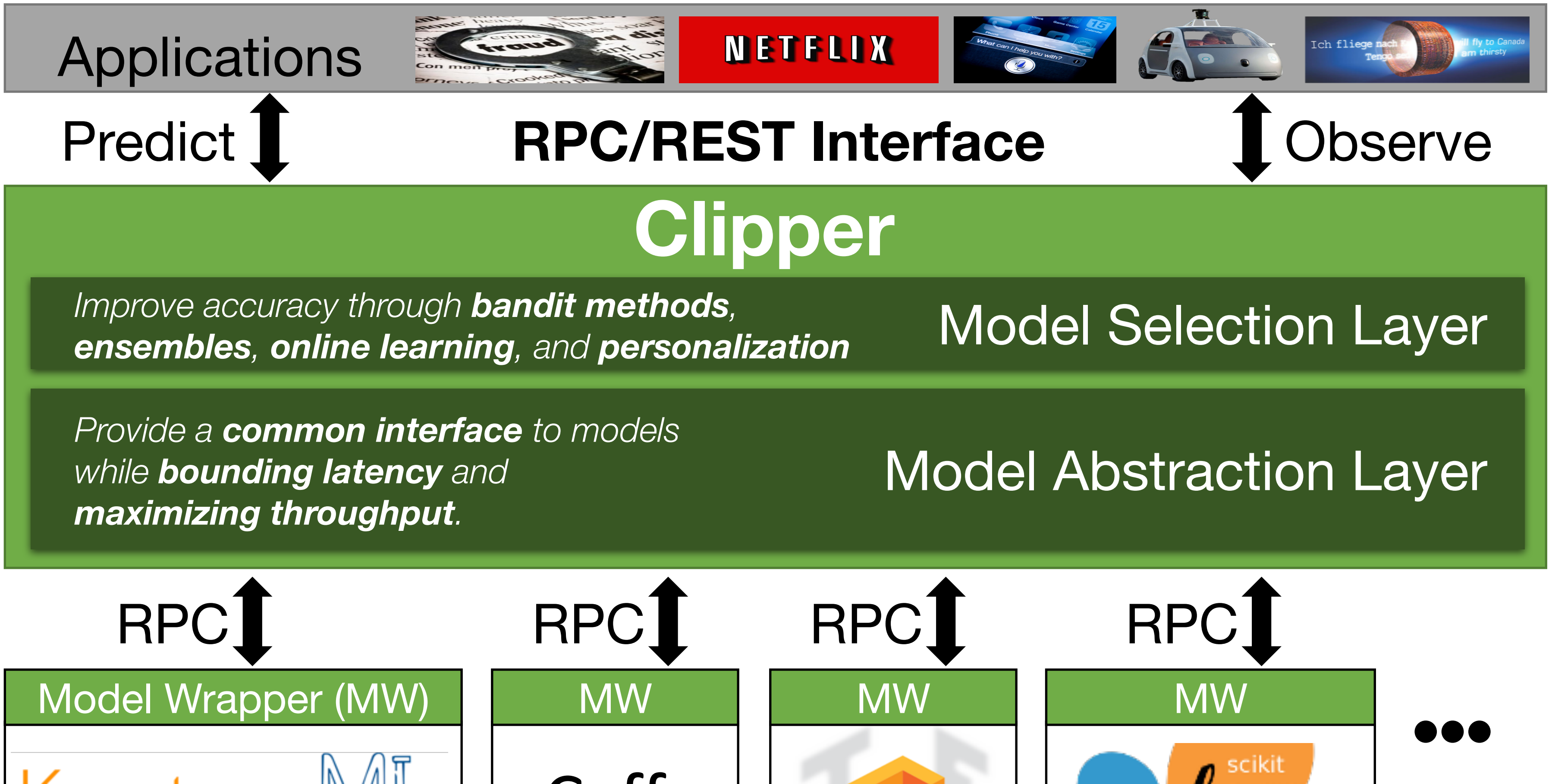
Clipper

theano
Dato Create
KeystoneML
Caffe
TensorFlow
scikit learn
dmlc mxnet
VW
KALDI

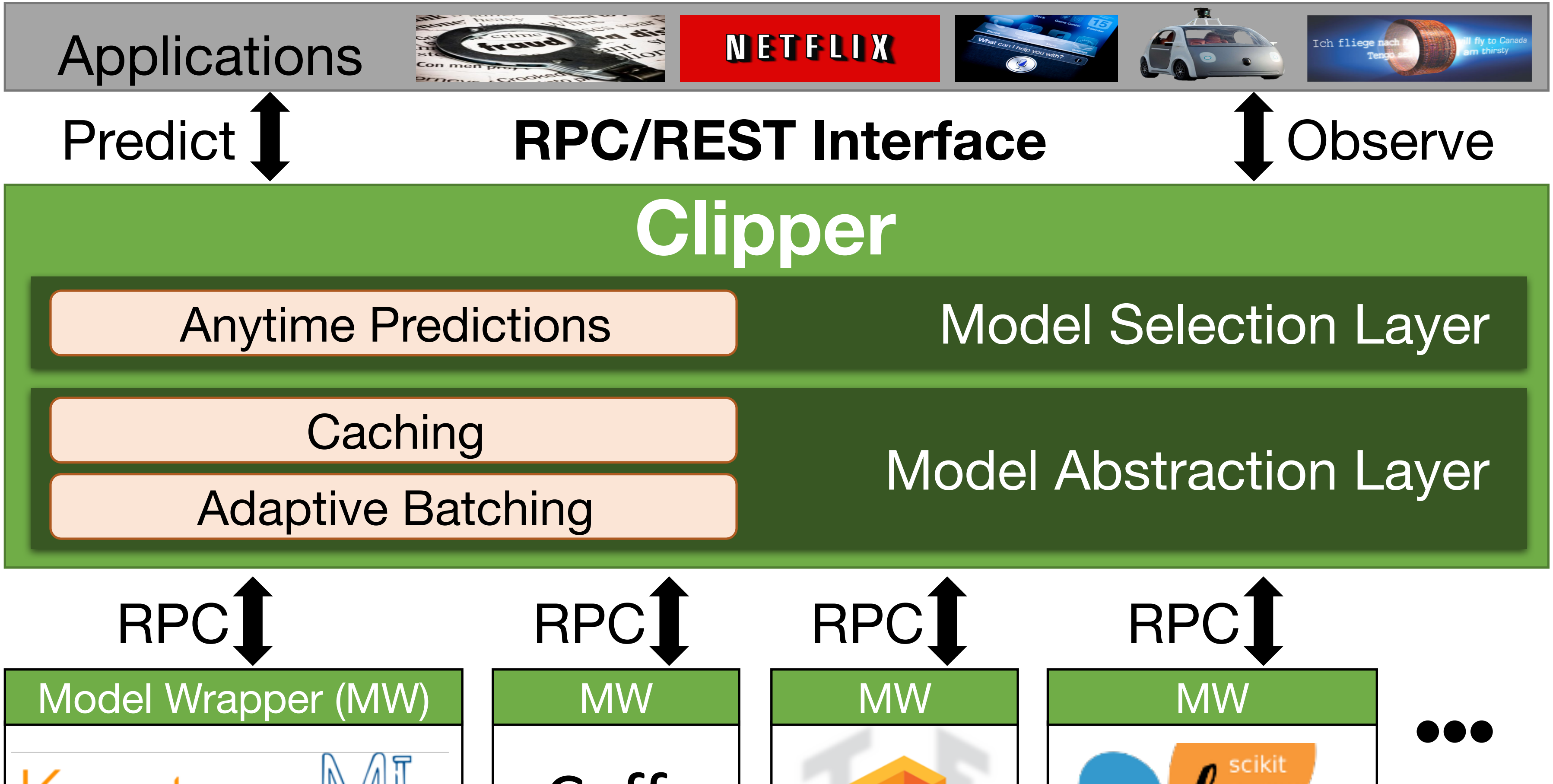
Clipper Architecture

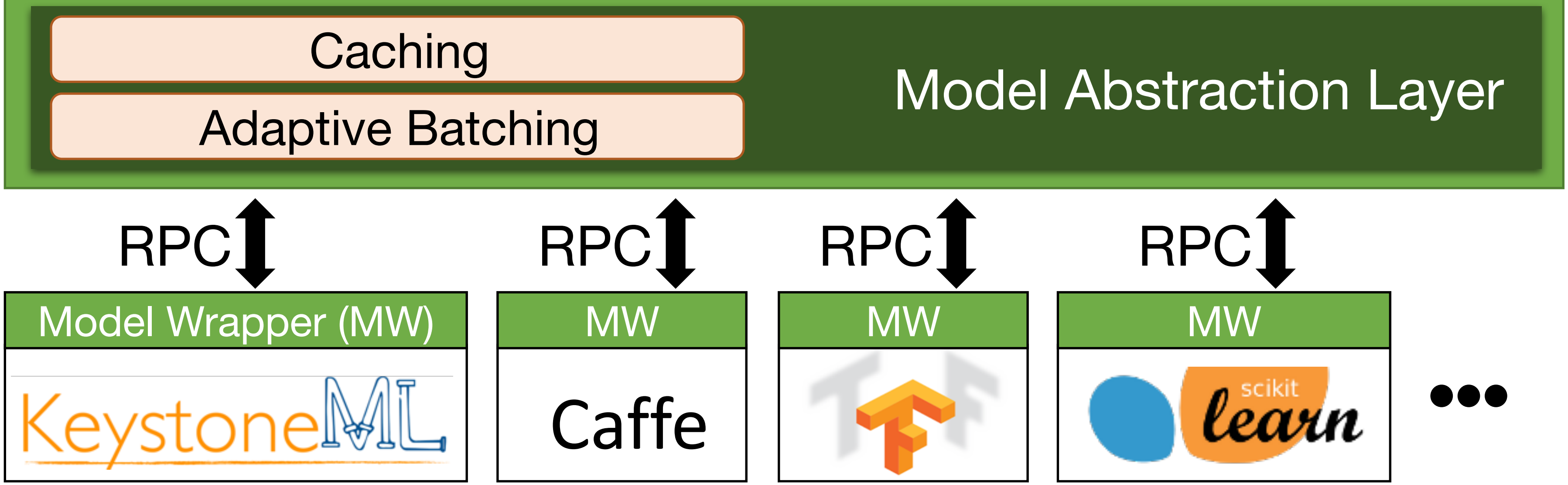


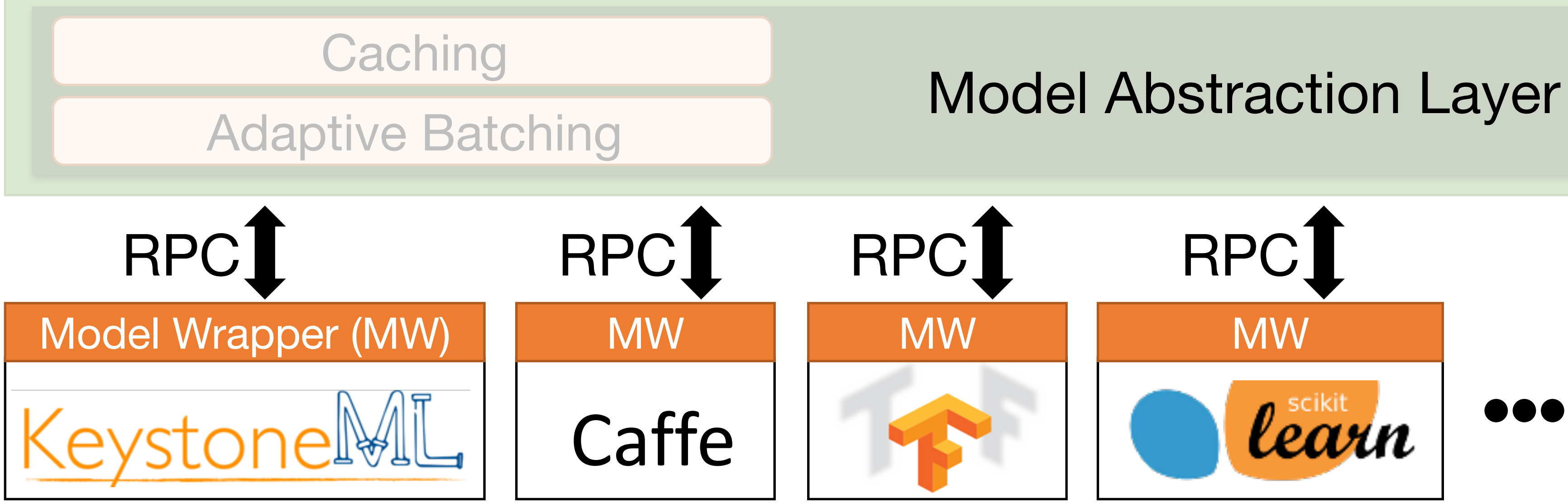
Clipper Architecture



Clipper Architecture

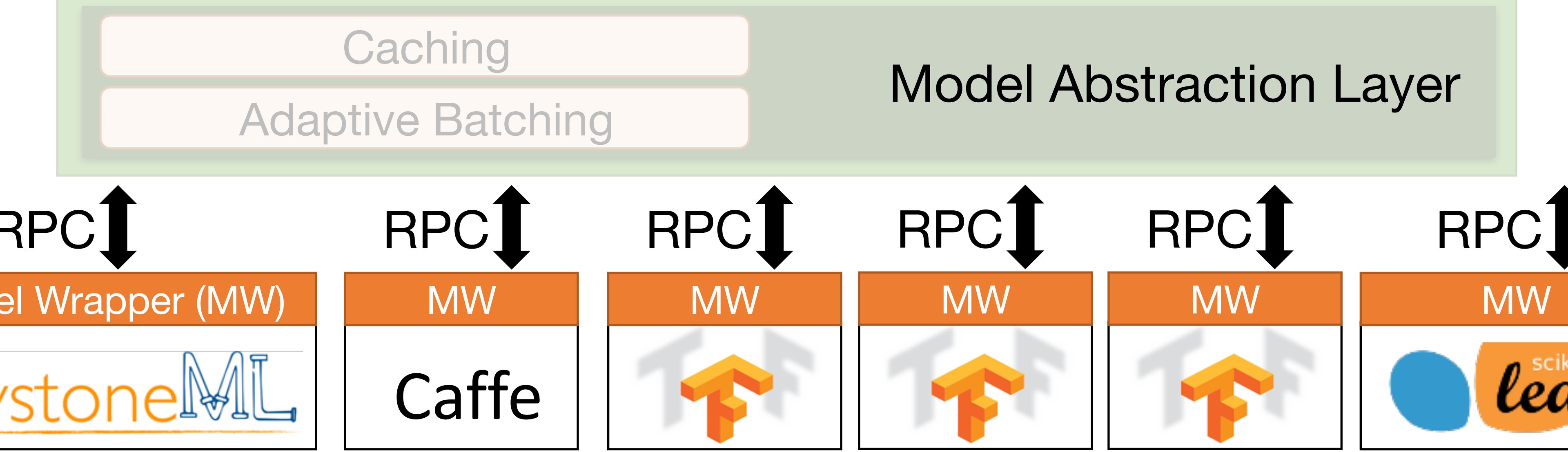



Caching
Adaptive Batching
Model Abstraction Layer
RPC
RPC
RPC
RPC
Model Wrapper (MW)
MW
MW
MW
KeystoneML
Caffe
scikit learn



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

- Models run in separate processes as Docker containers
    - Resource isolation



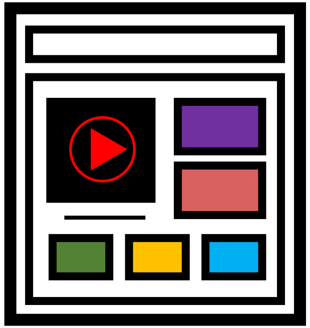
Provide a **common interface** to models while **bounding latency** and **maximizing throughput.**

- Models run in separate processes as Docker containers
  - Resource isolation
- Scaling under heavy load

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

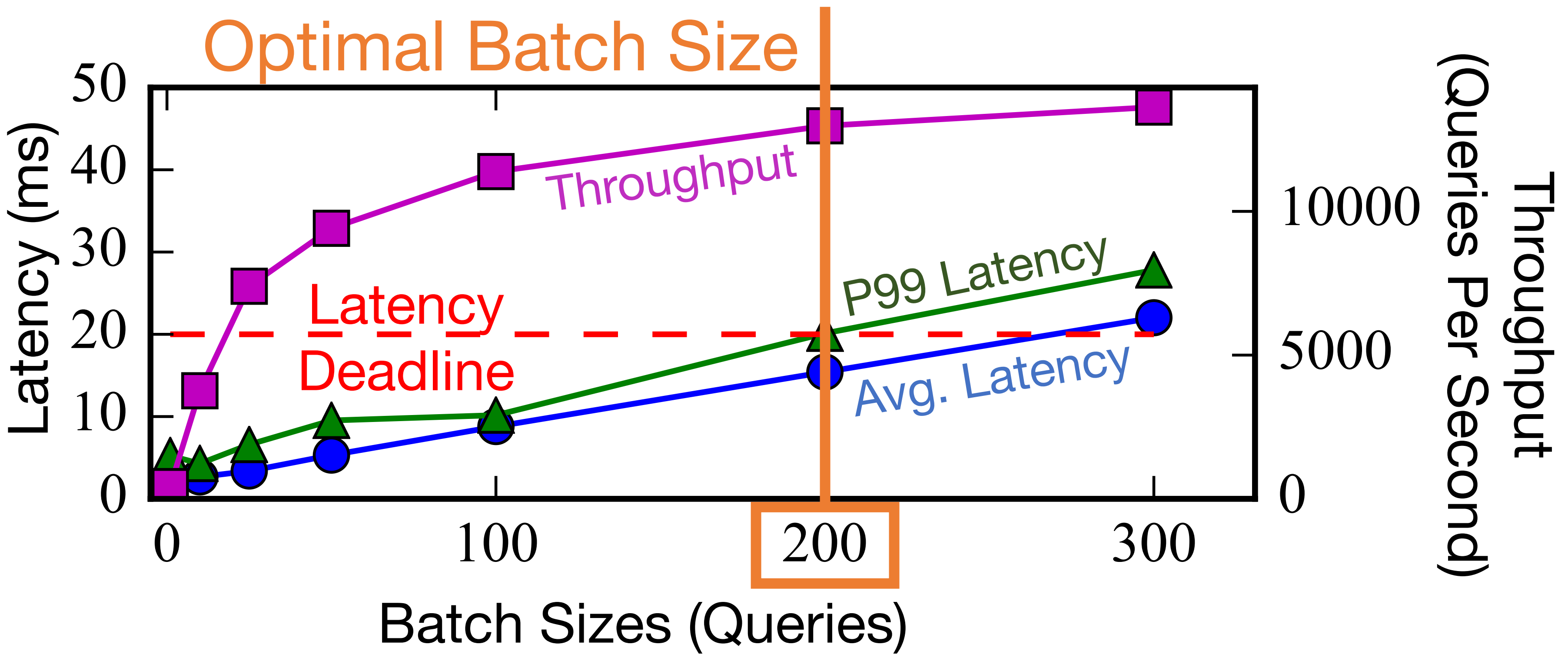
- Optimal batch depends on:
  - hardware configuration
  - model and framework
  - system load

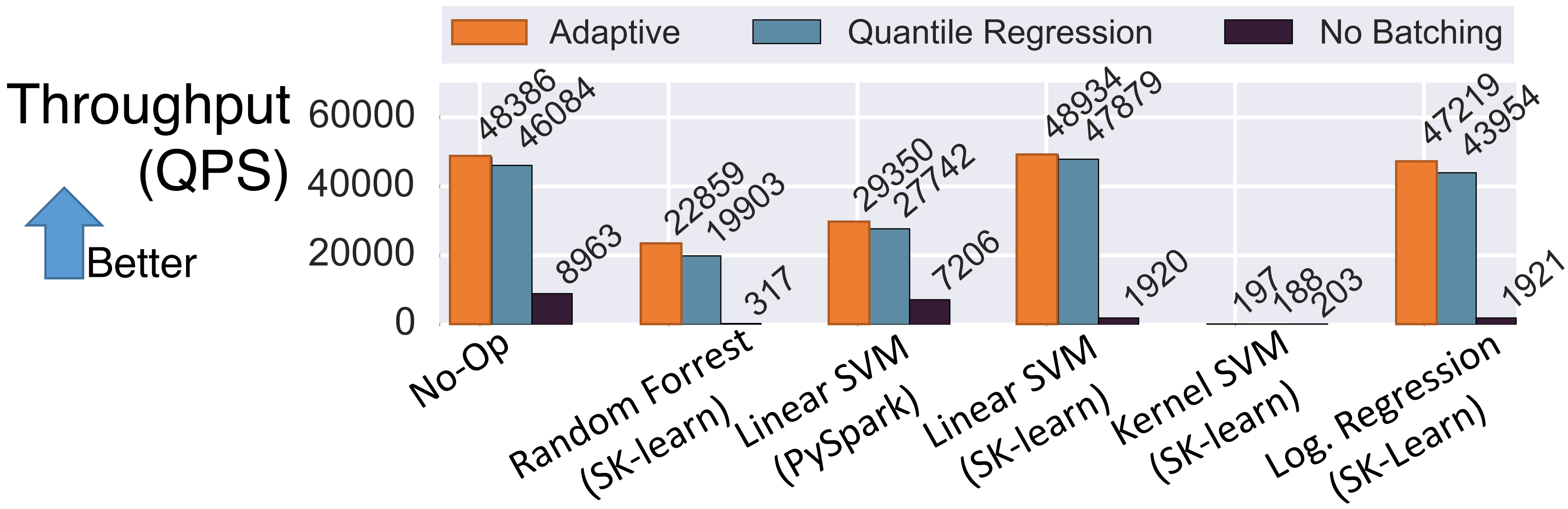
Clipper Solution:

*be as **slow** as **allowed**…*

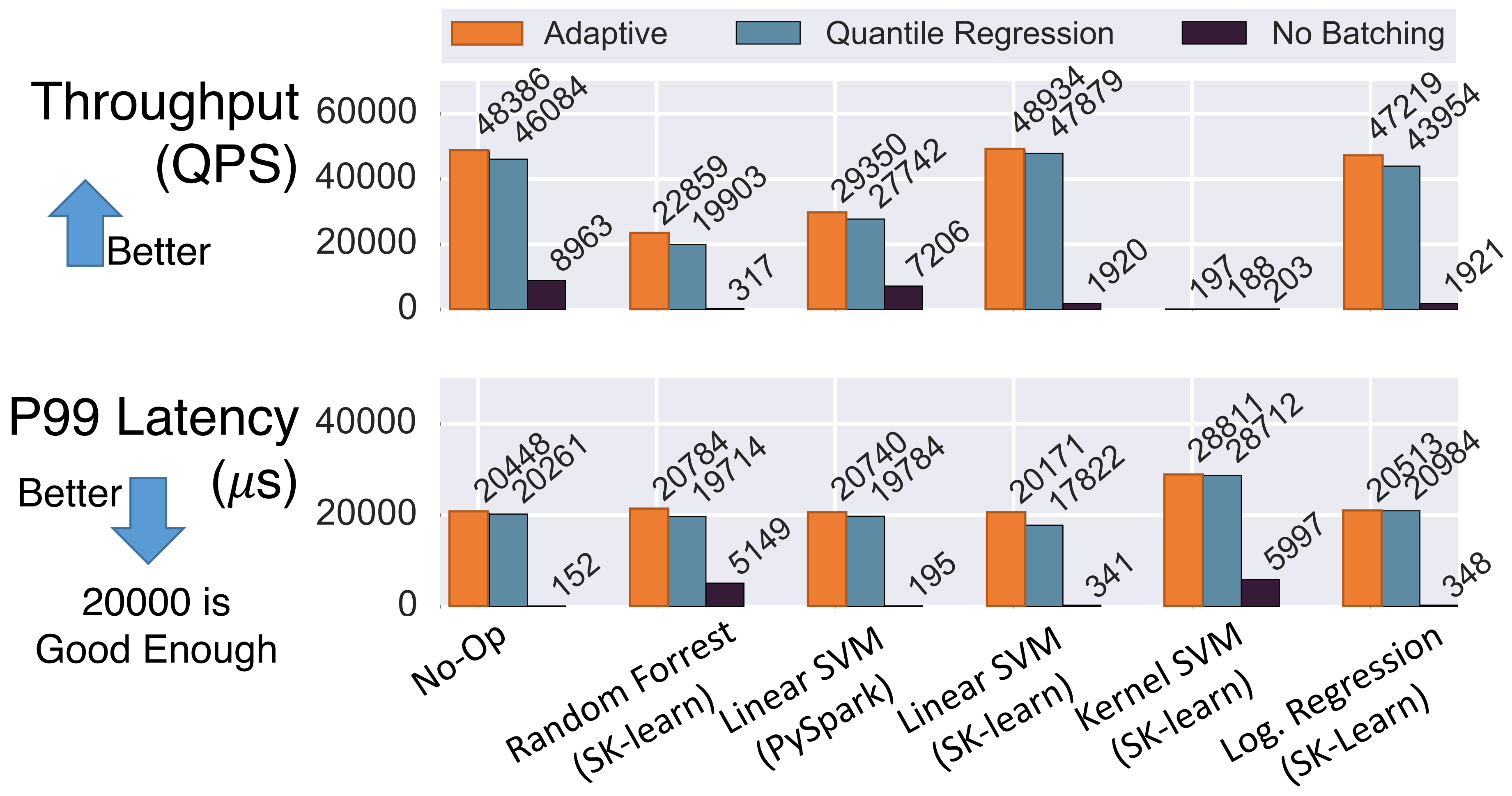
- Application specifies latency objective
- Clipper uses TCP-like tuning algorithm to **increase latency** up to the objective

Tensor Flow Conv. Net (GPU)



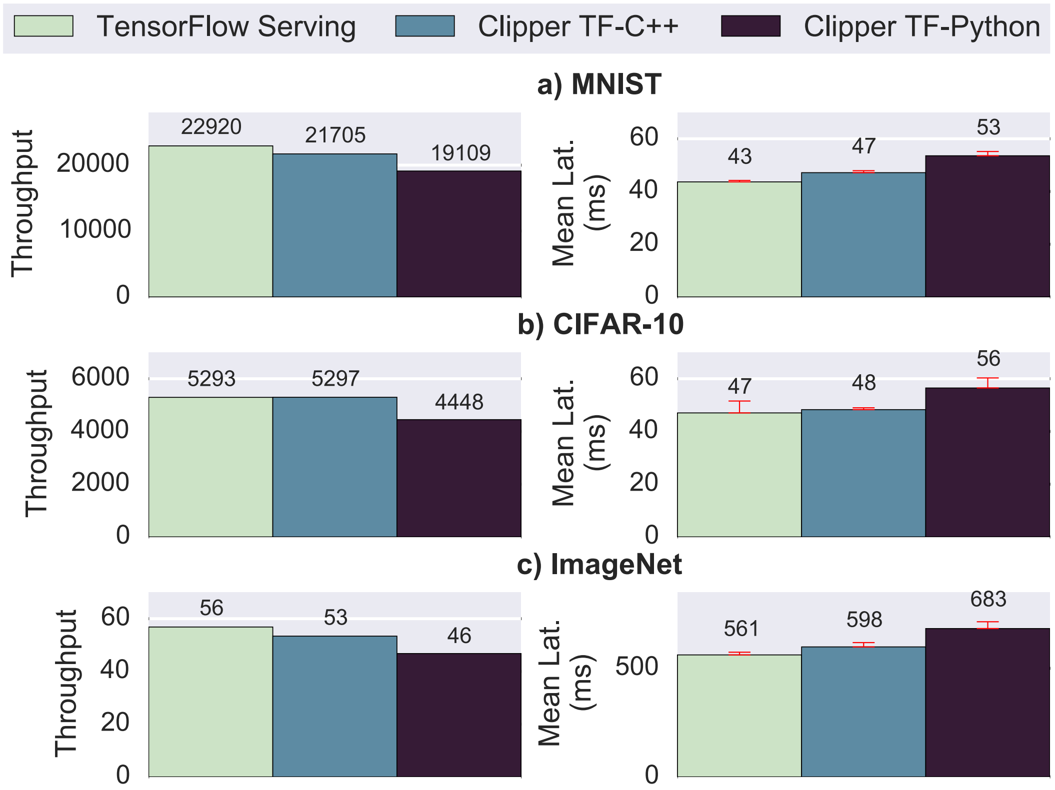


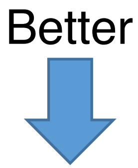
Adaptive
Quantile Regression
No Batching
Throughput (QPS)
Better
60000
40000
20000
0
48386
46084
8963
22859
19903
317
29350
27742
7206
48934
47879
1920
197
188
203
47219
43954
1921
No-Op
Random Forrest (SK-learn)
Linear SVM (PySpark)
Linear SVM (SK-learn)
Kernel SVM (SK-learn)
Log. Regression (SK-Learn)



Adaptive
Quantile Regression
No Batching
Throughput (QPS)
Better
60000
40000
20000
0
48386
46084
8963
22859
19903
317
29350
27742
7206
48934
47879
1920
197
188
203
47219
43954
1921
P99 Latency (μs)
Better
20000 is Good Enough
40000
20000
0
20448
20261
152
20784
19714
5149
20740
19784
195
20171
17822
341
28811
28712
5997
20513
20984
348
No-Op
Random Forrest (SK-learn)
Linear SVM (PySpark)
Linear SVM (SK-learn)
Kernel SVM (SK-learn)
Log. Regression (SK-Learn)

Overhead of modularity?



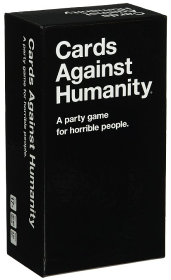


40000 is Good Enough

The decoupled Clipper architecture can be as fast as the in-process approach adopted by TensorFlow-Serving

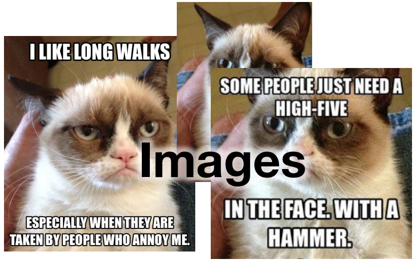
Approximate Caching to Reduce Latency

- Opportunity for caching



Popular items may be evaluated frequently

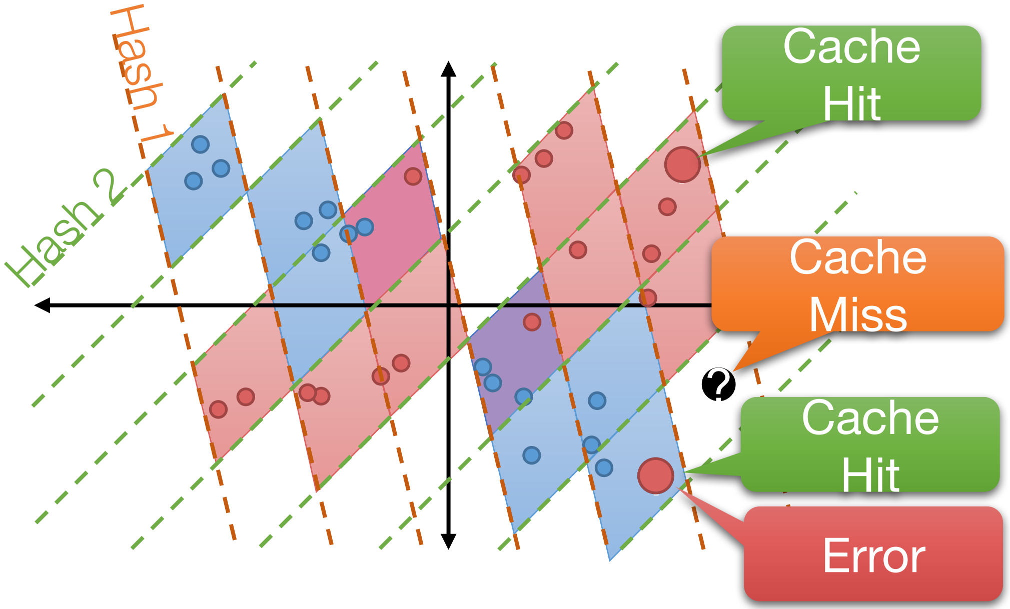
- Need for **approximation**



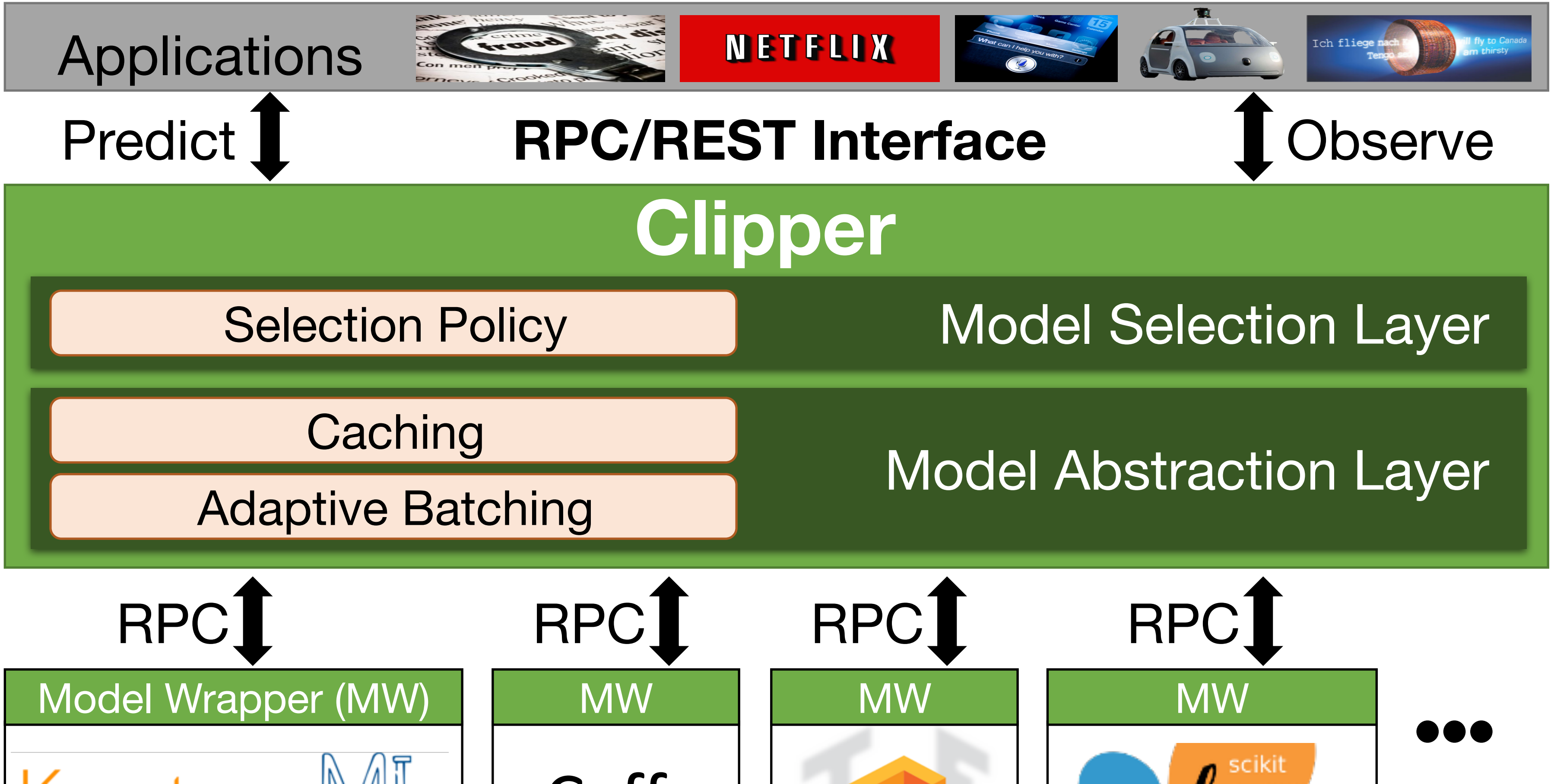
High Dimensional and continuous valued queries have low cache hit rate.

Clipper Solution: *Approximate Caching*

apply *locality sensitive hash functions*



Clipper Architecture



Selection Policy

Model Selection Layer

Goal:

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

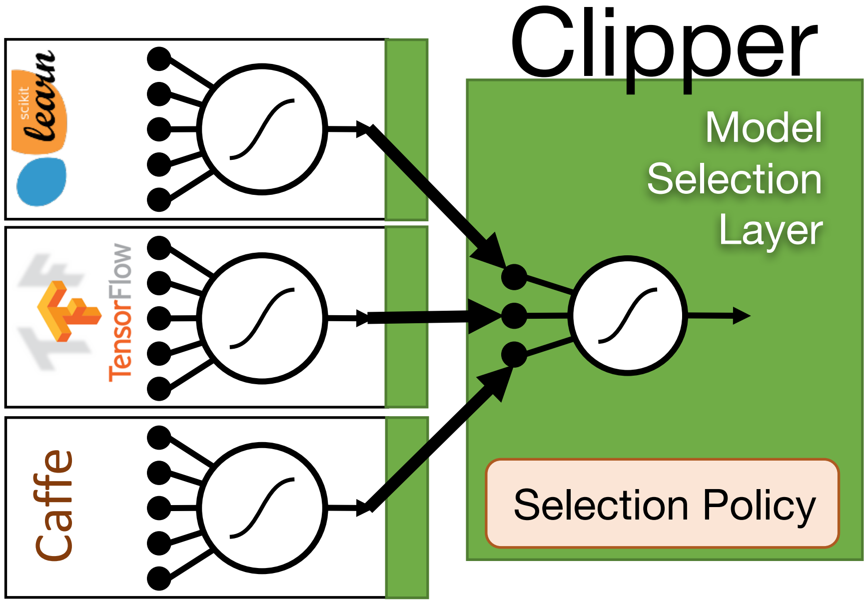
Incorporate feedback in real-time to achieve:

- **robust predictions** by adaptively combining predictions from multiple models and frameworks
- **online learning** and **personalization** by selecting and personalizing **predictions** in response to feedback

Model Selection Policy

Improves prediction **accuracy** by:

- **Combining predictions** from multiple frameworks
  - Ensemble methods
- Incorporate real-time **feedback**
  - Personalized ensembles
  - Bandit algorithms
- Estimates **confidence** of predictions
  - Agreement between models



Ensemble Prediction **Accuracy** (ImageNet)

System	Model	Error Rate	#Errors
Caffe	VGG	13.05%	6525
Caffe	LeNet	11.52%	5760
Caffe	ResNet	9.02%	4512
TensorFlow	Inception v3	6.18%	3088

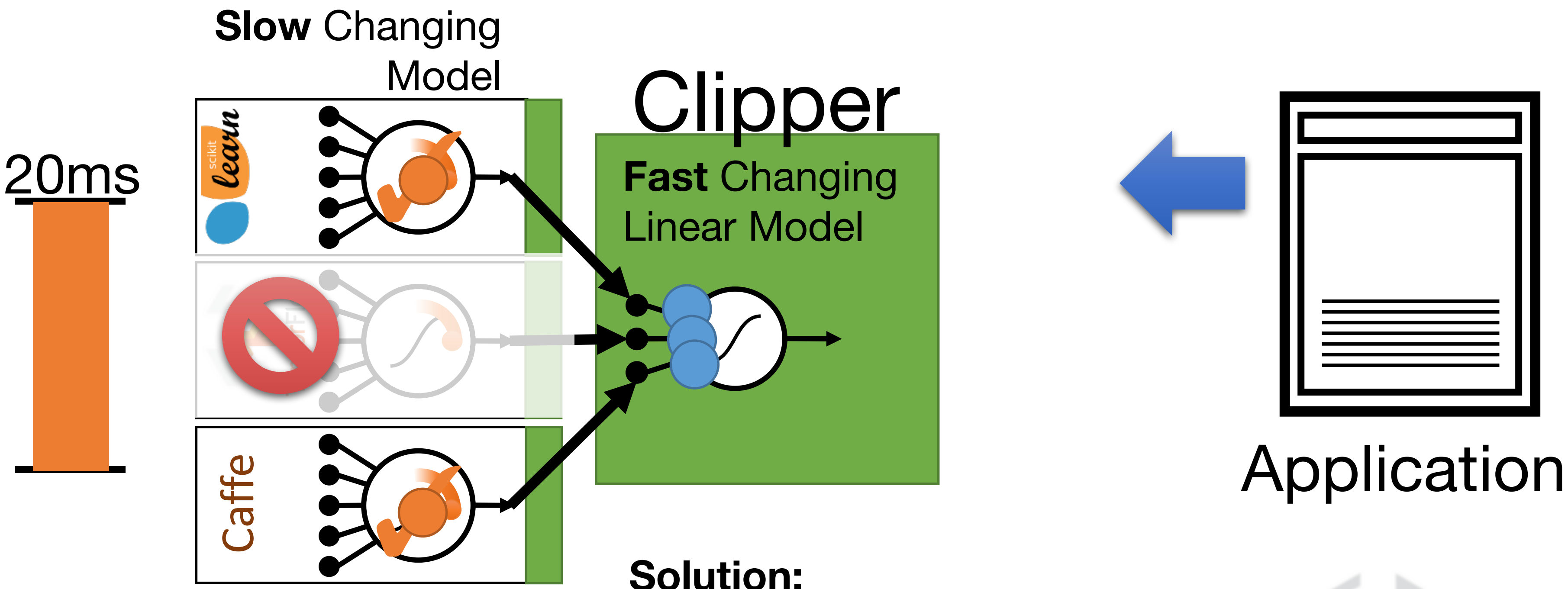
sequence of pre-trained models

Ensemble Prediction **Accuracy** (ImageNet)

System			rrors
Caffe			6525
Caffe			5760
Caffe	ResNet	9.02%	4512
TensorFlow	Inception v3	6.18%	3088
Clipper	**Ensemble**	**5.86%**	**2930**

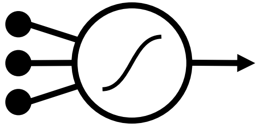
5.2% relative improvement in prediction accuracy!

Ensemble Methods Create Stragglers



Solution:
Replace missing prediction with an estimator ➡ E[TensorFlow (x)]

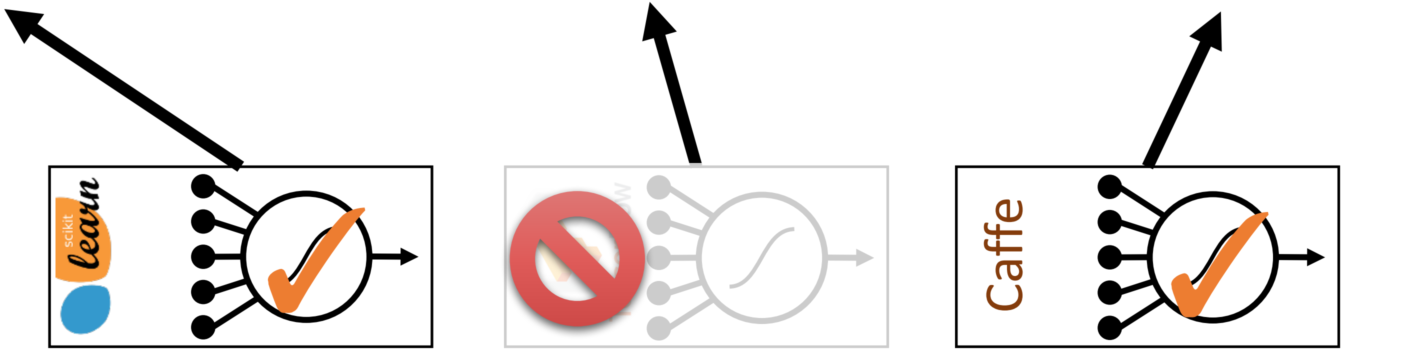
Anytime Predictions



Fast Changing Model

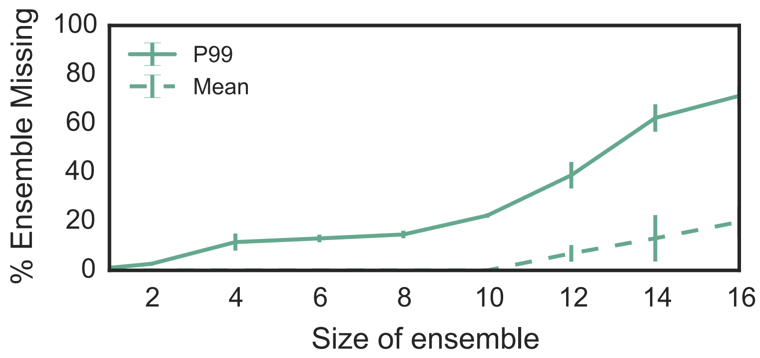
$$w_{\text{scikit}}\, f_{\text{scikit}}(x) \; + \; w_{\text{TF}}\; \mathbb{E}_X\left[f_{\text{TF}}(X)\right] + \; w_{\text{Caffe}} f_{\text{Caffe}}(x)$$

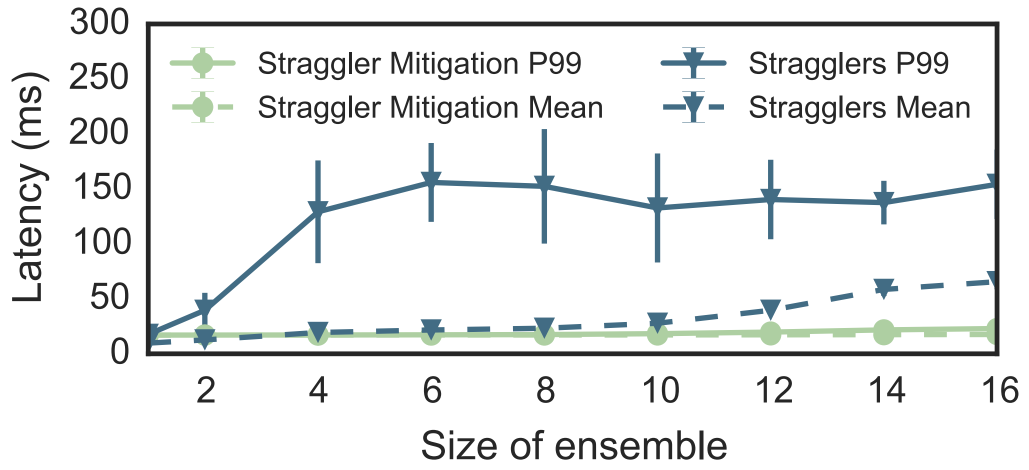
Slow Changing Model

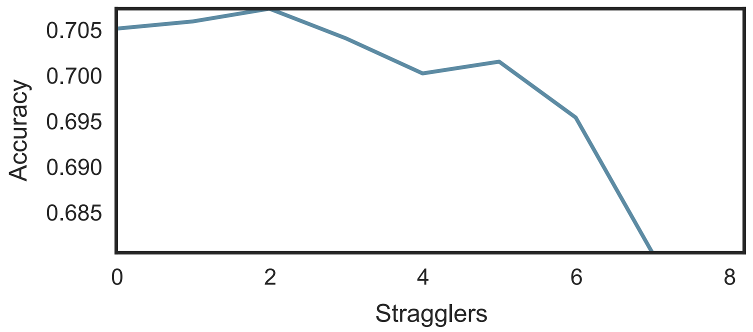


Anytime Predictions

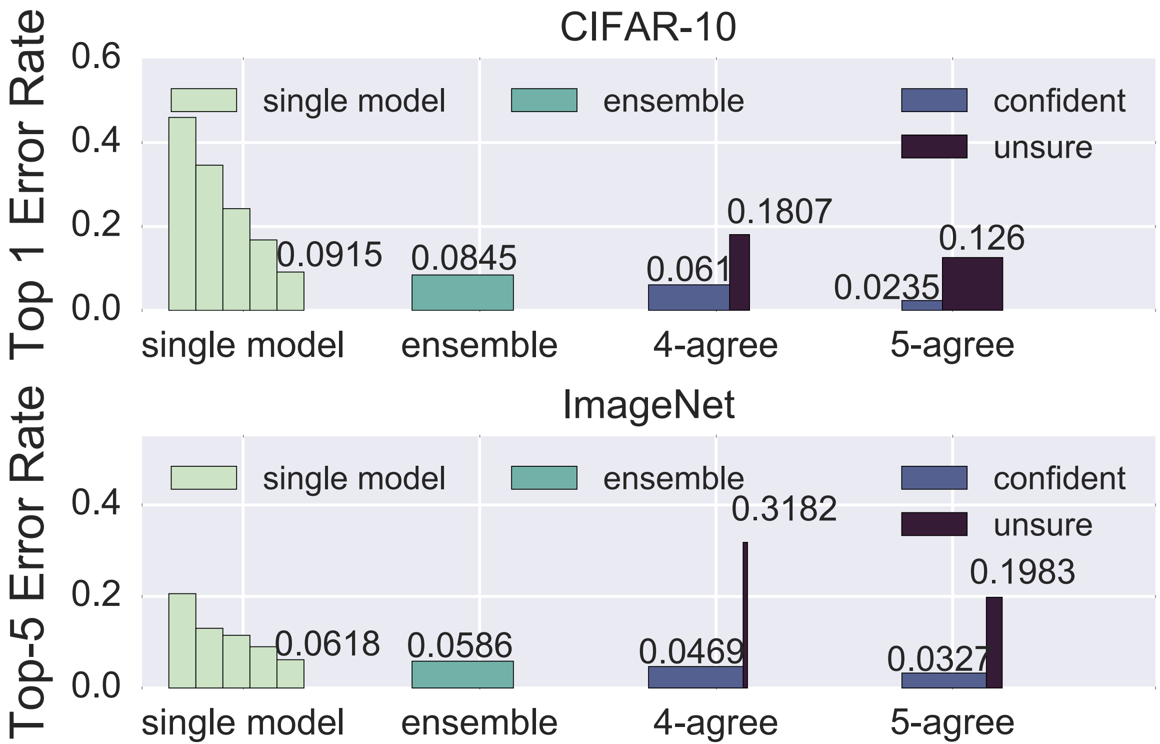
- Tolerates some loss of models
  - Depends heavily on ensemble







Ensemble’s to Estimate Confidence



Clipper

Clipper is a prediction serving system that spans multiple ML Frameworks and is designed to

- ➢ to **simplifying** model serving
- ➢ **bound latency** and **increase throughput**
- ➢ and enable **real-time learning** and **personalization**

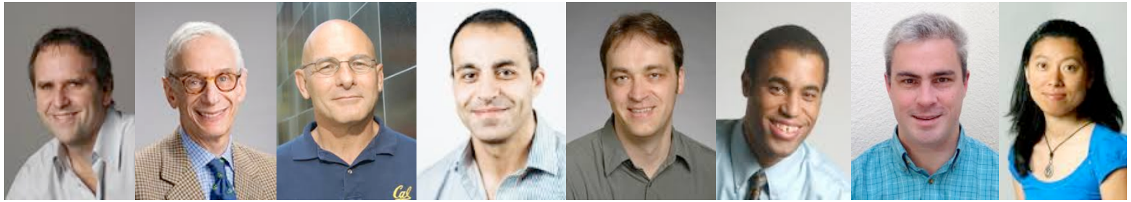
across machine learning frameworks

"Clipper: A Low-Latency Online Prediction Serving System"
https://github.com/ucbrise/clipper (open source)

Ongoing Clipper Subprojects

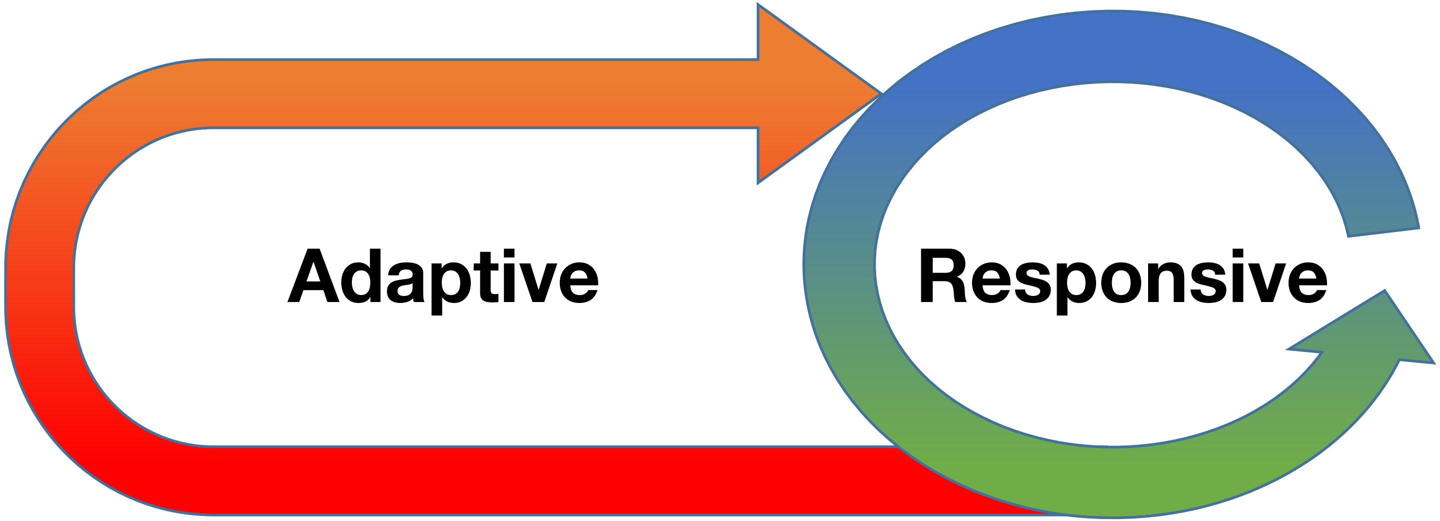
- **Adaptive Batching for Prediction**
  - Leverage internal data-parallelism and hardware acceleration
- **Approximate Caching**
  - Detect “similar” queries and re-use cached predictions
- **Prediction Cascades**
  - Automatically deriving cascades of increasingly GPU intensive models
- **RL/Control**
  - Serving and updating RL policies based on feedback
- **Scheduling and resource allocation**
  - Reduce the need to over-provision for bursty workloads

riselab
UC Berkeley



We are developing new technologies that will enables applications to make low-latency intelligent decision on live data with strong security guarantees.

riselab
UC Berkeley



Secure