Prediction Serving

what happens after learning?

Joseph E. Gonzalez

Asst. Professor, UC Berkeley

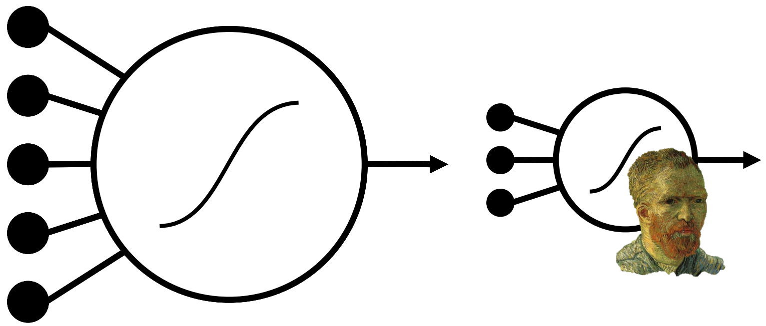
jegonzal@cs.berkeley.edu

Co-founder, Dato Inc.

joseph@dato.com

Outline

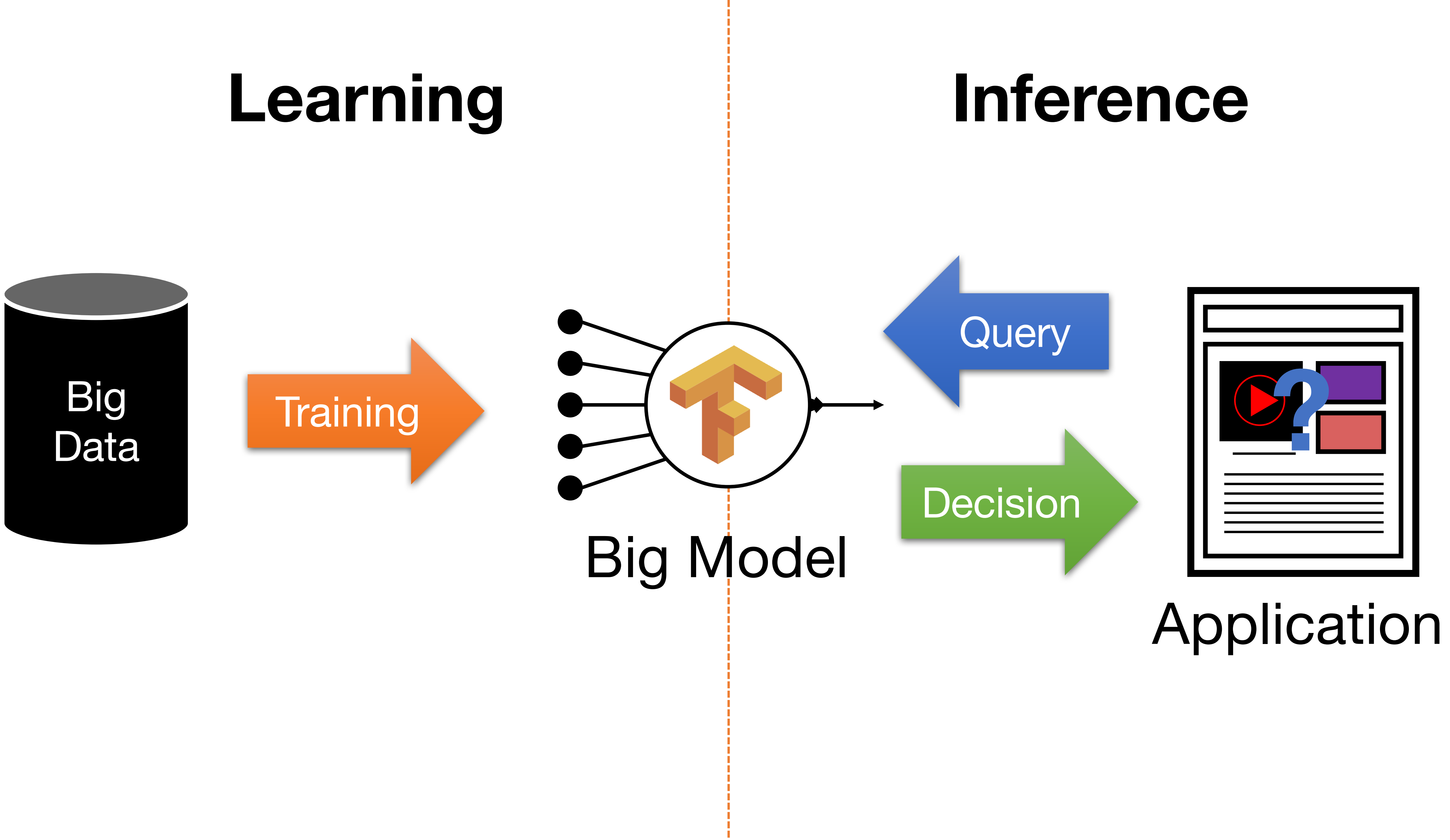
VELOX



Daniel Crankshaw, Xin Wang
Michael Franklin, & Ion Stoica

Learning

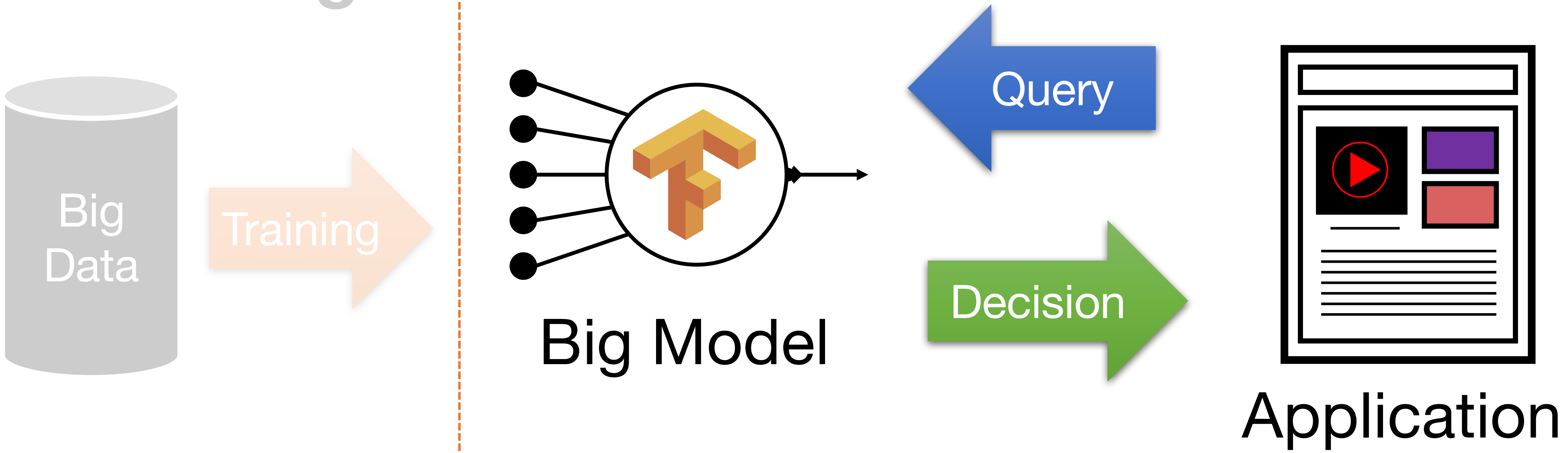
Timescale: minutes to days
Systems: offline and batch optimized
Heavily studied ... major focus of the ***AMPLab***



Learning
Inference
Big
Data
Training
Big Model
Query
Decision
Application

Learning

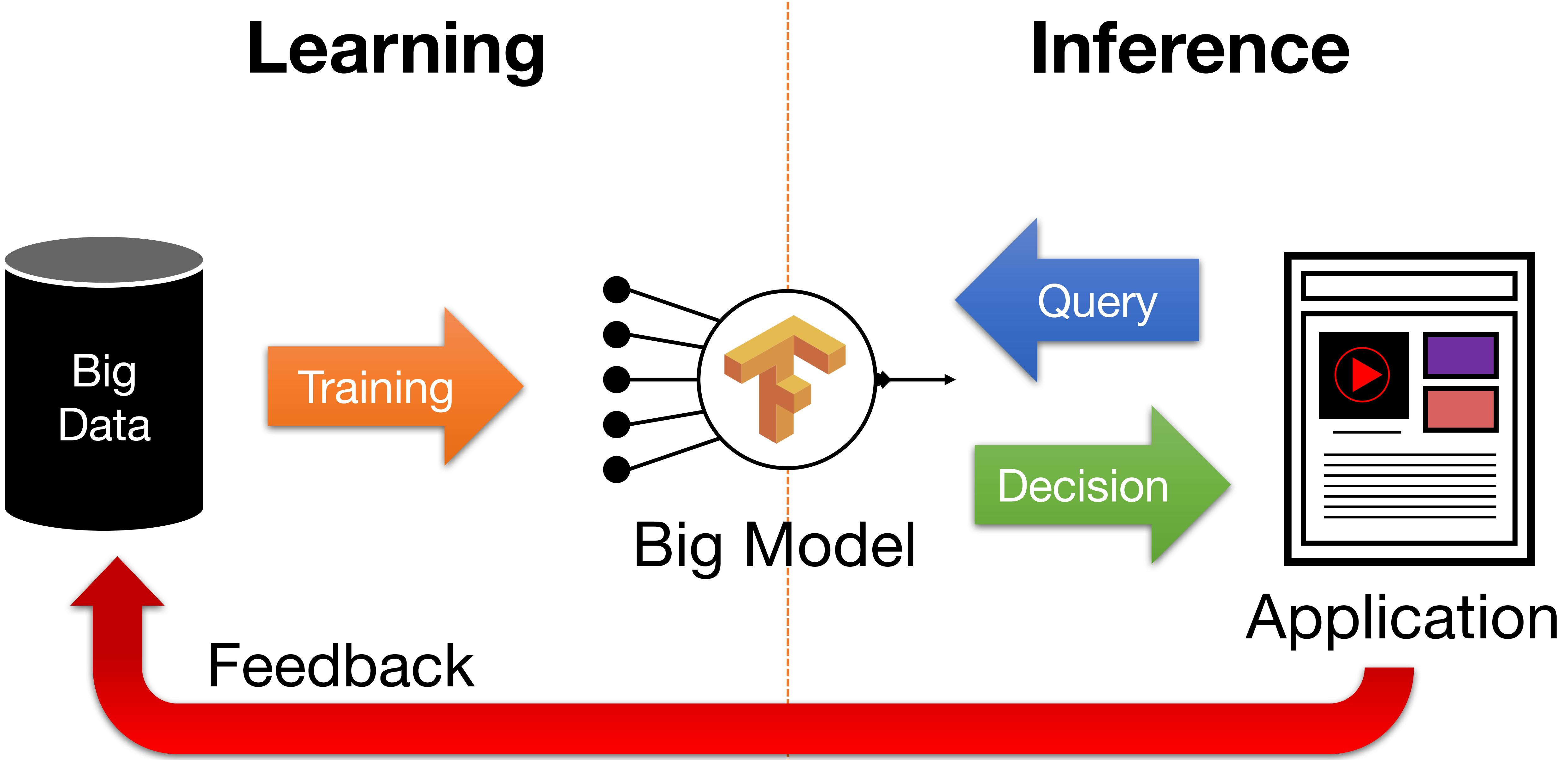
Inference

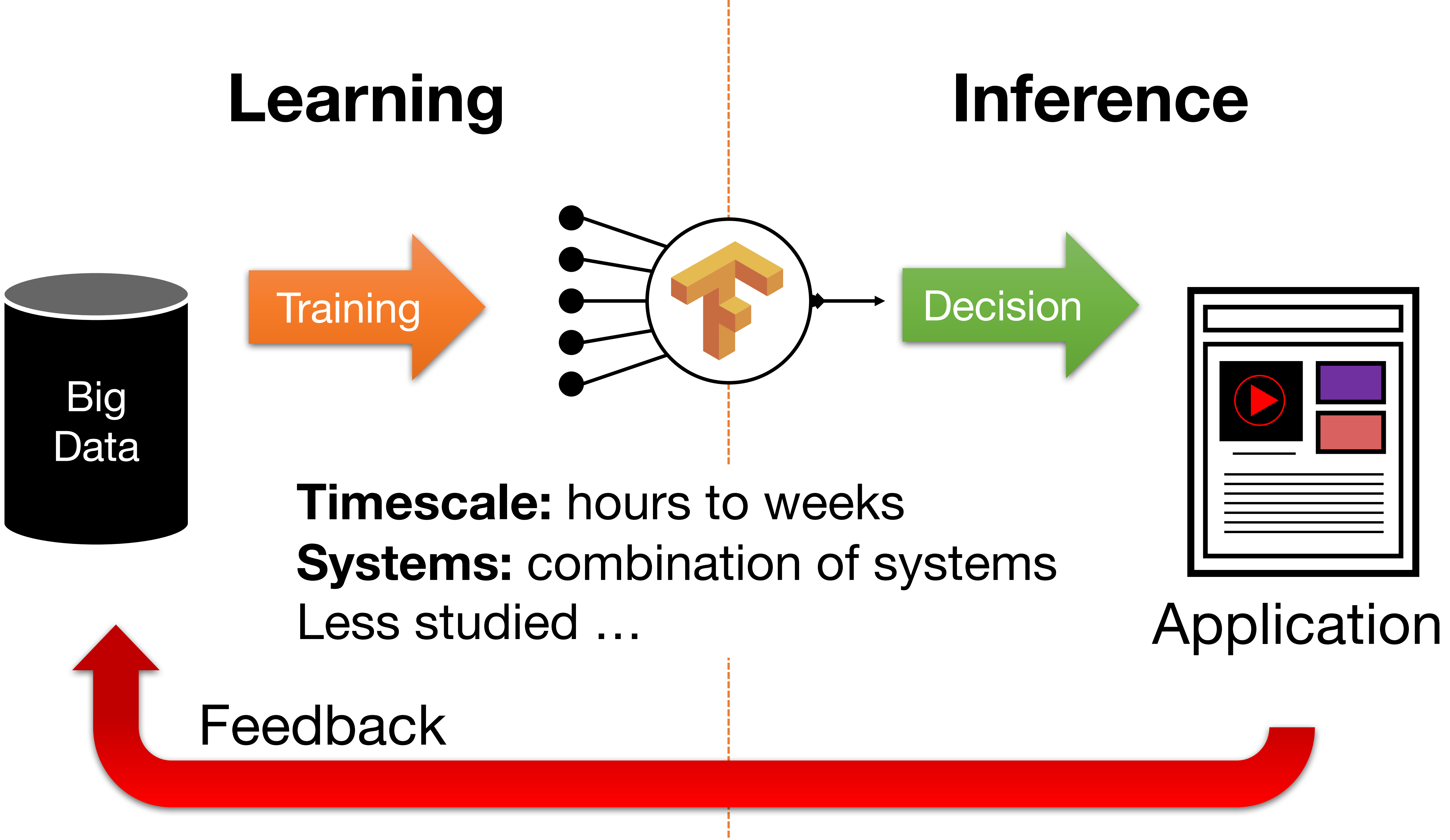


Timescale: ~10 milliseconds

Systems: *online* and *latency* optimized

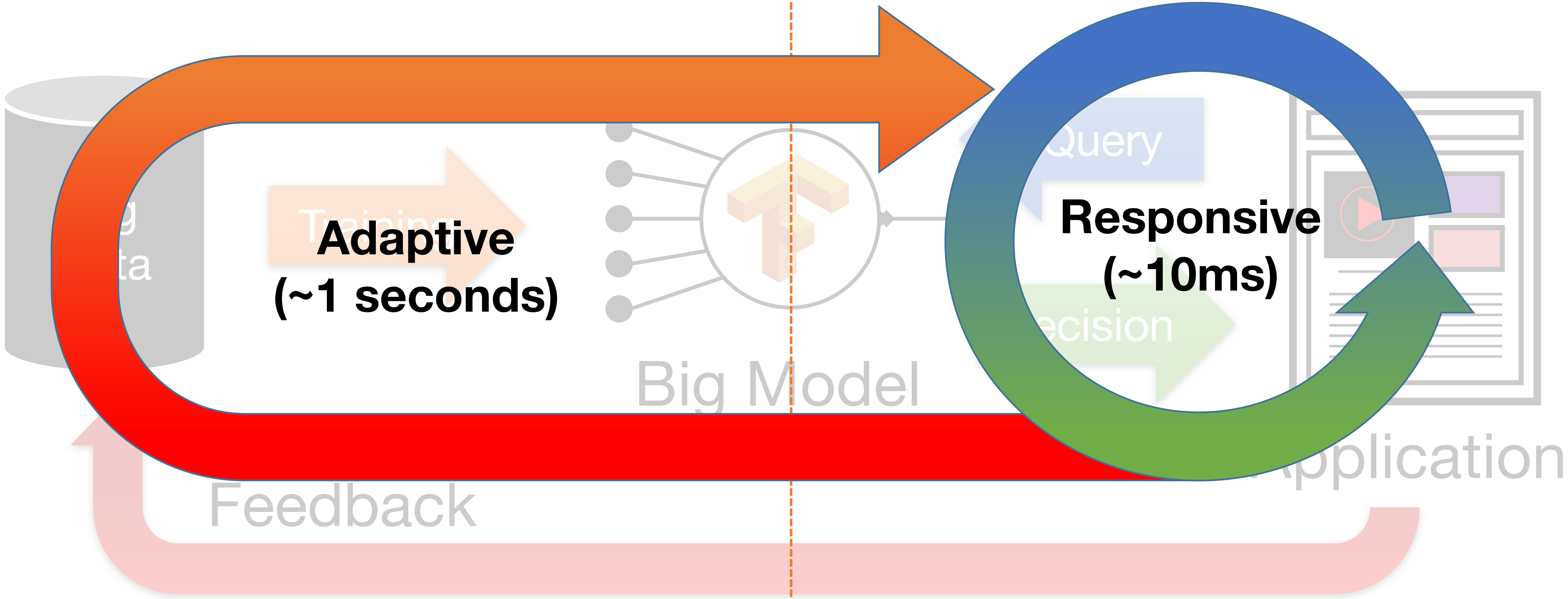
Less studied …


Learning
Inference
Big
Data
Training
Big Model
Query
Decision
Application
Feedback


Learning
Inference
Big Data
Training
Decision
Application
Timescale: hours to weeks
Systems: combination of systems
Less studied …
Feedback

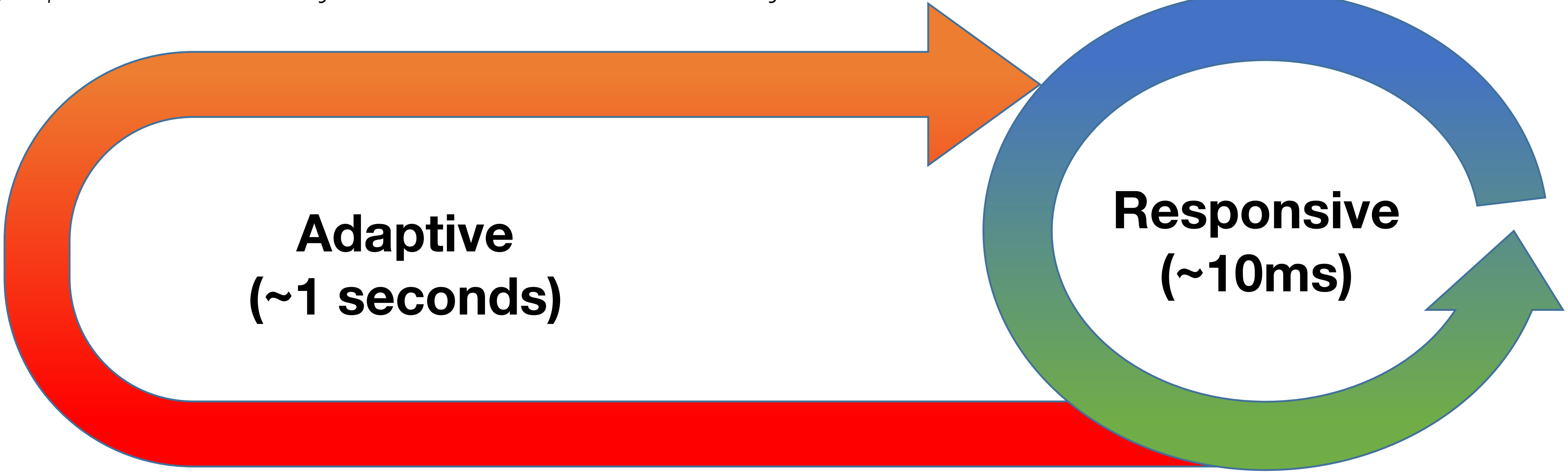
Learning

Inference



VELOX Model Serving System [CIDR'15]

Daniel Crankshaw, Peter Bailis, Haoyuan Li, Zhao Zhang,
Joseph Gonzalez, Michael J. Franklin, Ali Ghodsi, and Michael I. Jordan

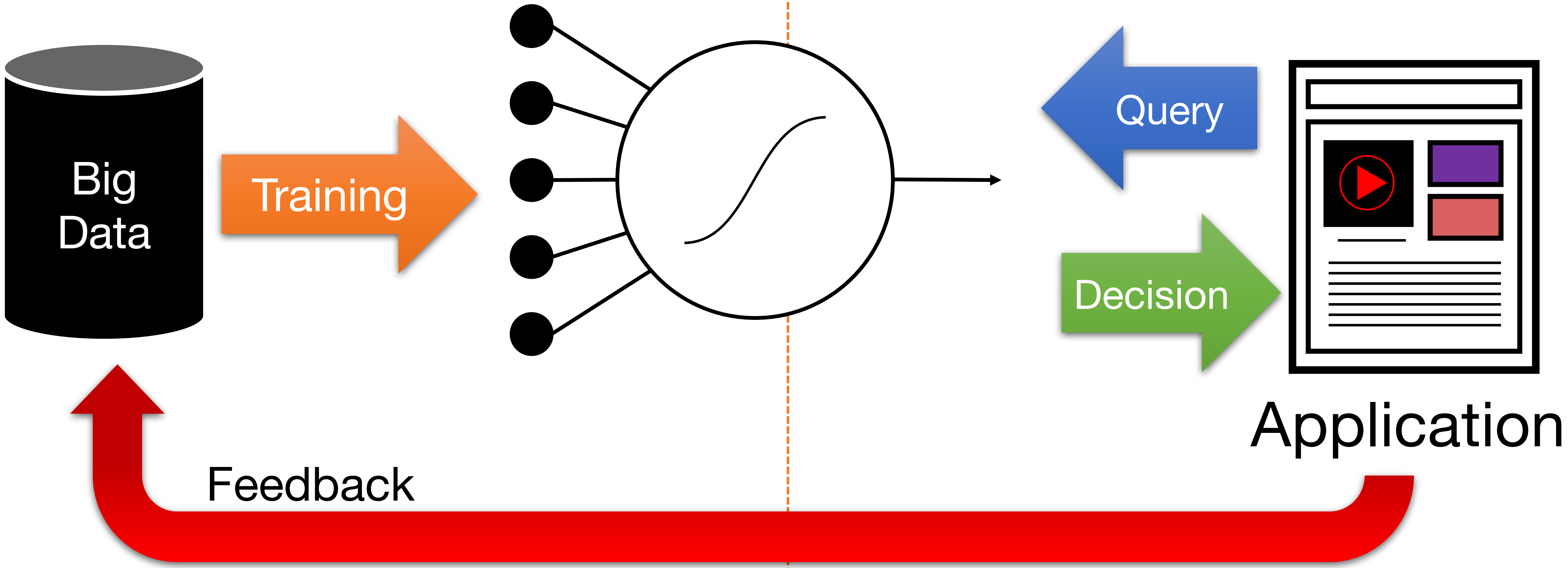


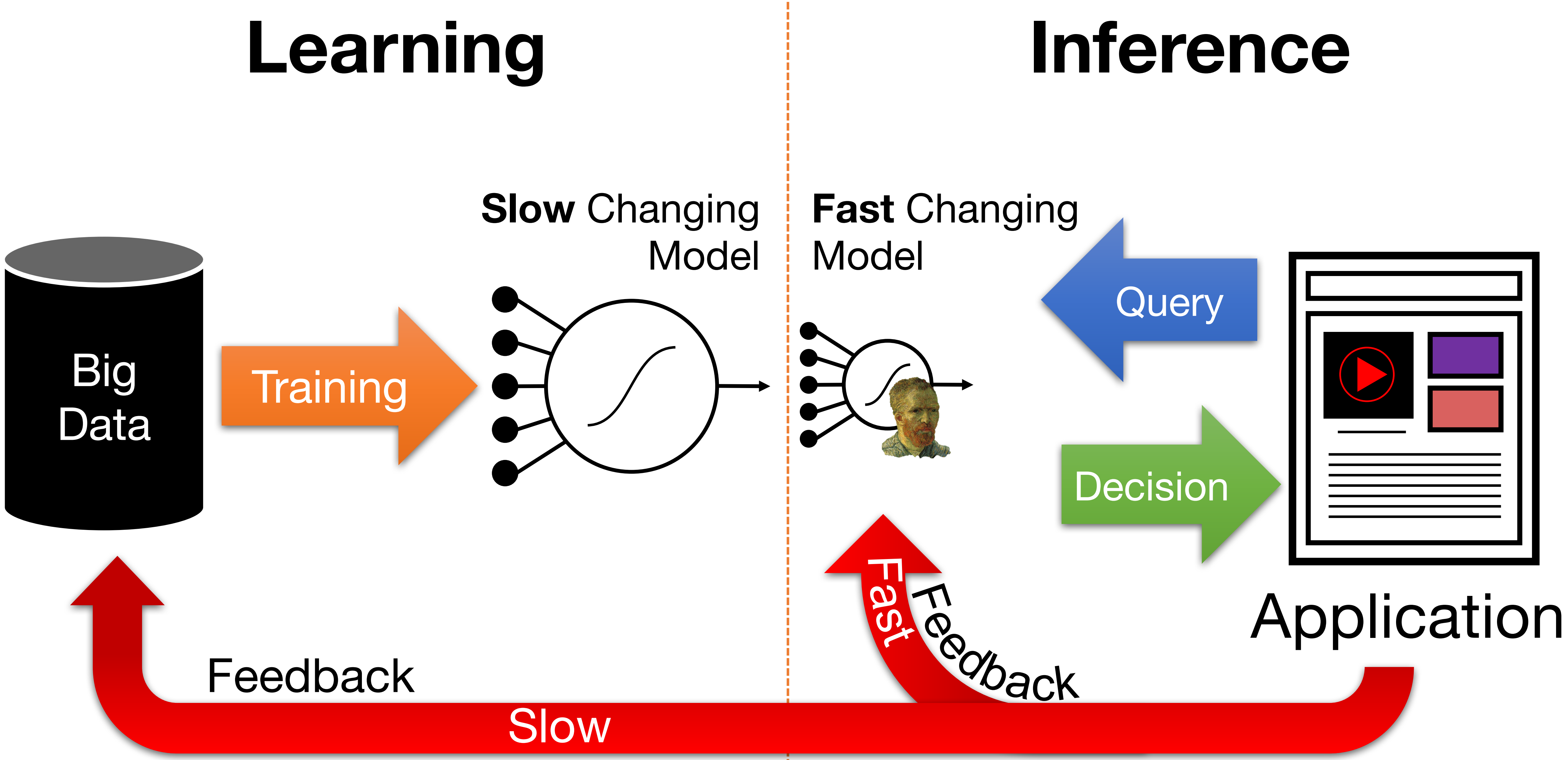
Key Insight:

Decompose models into fast and slow changing components

Learning

Inference




Learning
Inference
Slow Changing Model
Fast Changing Model
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:

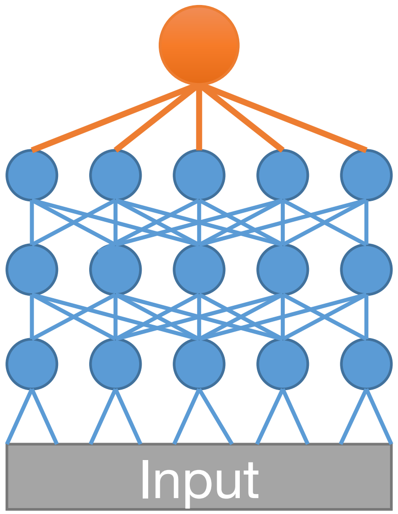
- Simple to train + more robust model
- Address rapidly changing user statistics

Common modeling structure

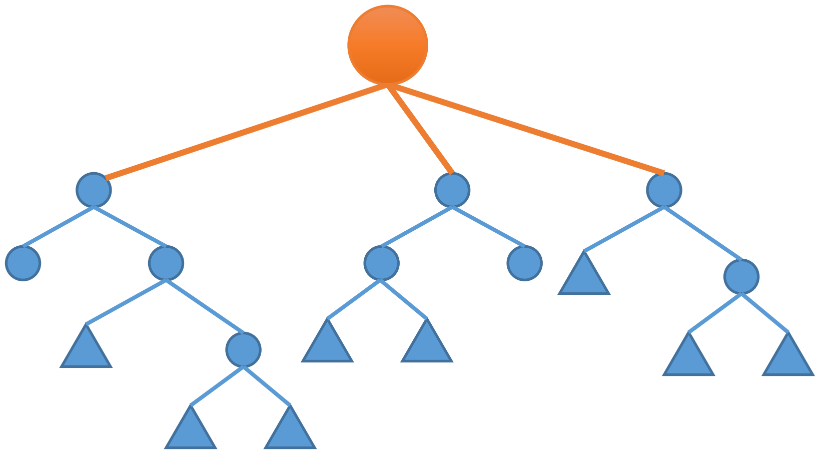
$$f(x;\theta)^T w_u$$

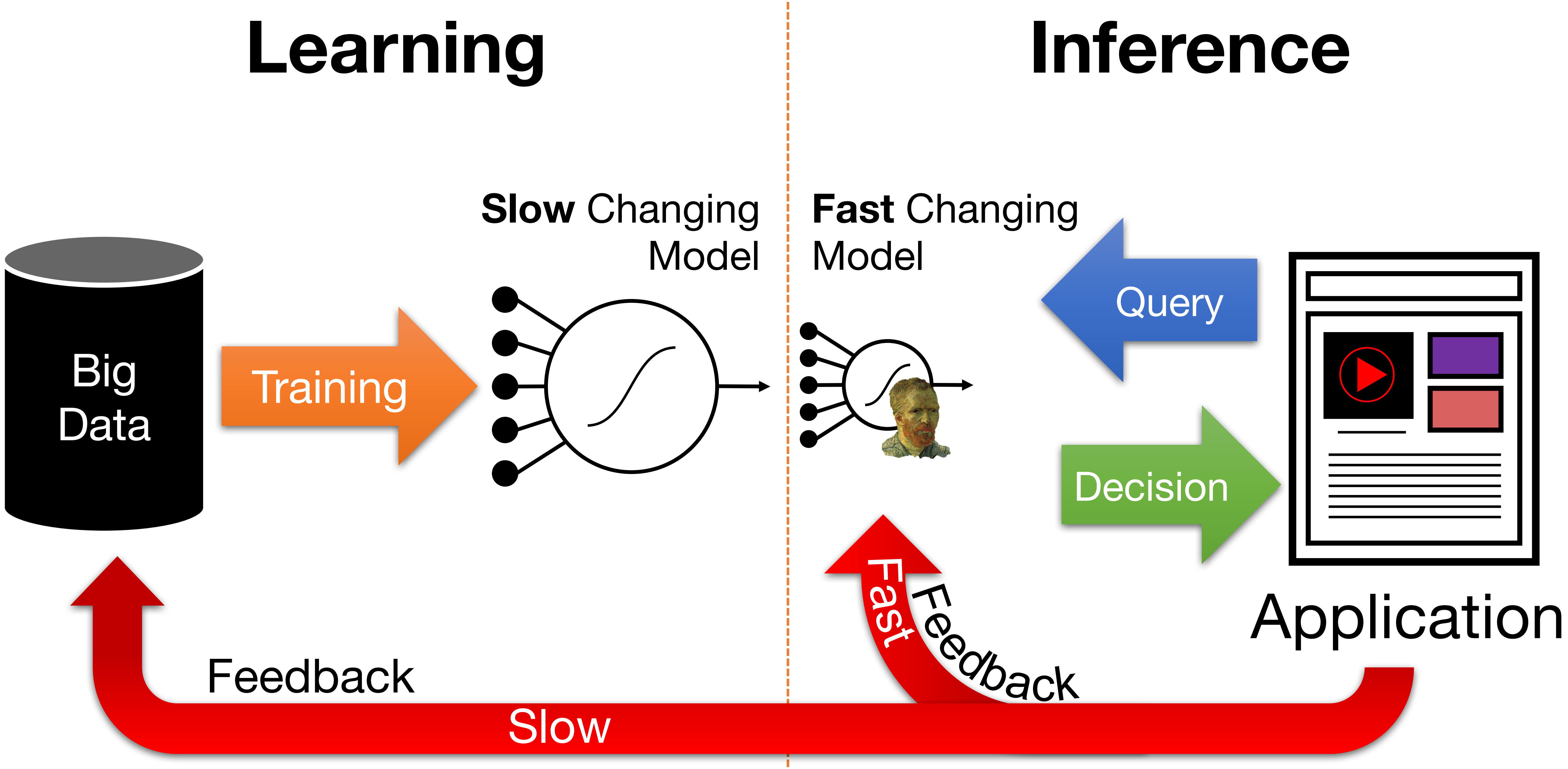
Matrix Factorization

Deep Learning

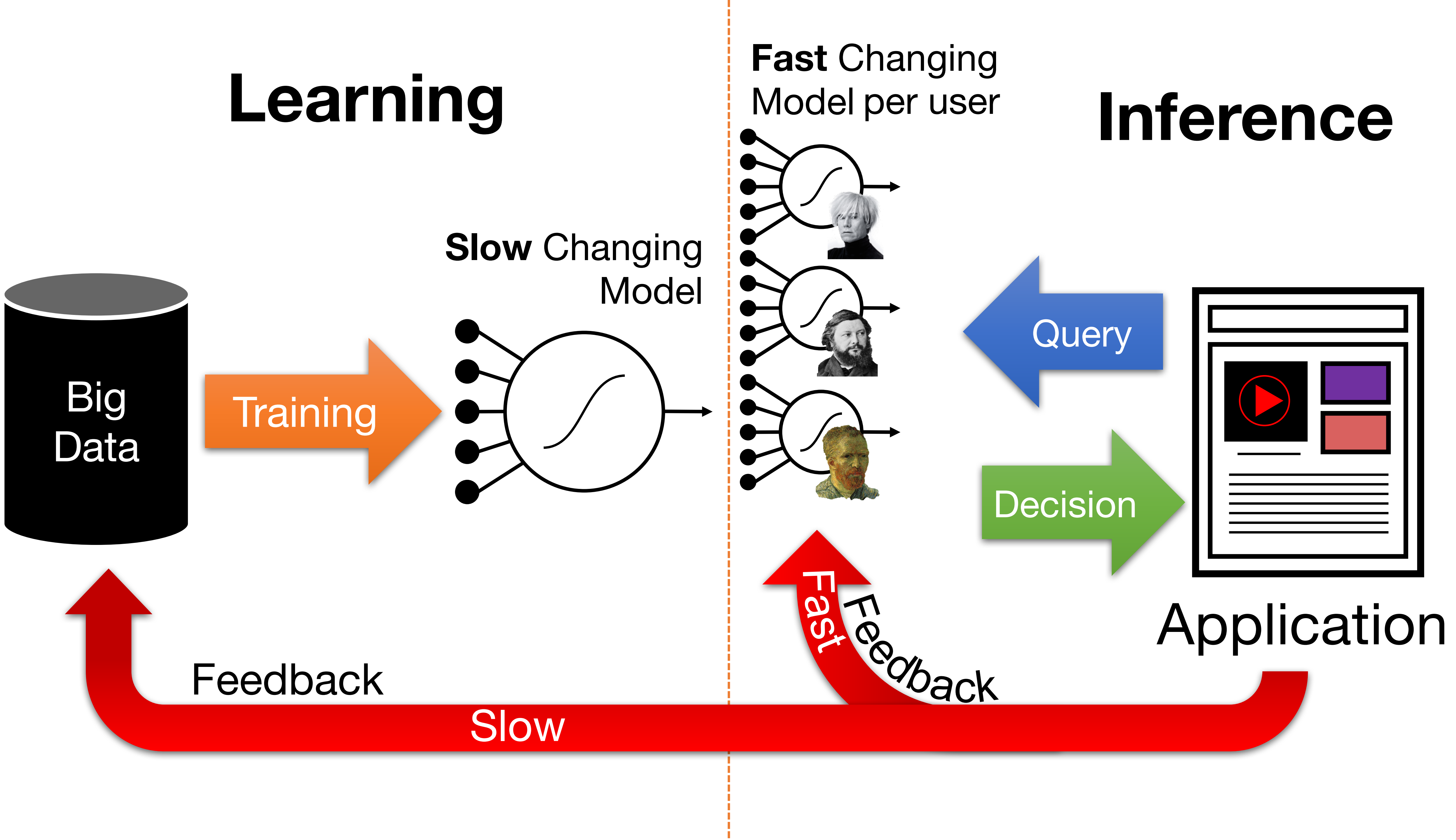


Ensemble Methods

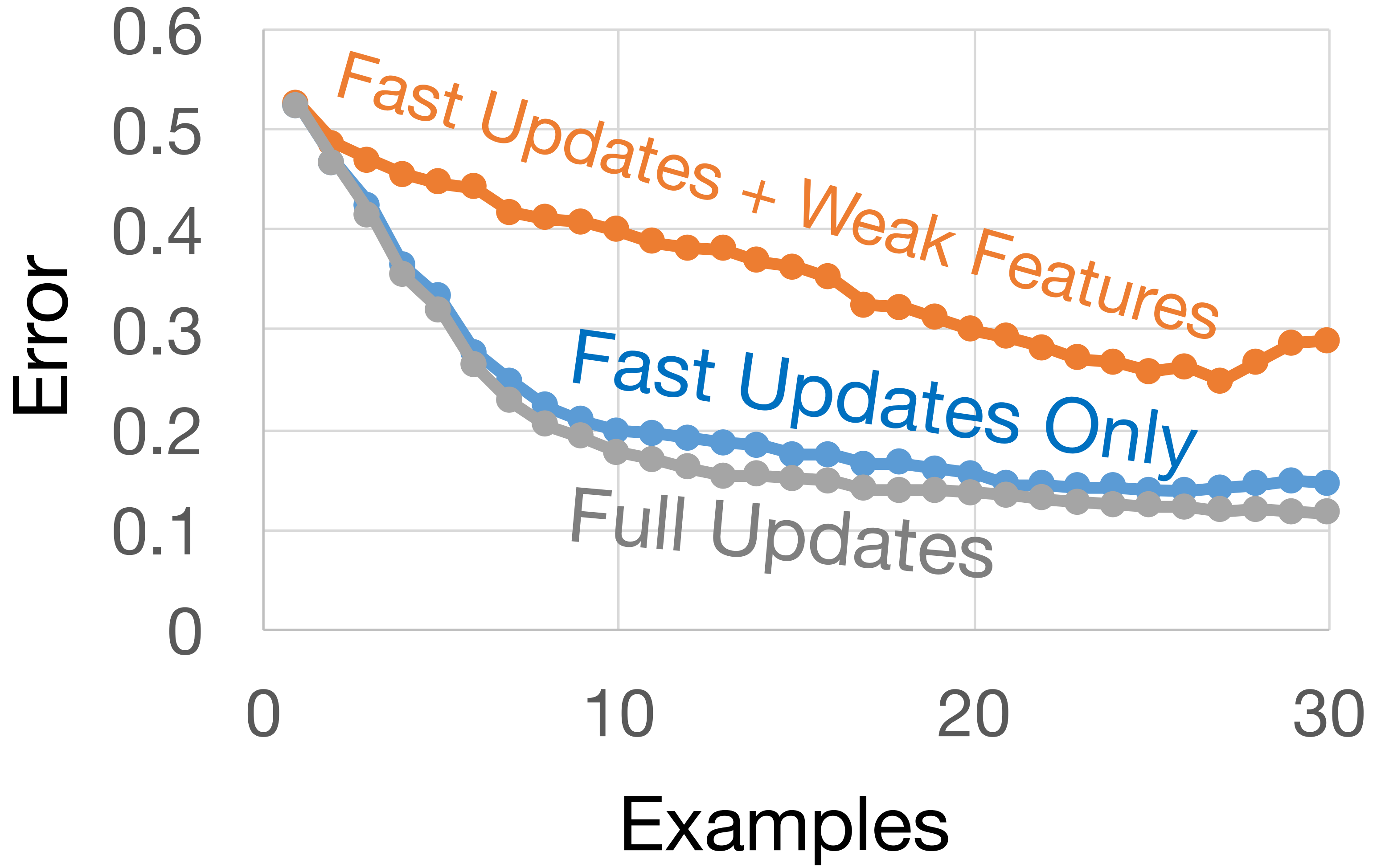




Learning
Inference
Slow Changing Model
Fast Changing Model
Big Data
Training
Query
Decision
Application
Feedback
Slow
Fast Feedback


Learning
Inference
Fast Changing
Model per user
Slow Changing
Model
Big
Data
Training
Query
Decision
Application
Feedback
Slow
Fast
Feedback

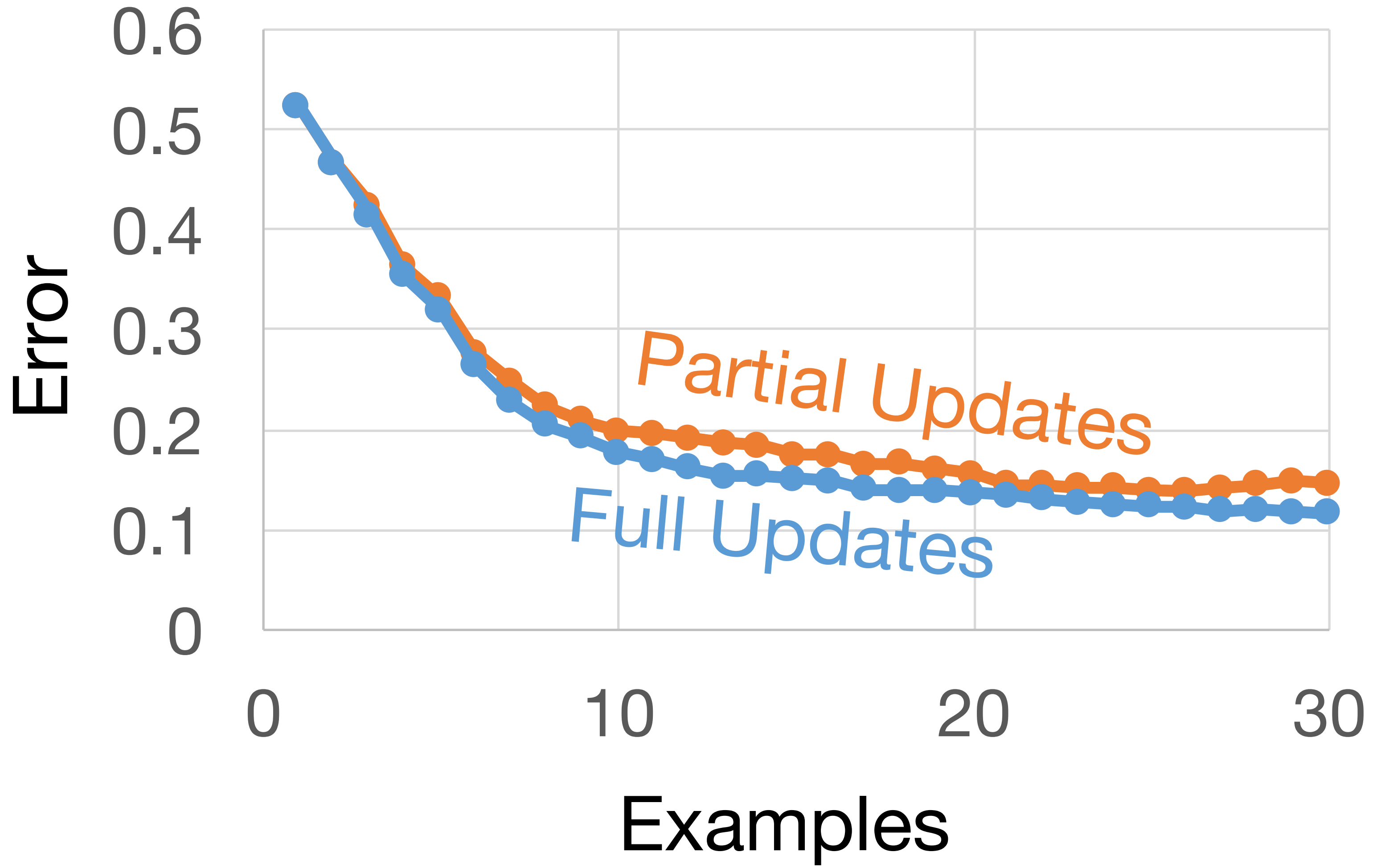
Velox Online Learning for Recommendations
(20-News Groups)



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

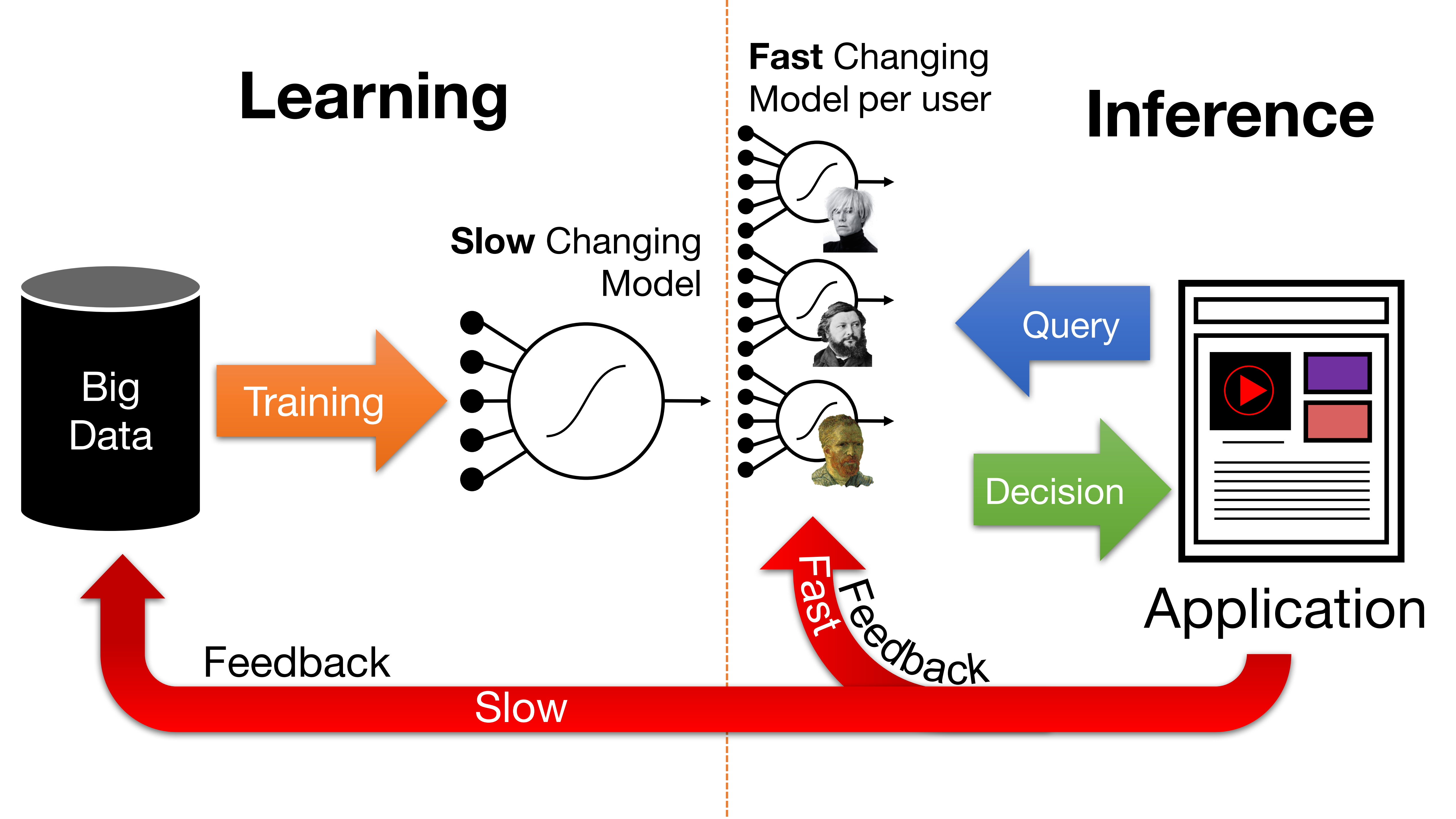
>4 orders-of-magnitude ***faster adaptation*** *given sufficient offline training data*

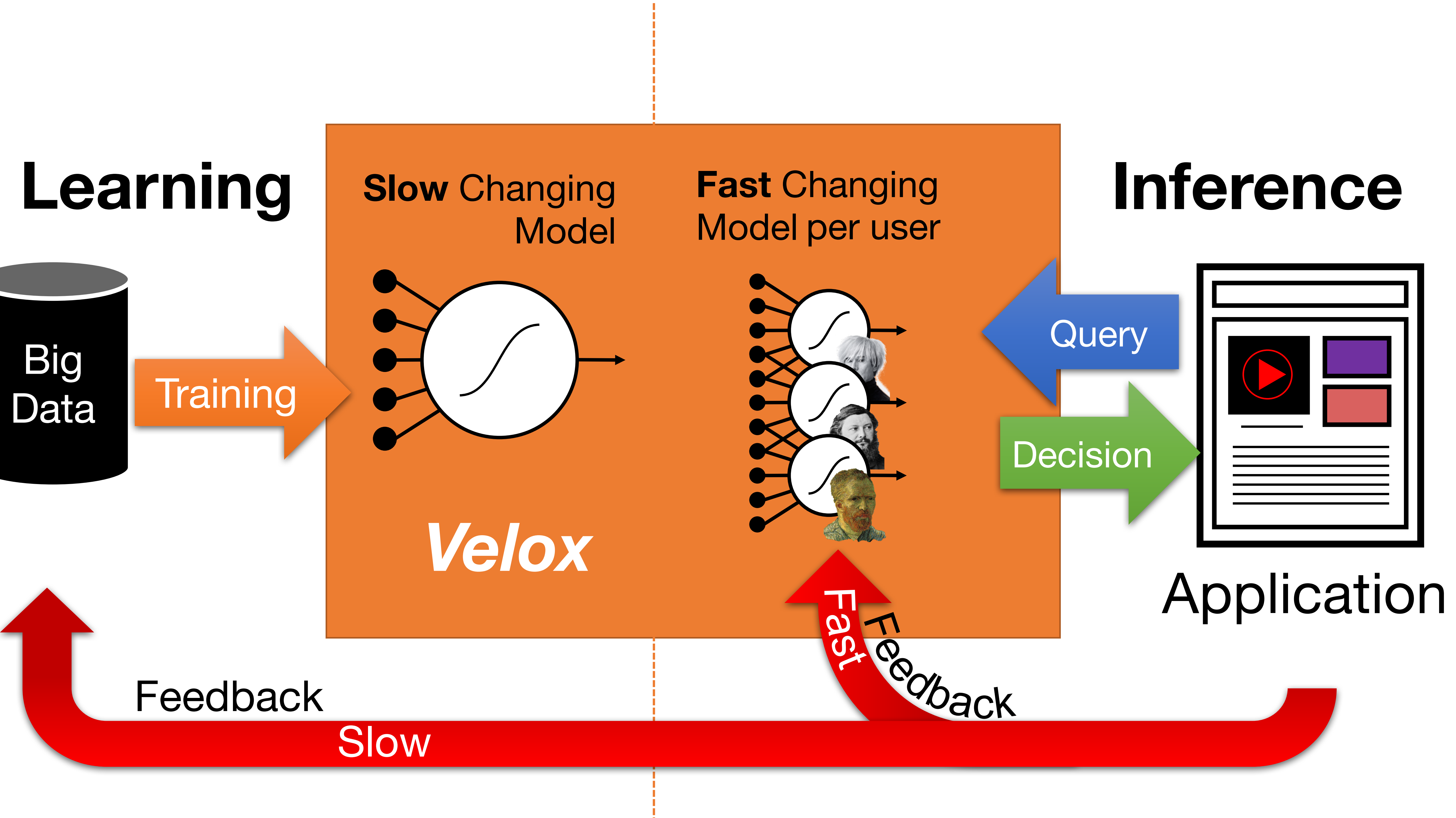
Velox Online Learning for Recommendations
(20-News Groups)



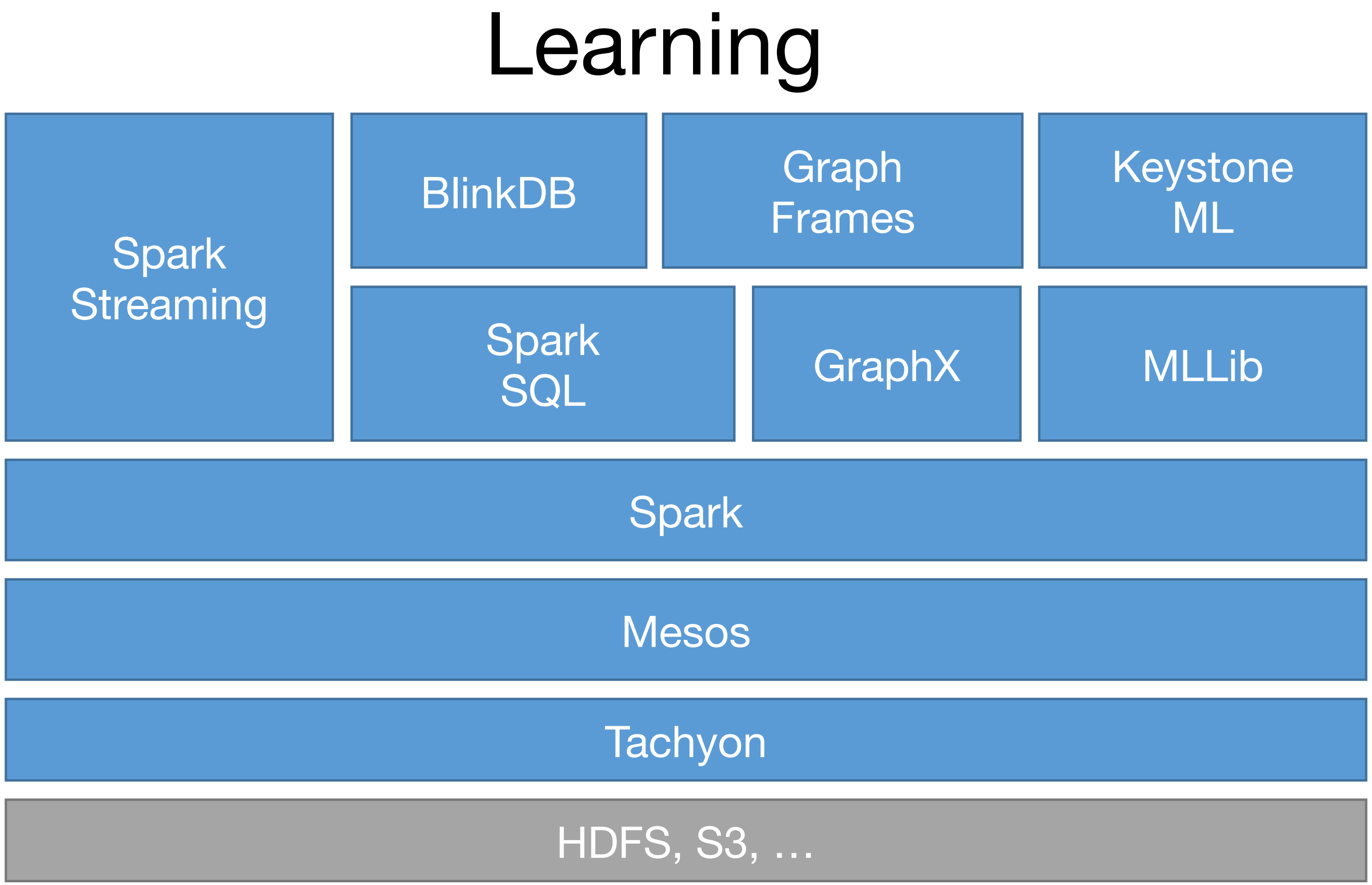
Partial Updates: *0.4 ms*
Retraining: *7.1 seconds*

>4 orders-of-magnitude
faster adaptation


Learning
Inference
Fast Changing
Model per user
Slow Changing
Model
Big
Data
Training
Query
Decision
Application
Feedback
Slow
Fast
Feedback


Learning
Slow Changing Model
Fast Changing Model per user
Inference
Big Data
Training
Query
Decision
Velox
Application
Fast Feedback
Feedback
Slow

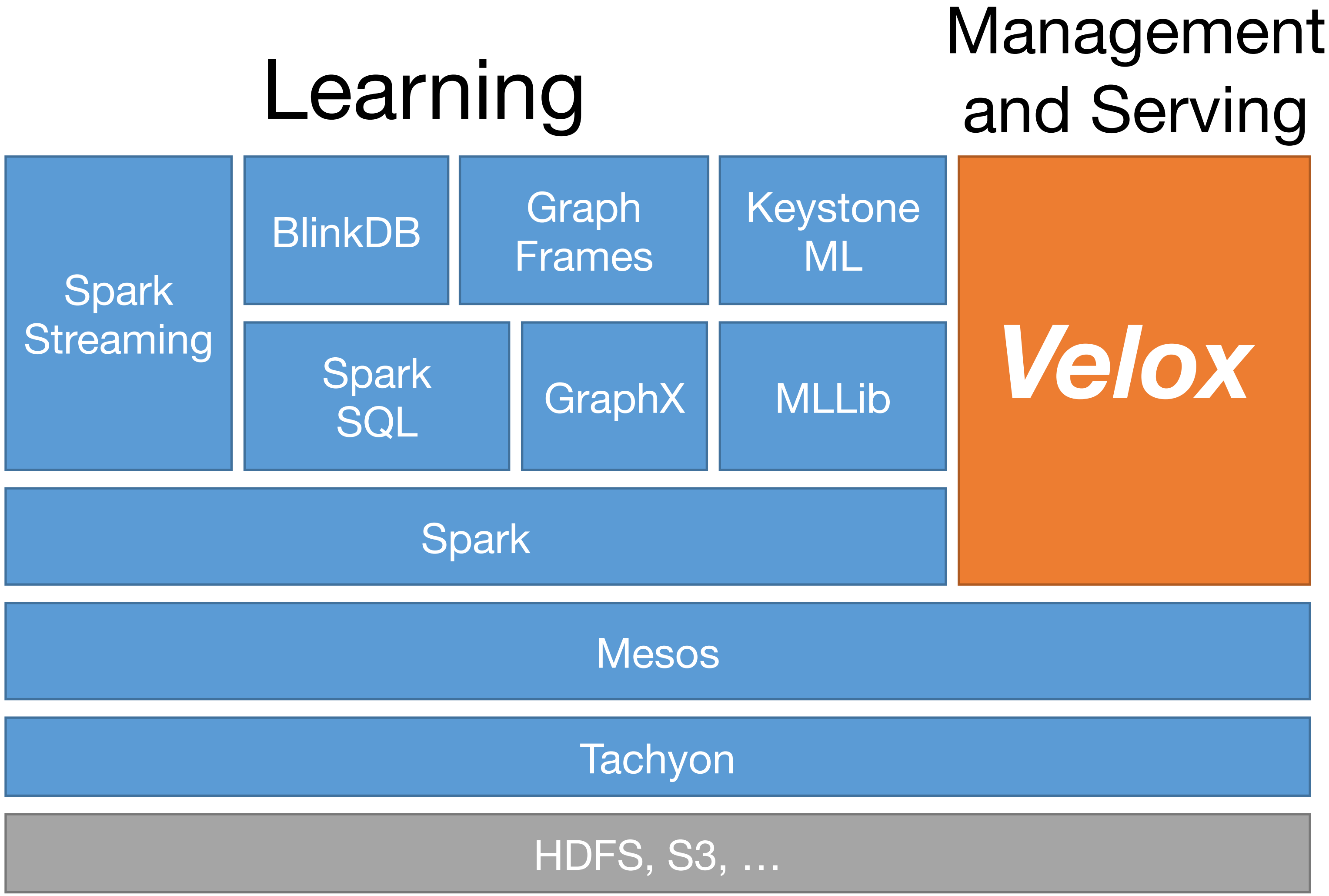
VELOX: the Missing Piece of BDAS



amplab

Berkeley
Data
Analytics
Stack

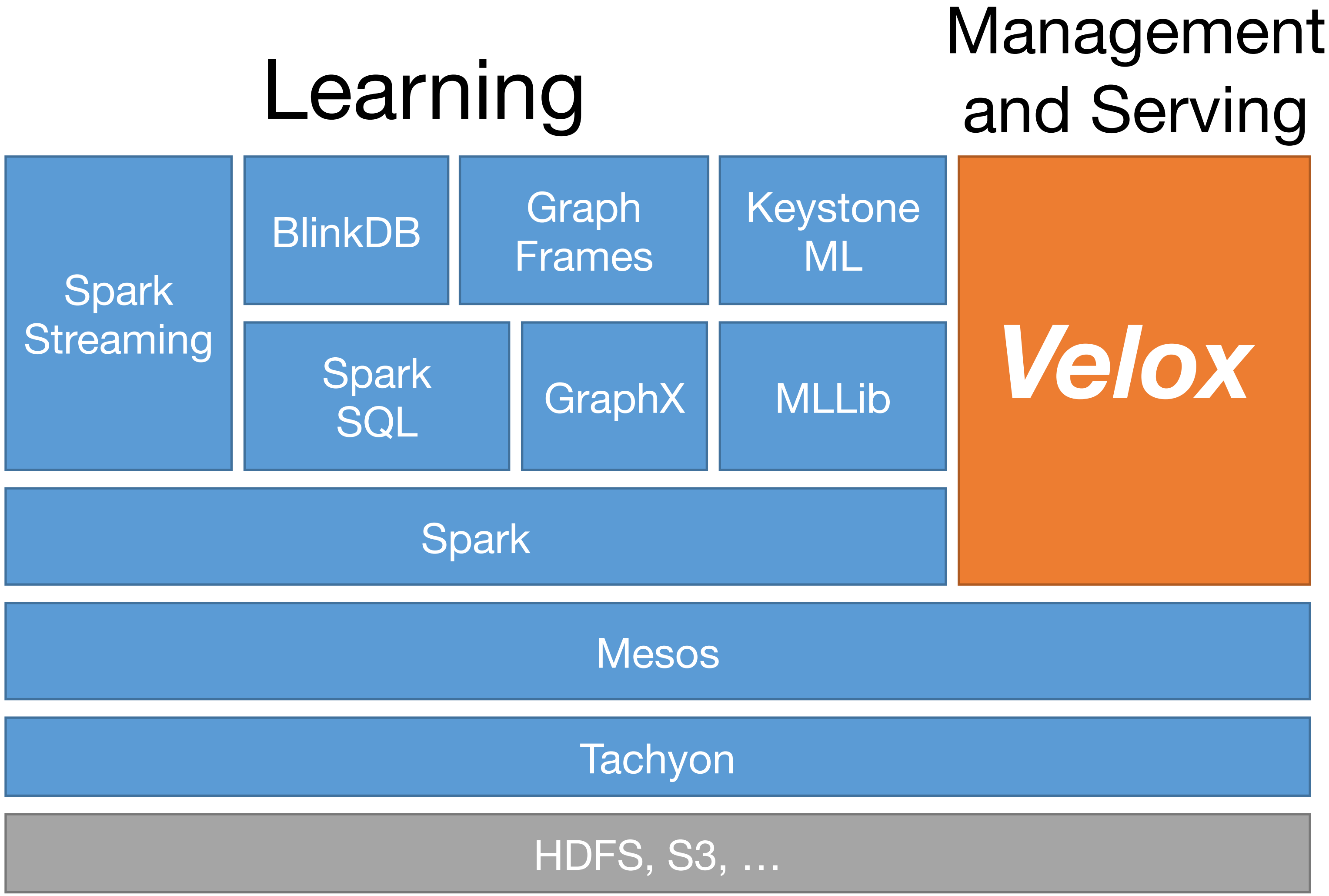
VELOX: the Missing Piece of BDAS



amplab

Berkeley
Data
Analytics
Stack

VELOX: the Missing Piece of BDAS



amplab

Berkeley
Data
Analytics
Stack

VELOX Architecture

Fraud Detection

Content Rec.

NETFLIX

Keystone ML

MLLib

Spark

Velox

Single JVM Instance

VELOX Architecture

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

NETFLIX

Keystone ML

MLLib

Spark

Velox

Single JVM Instance

scikit learn

Dato Create

Caffe

dmlc mxnet

TensorFlow

VW

KALDI

theano

VELOX as a Middle Layer Arch?

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

NETFLIX

Generalize *Velox*?

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

Clipper

A Low-Latency Online Prediction Serving System

Daniel Crankshaw

Xin Wang

Michael Franklin

Joseph E. Gonzalez

Ion Stoica

Clipper **Generalizes** Velox Across ML Frameworks

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

NETFLIX

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 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.

NETFLIX

Personal Asst.

Robotic Control

Machine Translation

Clipper

theano

Dato Create

KeystoneML

Caffe

TensorFlow

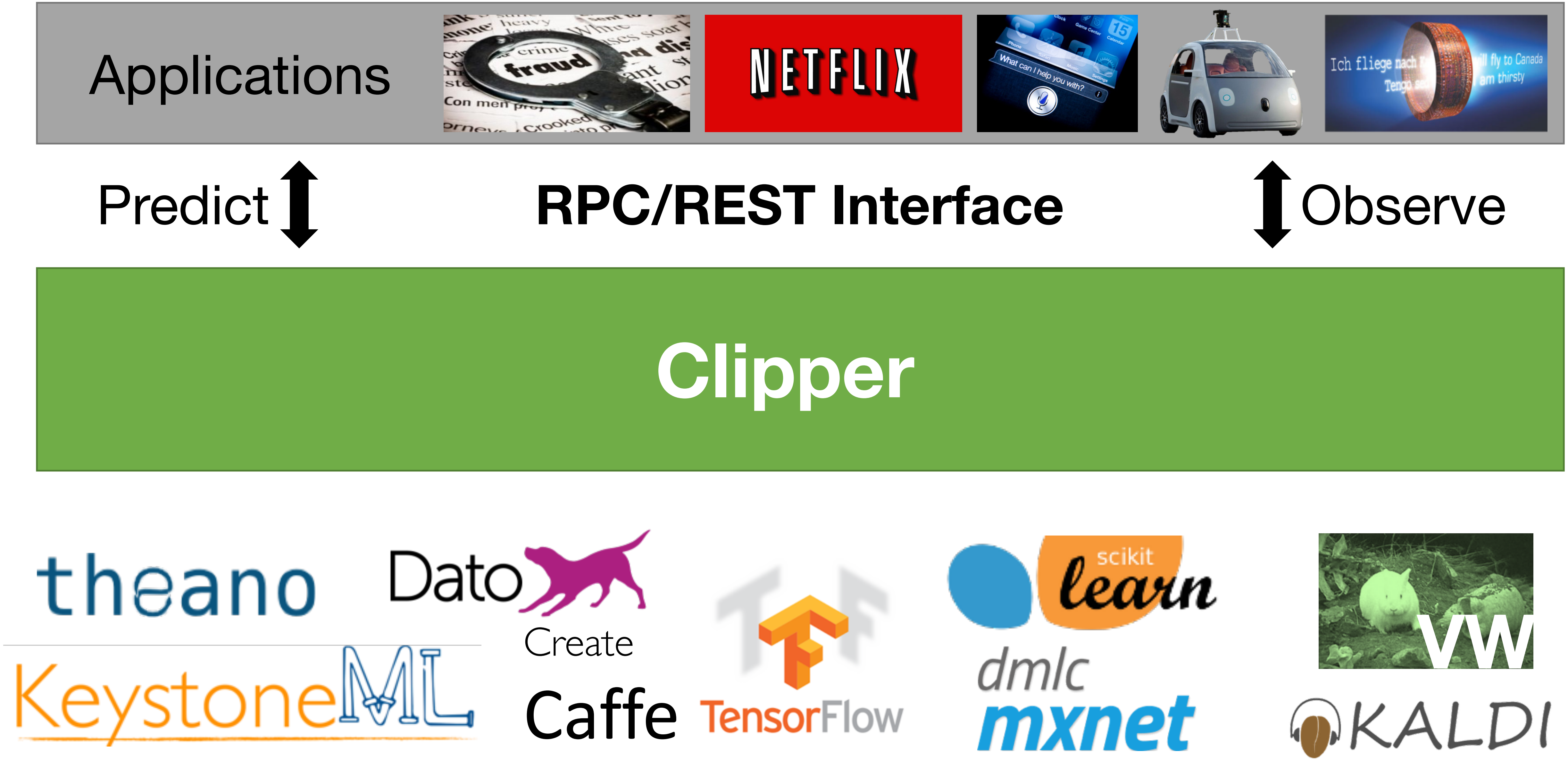
scikit learn

dmlc mxnet

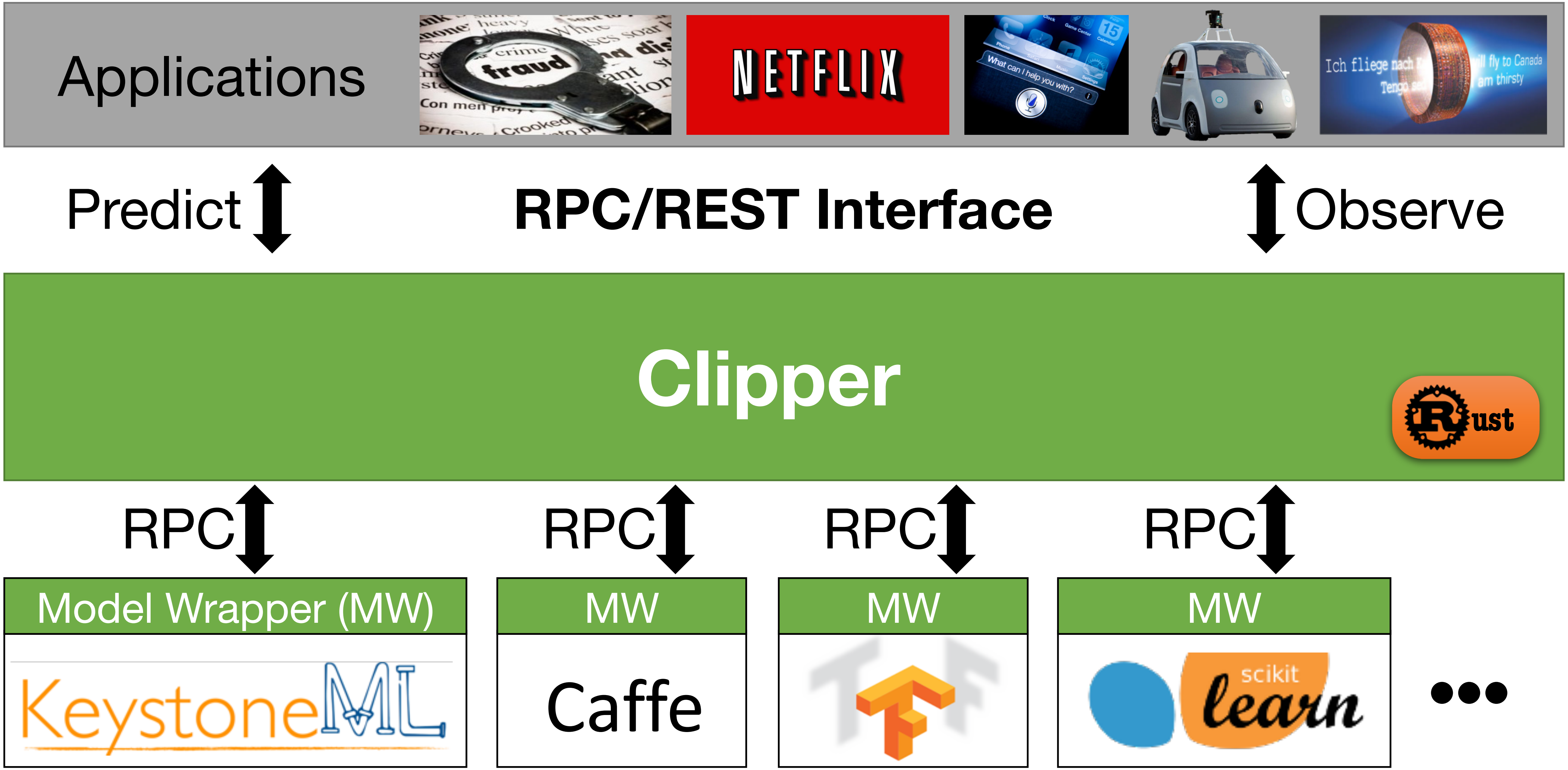
VW

KALDI

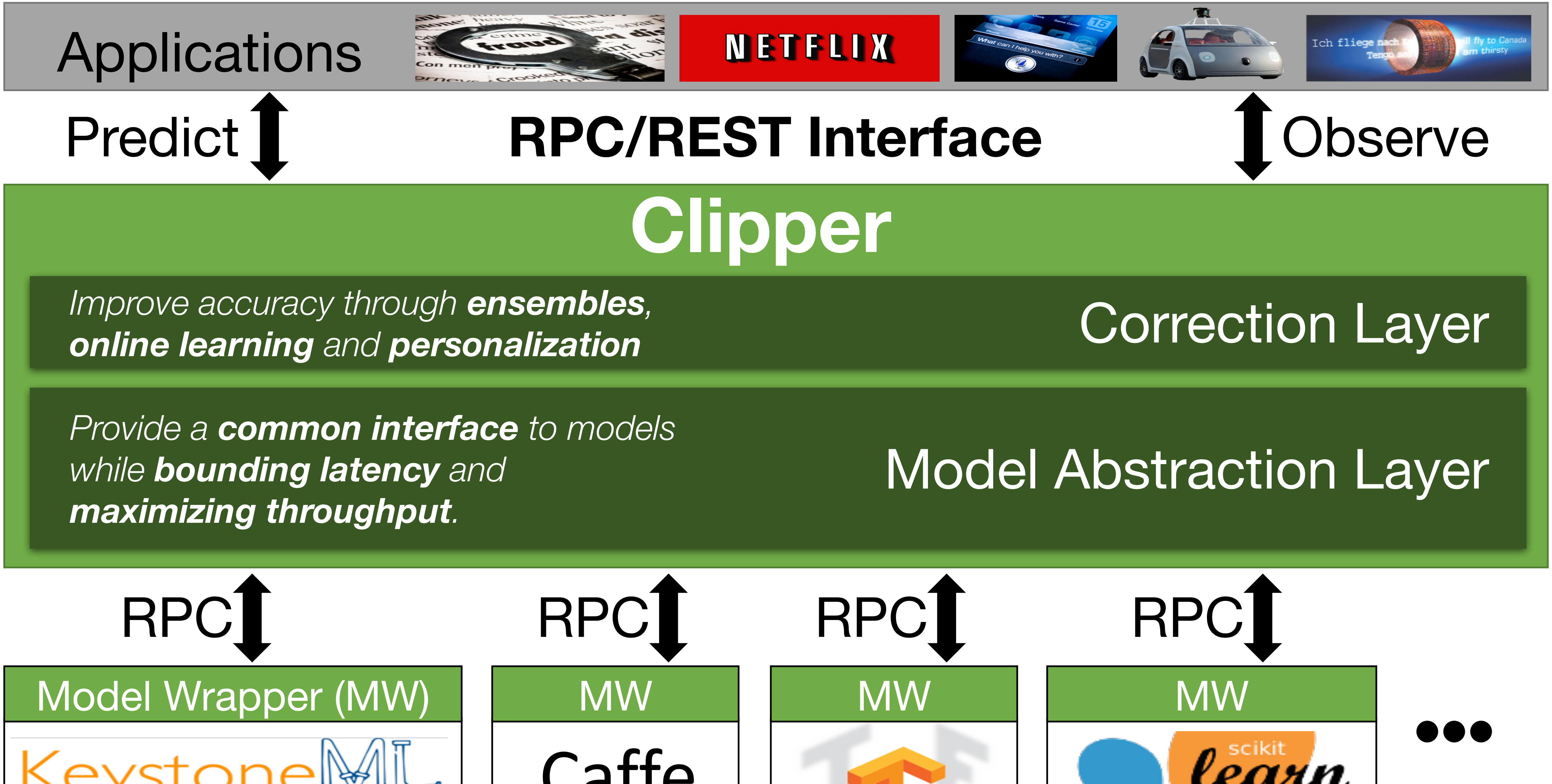
Clipper Architecture



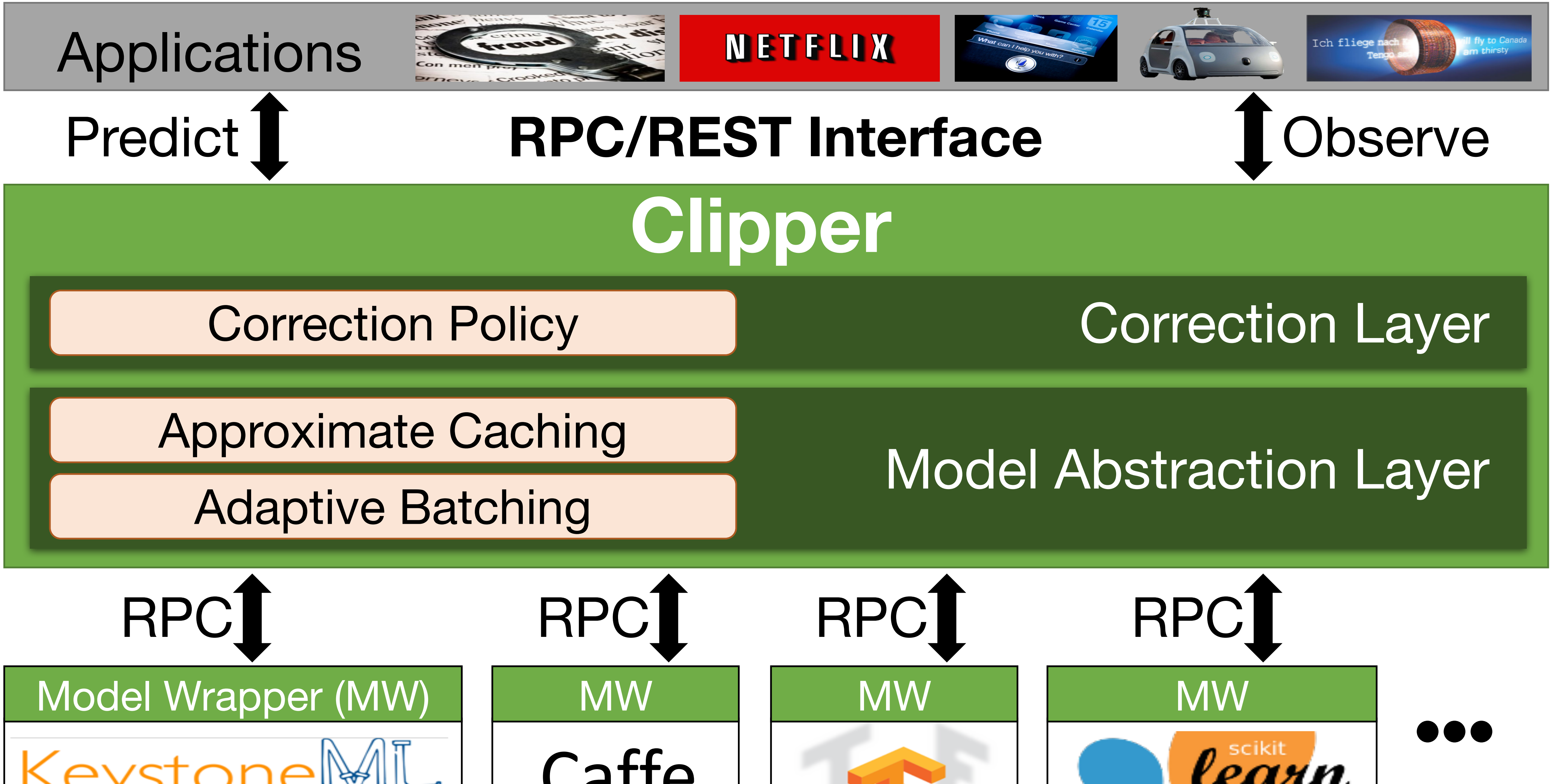
Clipper Architecture

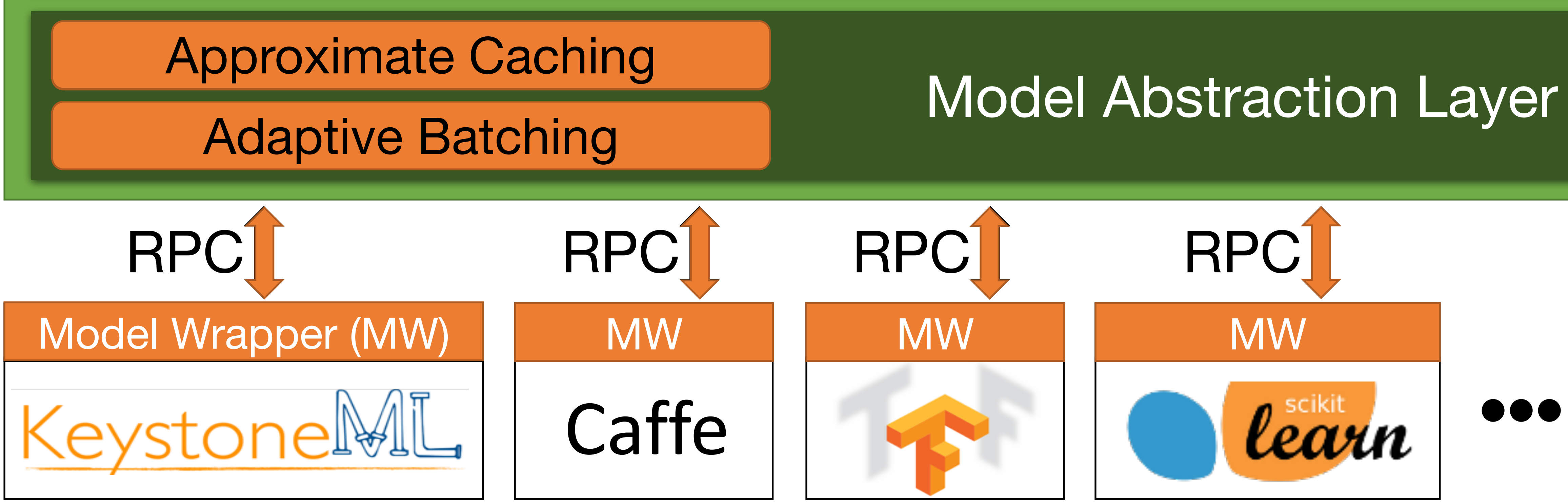


Clipper Architecture



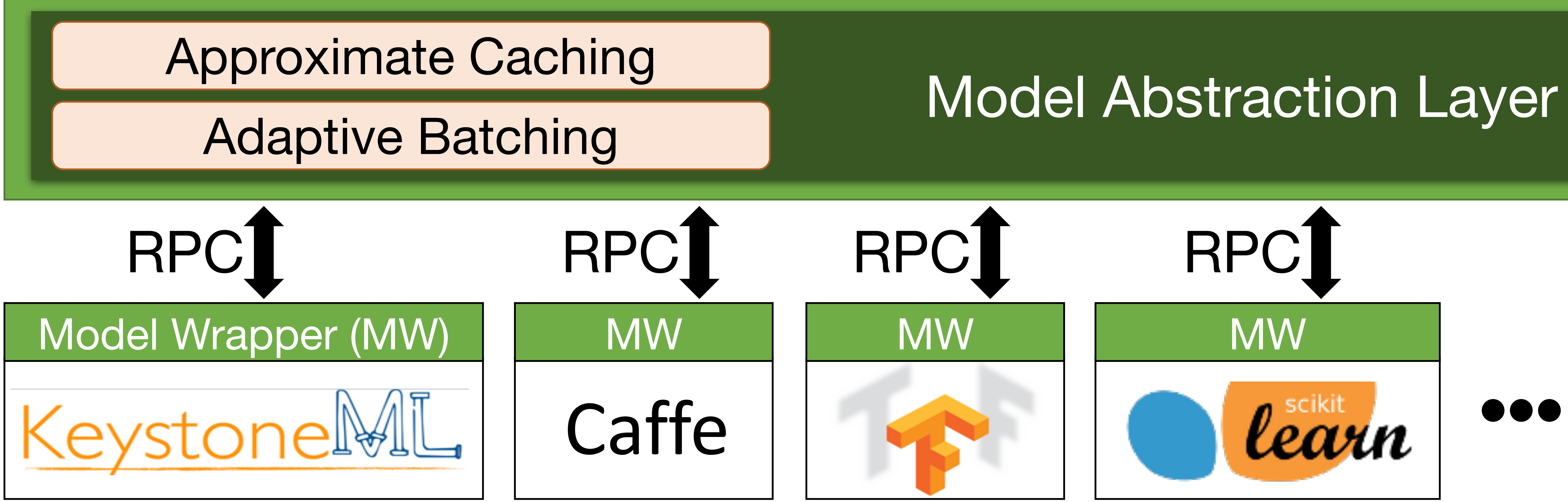
Clipper Architecture

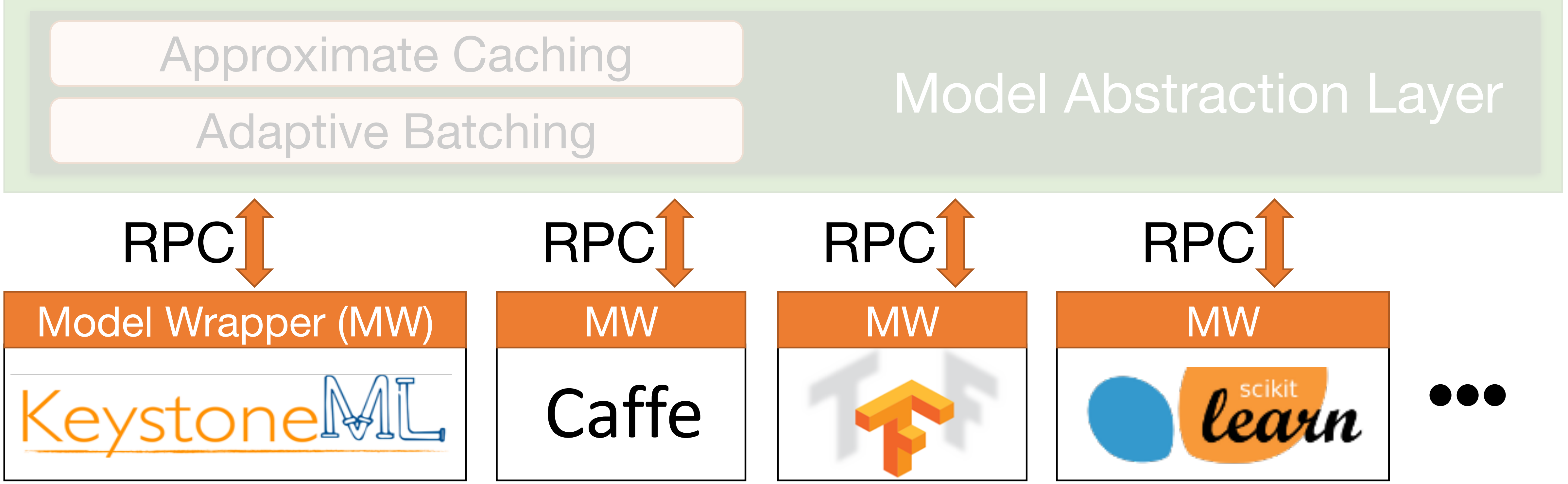




Provides a unified generic prediction API across **frameworks**

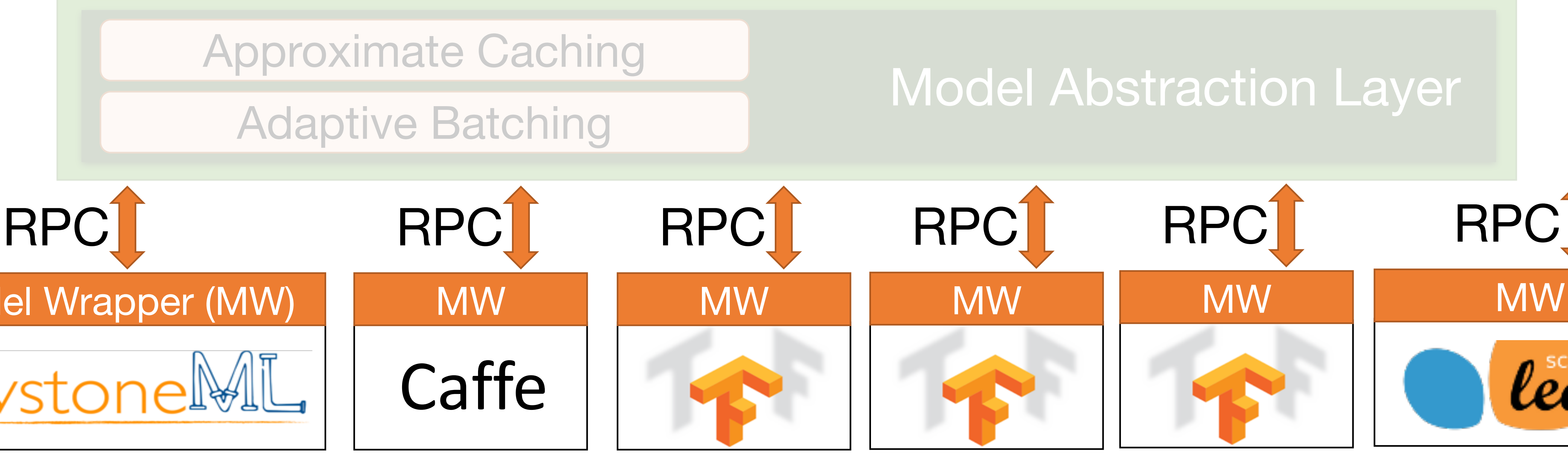
- **Reduce Latency** → Approximate Caching
- **Increase Throughput** → Adaptive Batching
- **Simplify Deployment** → RPC + Model Wrapper


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



Common Interface → Simplifies Deployment:

- ➢ Evaluate models using original code & systems
- ➢ Models run in separate processes
  - ➢ Resource isolation



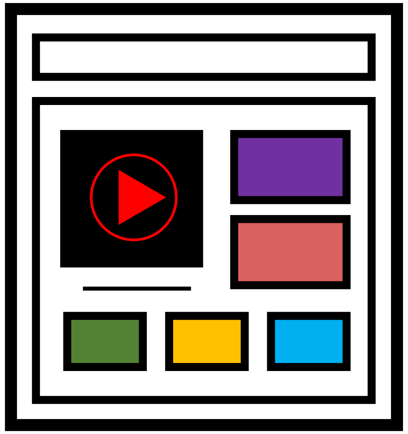
Common Interface → Simplifies Deployment:

- 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

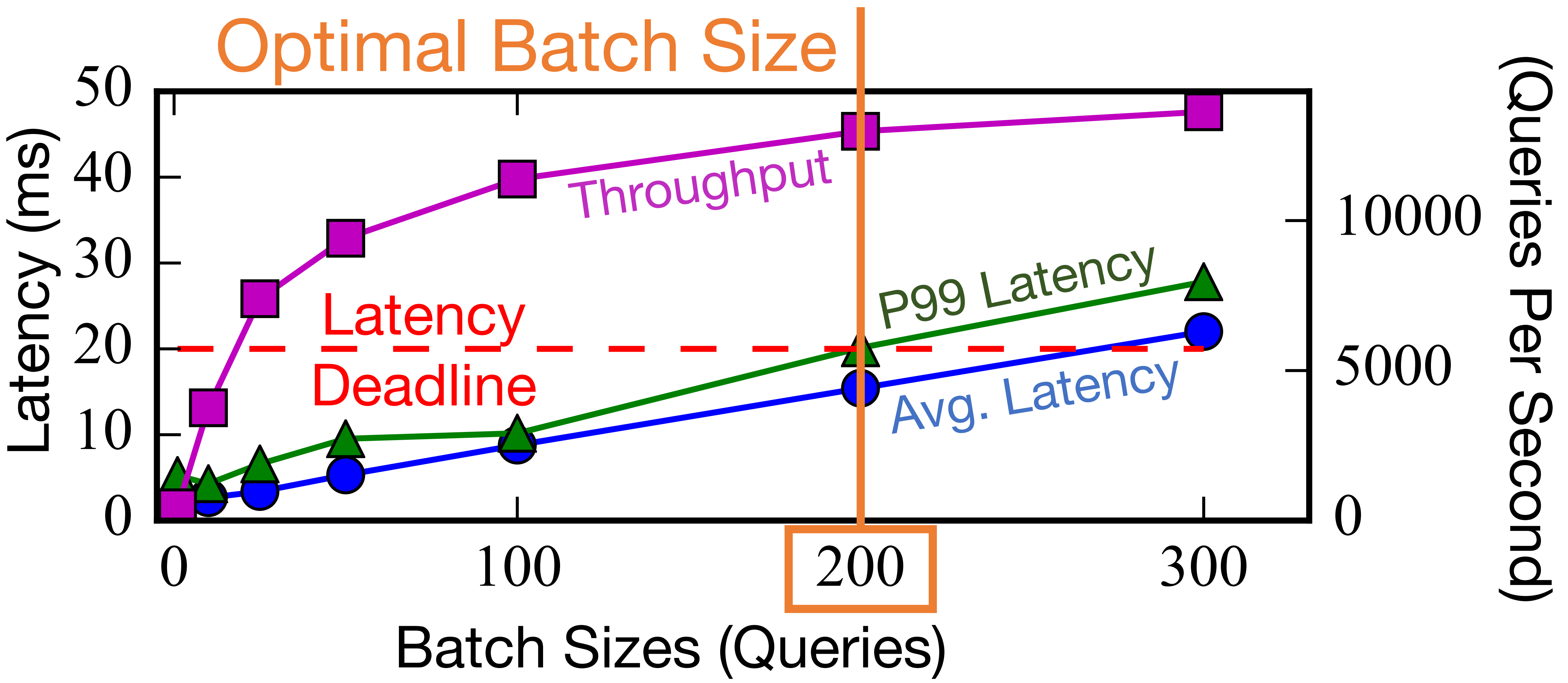
- Optimal batch depends on:
  - hardware configuration
  - model and framework
  - system load

Clipper Solution:

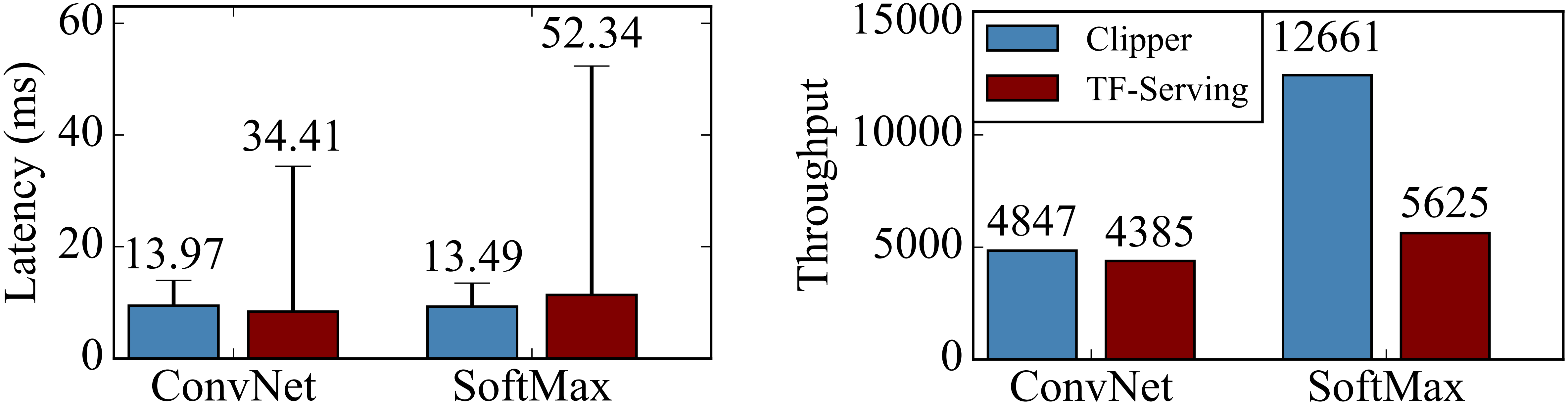
be as ***slow*** *as* ***allowed****…*

- 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)



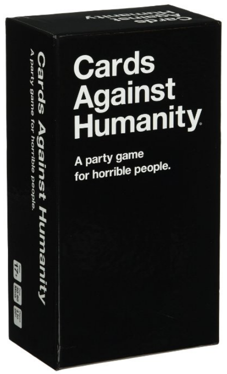
Comparison to TensorFlow Serving



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

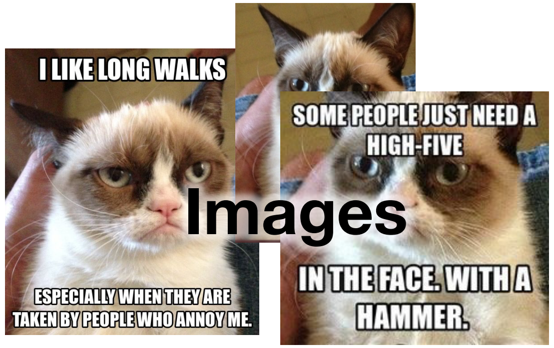
- ➢ Opportunity for caching



Popular items may be evaluated frequently

- ➢ Need for **approximation**

Bag-of-Words Model

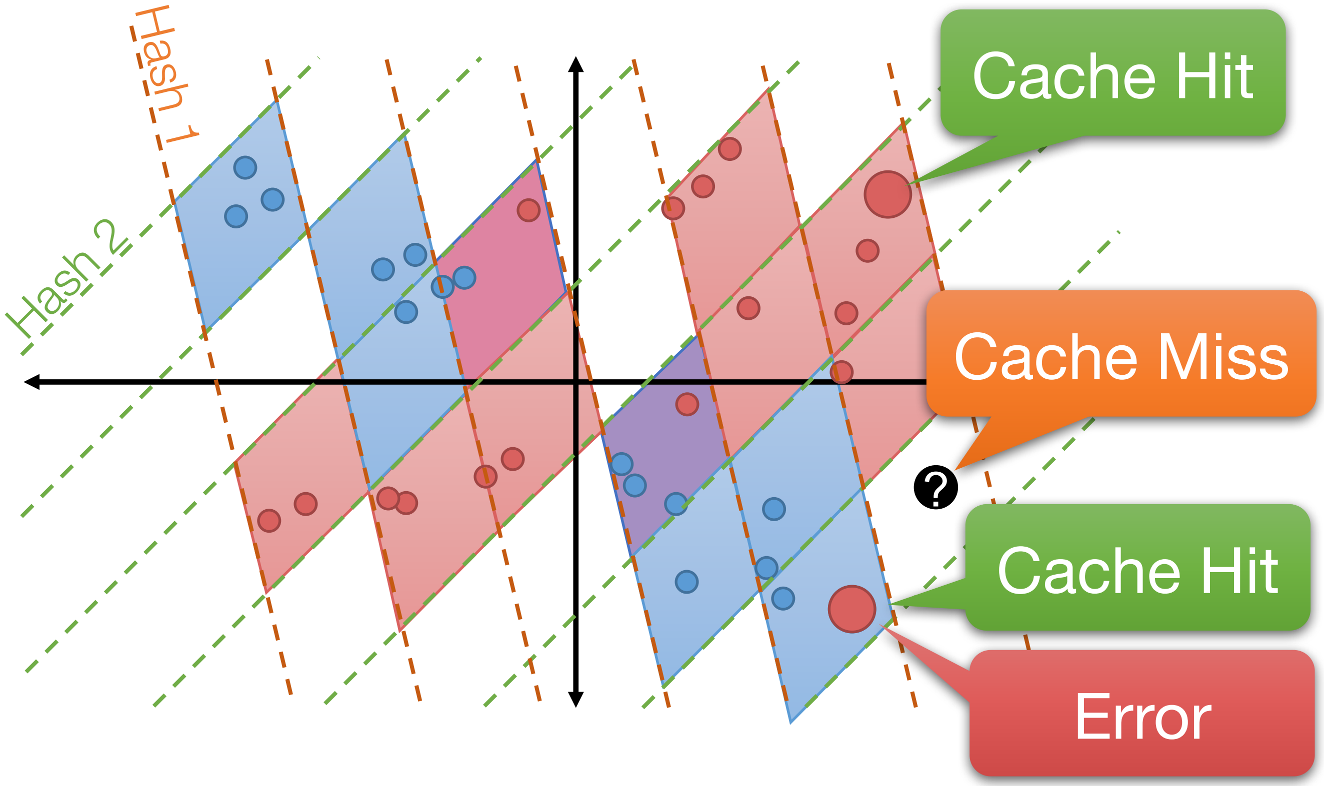


Images

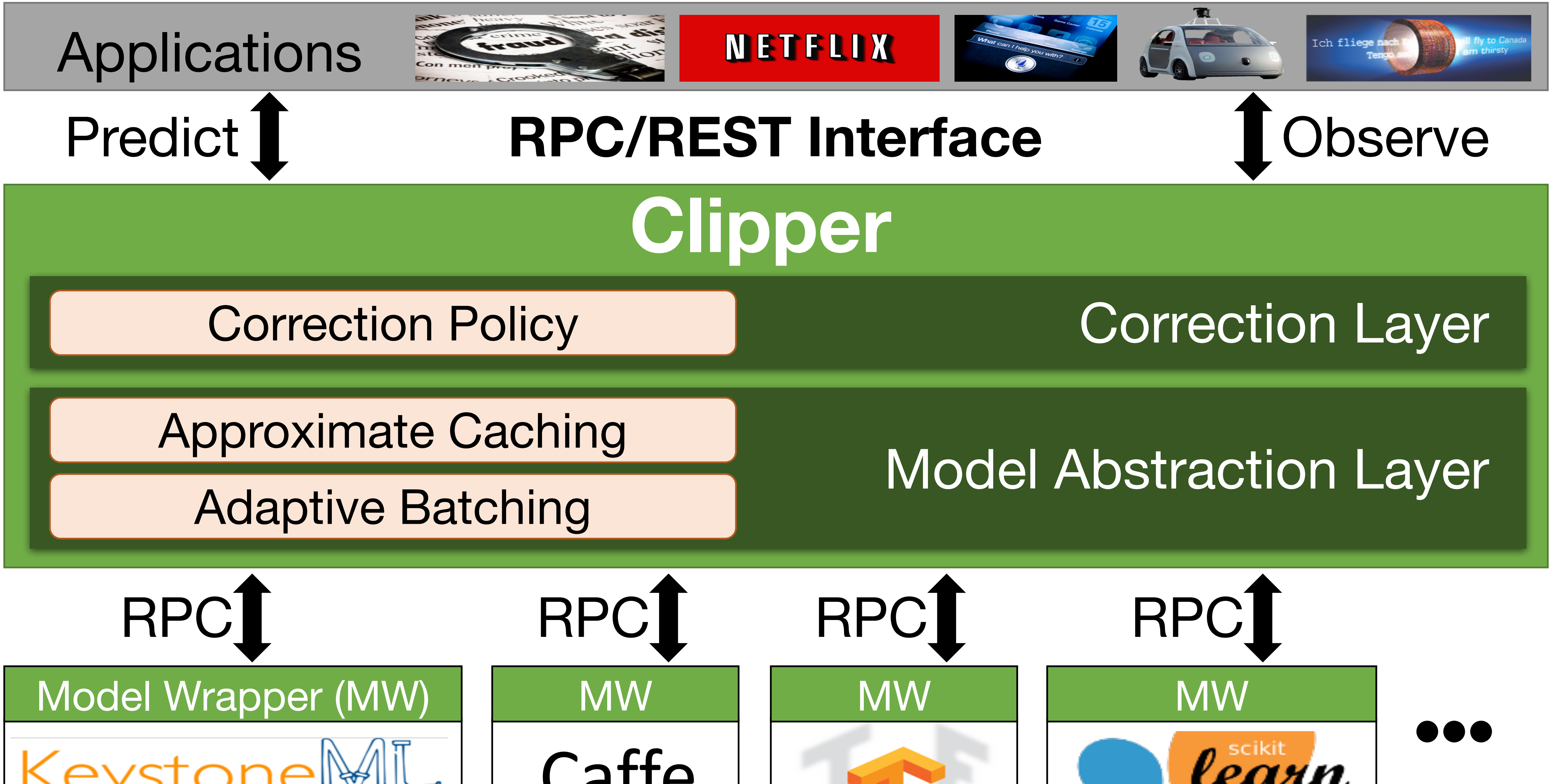
High Dimensional and continuous valued queries have low cache hit rate.

Clipper Solution: *Approximate Caching*

apply *locality sensitive hash functions*



Clipper Architecture



Correction Policy

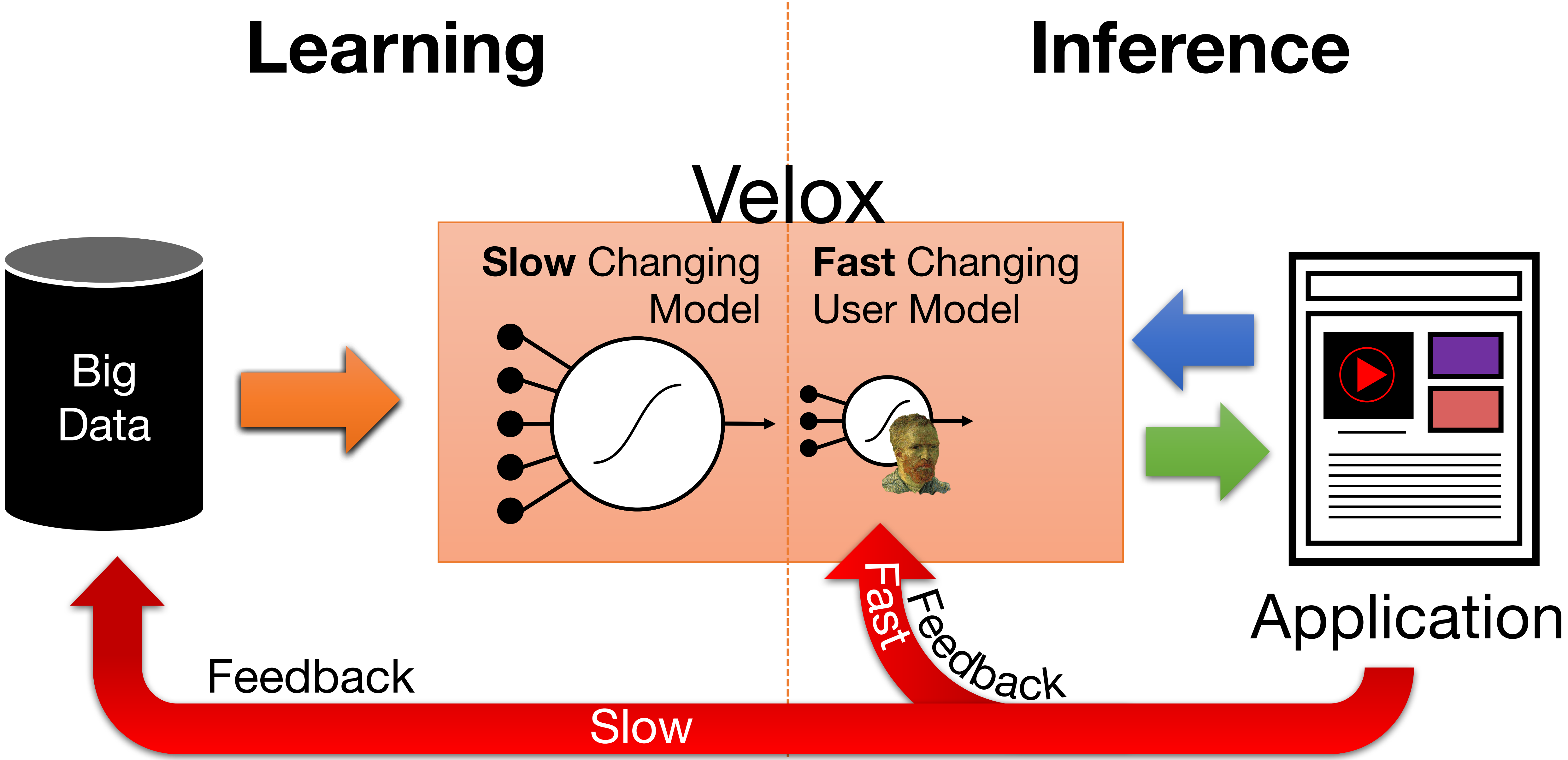
Correction Layer

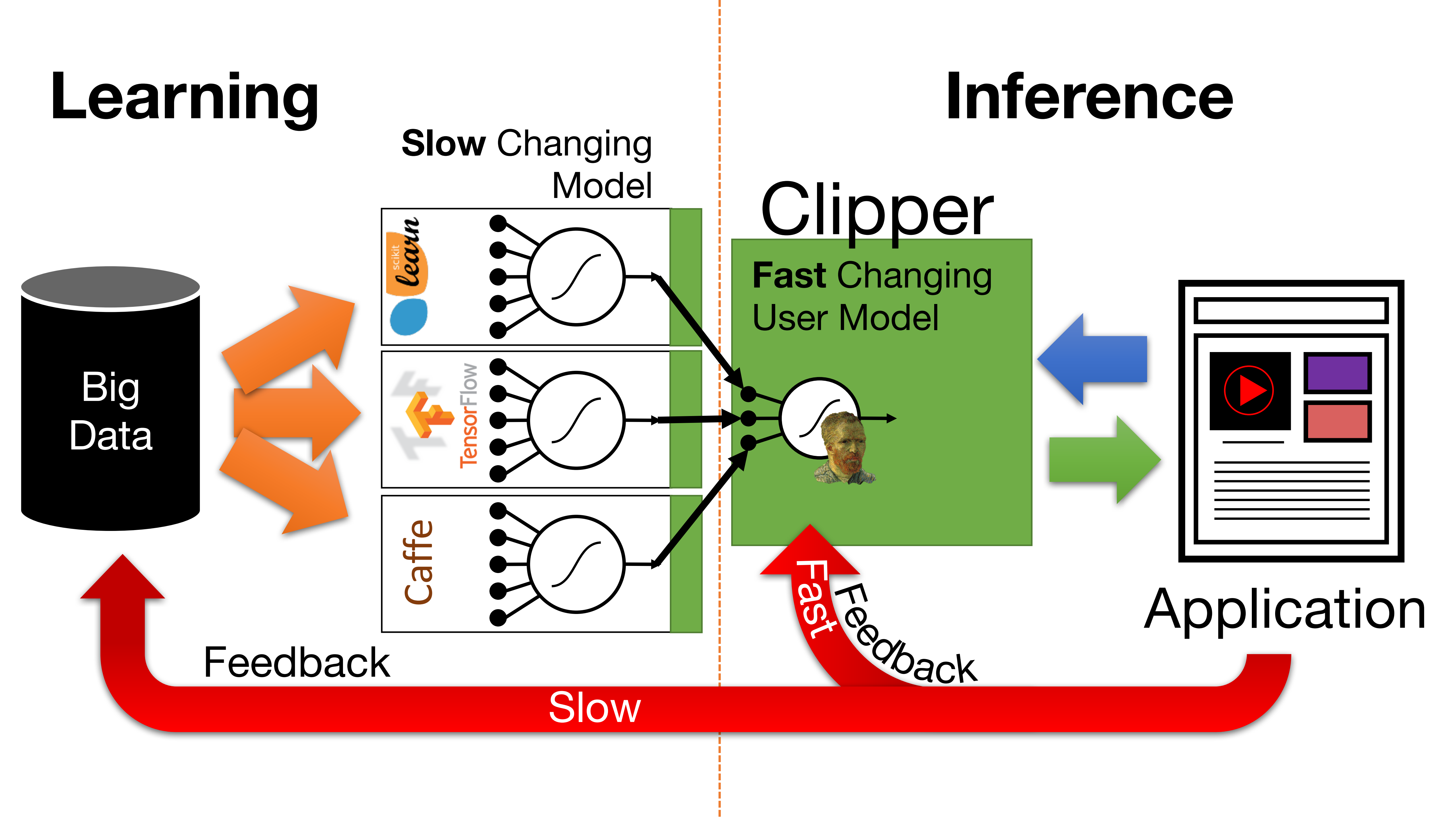
Goal:

*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

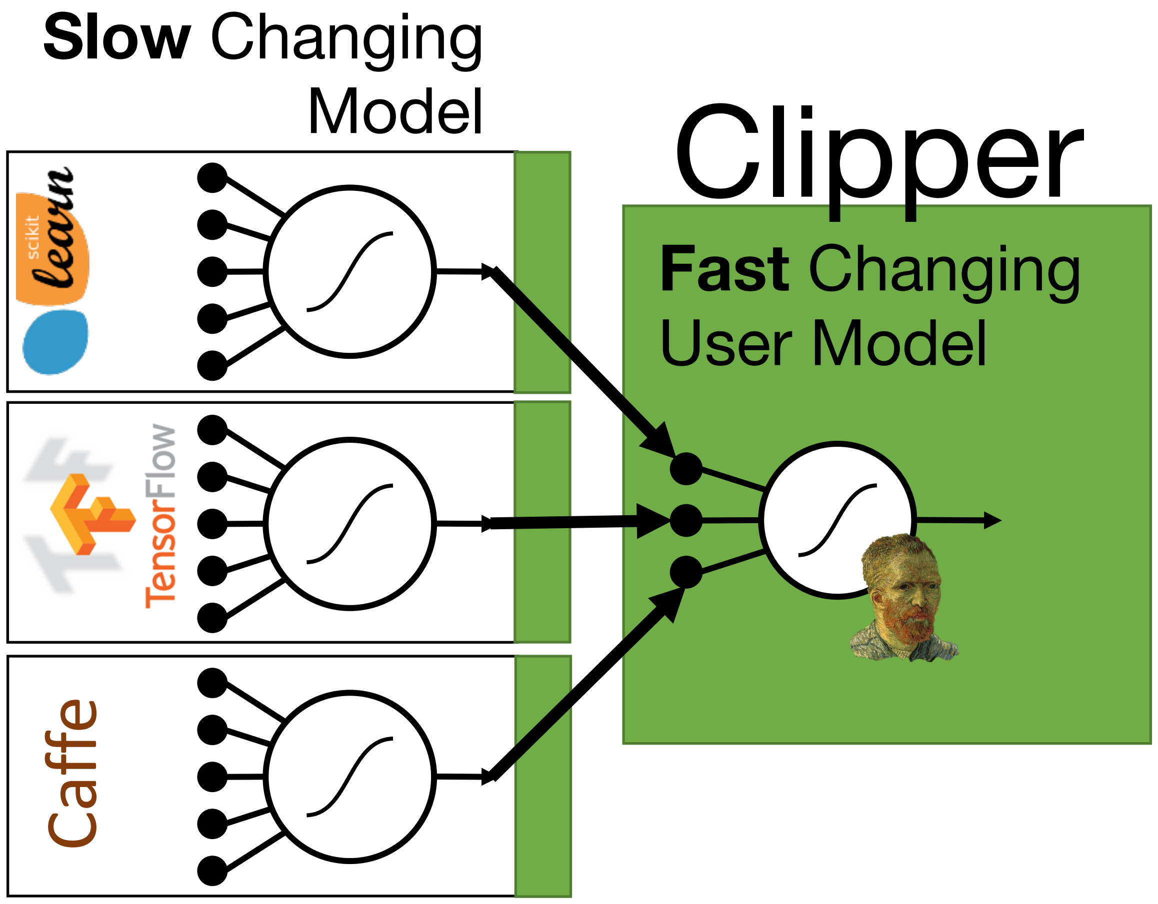

Learning
Inference
Velox
Big Data
Slow Changing Model
Fast Changing User Model
Application
Feedback
Slow
Fast Feedback


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

Correction Policy

Improves prediction **accuaray** by:

- Incorporating real-time **feedback**
- Managing **personalization**
- **Combine** models & **frameworks**
  - enables frameworks to **compete**



Improved 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 state-of-the-art models

Improved Prediction Accuracy

System			#Errors
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!

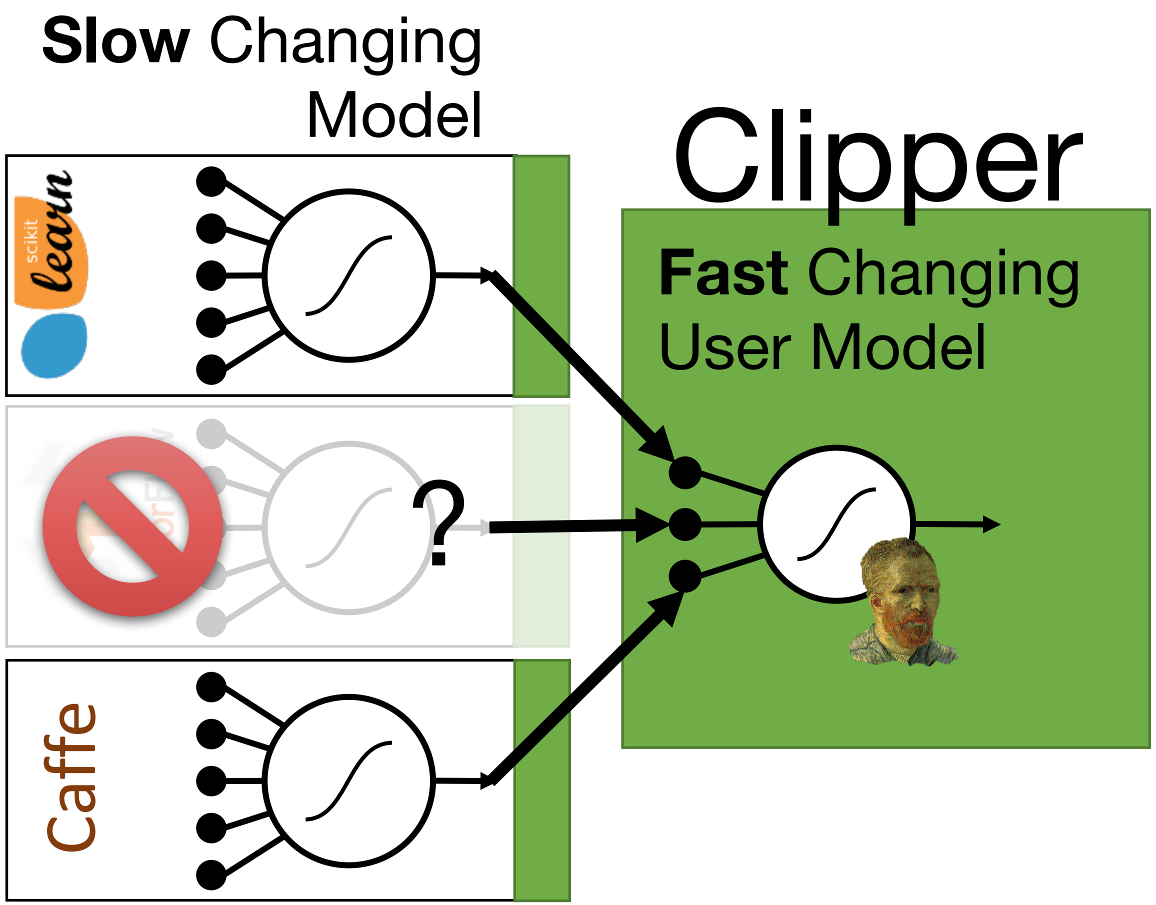
Cost of Ensembles

Increased Load

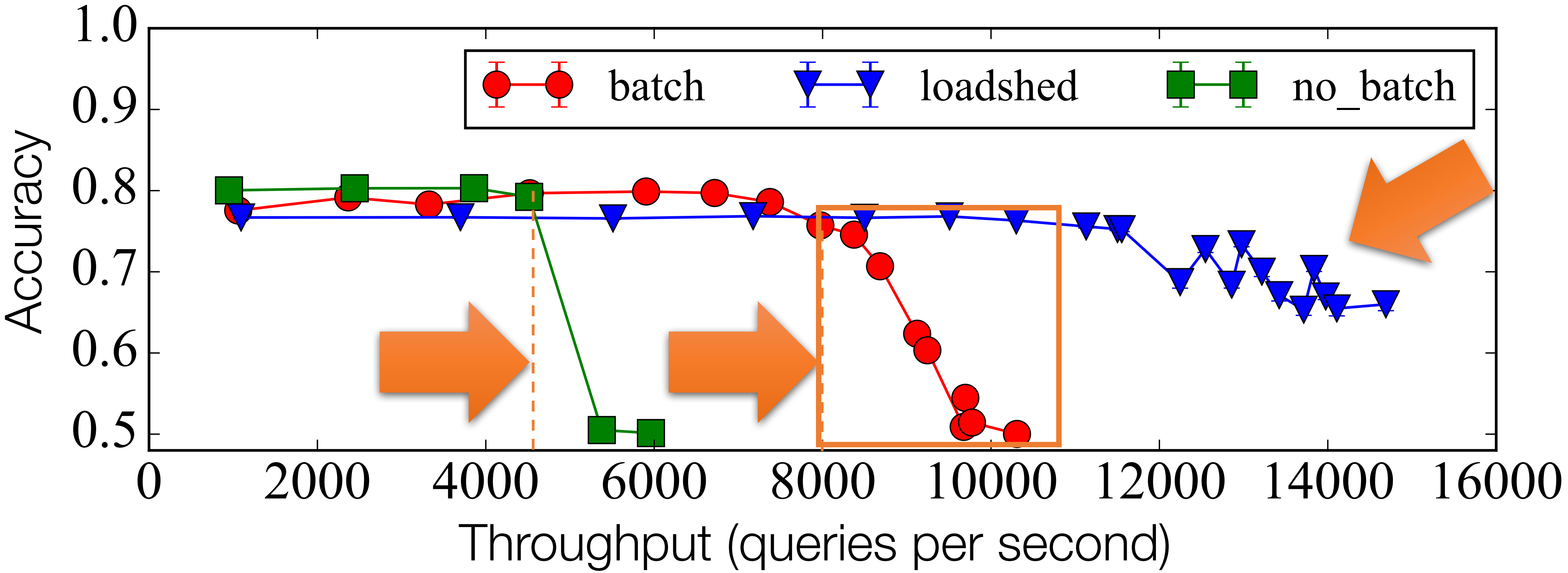
- *Solutions:*
  - **Caching** and **Batching**
  - **Load-shedding** correction policy can prioritize frameworks

Stragglers

  - e.g., framework fails to meet SLO
- *Solution:* **Anytime** predictions
  - Correction policy must render predictions with missing inputs
  - e.g., built-in correction policies **substitute expected value**



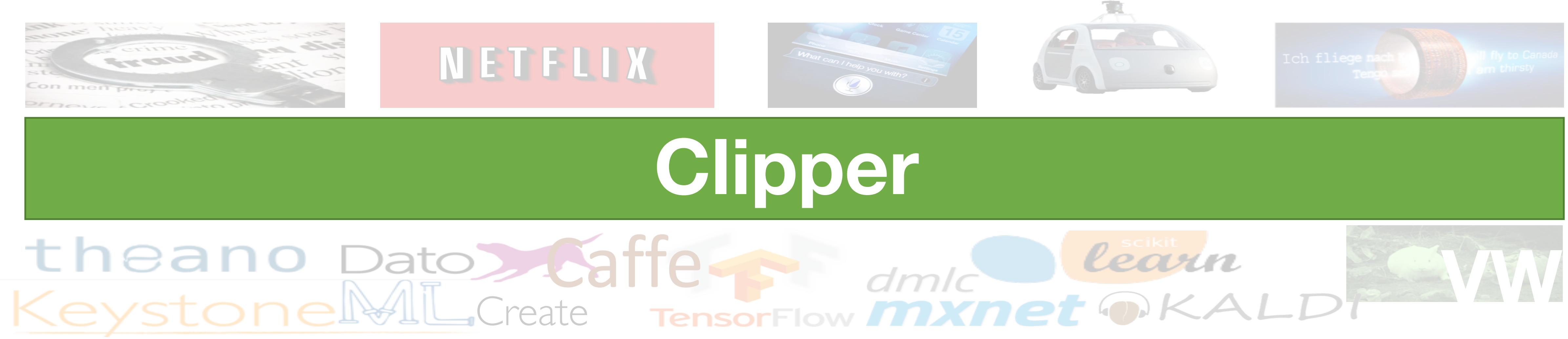
Evaluation of Throughput Under Heavy Load



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

Conclusion

Clipper sits between applications and ML frameworks to

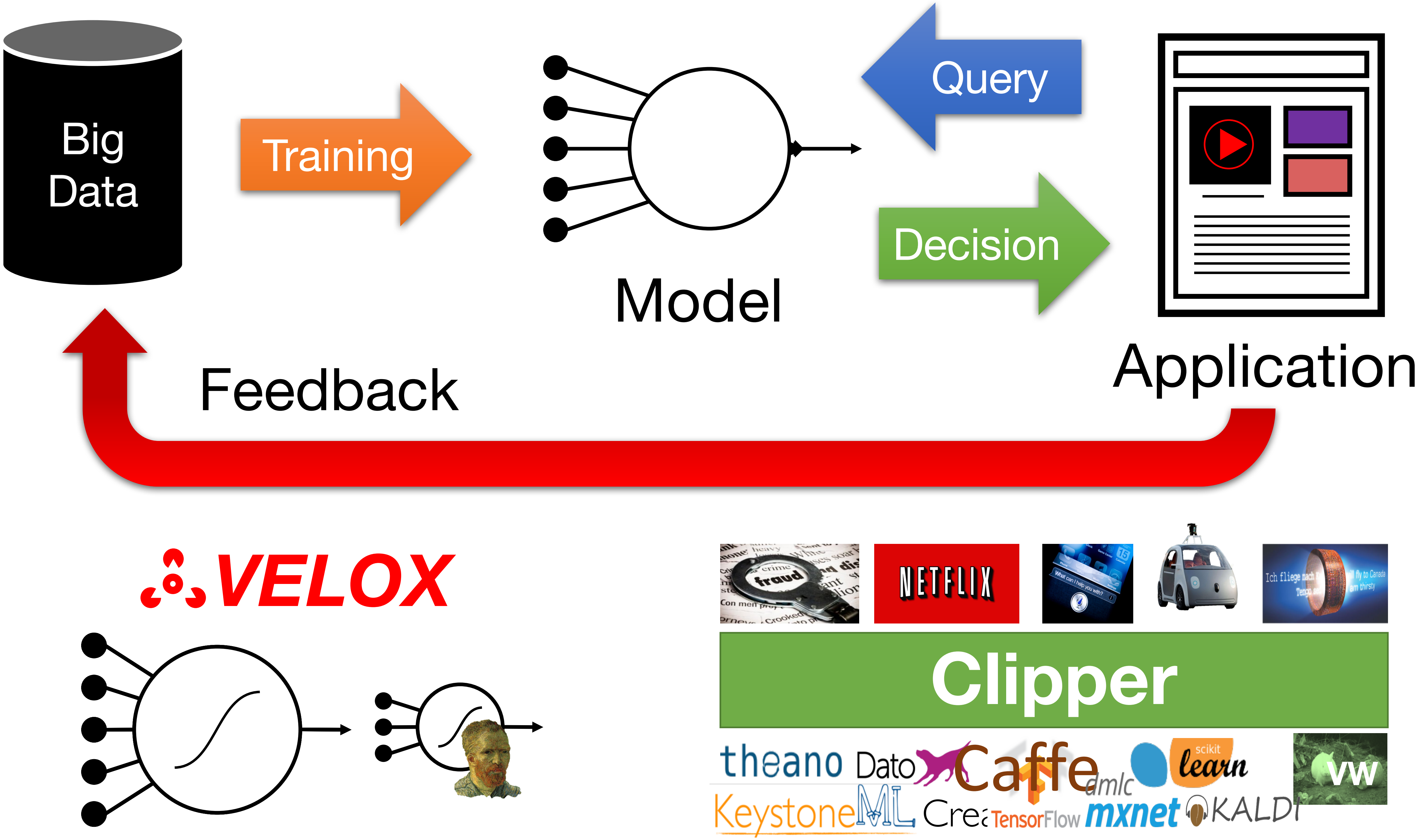


➢ to **simplifying deployment**

➢ **bound latency** and **increase throughput**

➢ and enable **real-time learning** and **personalization**

across **machine learning frameworks**

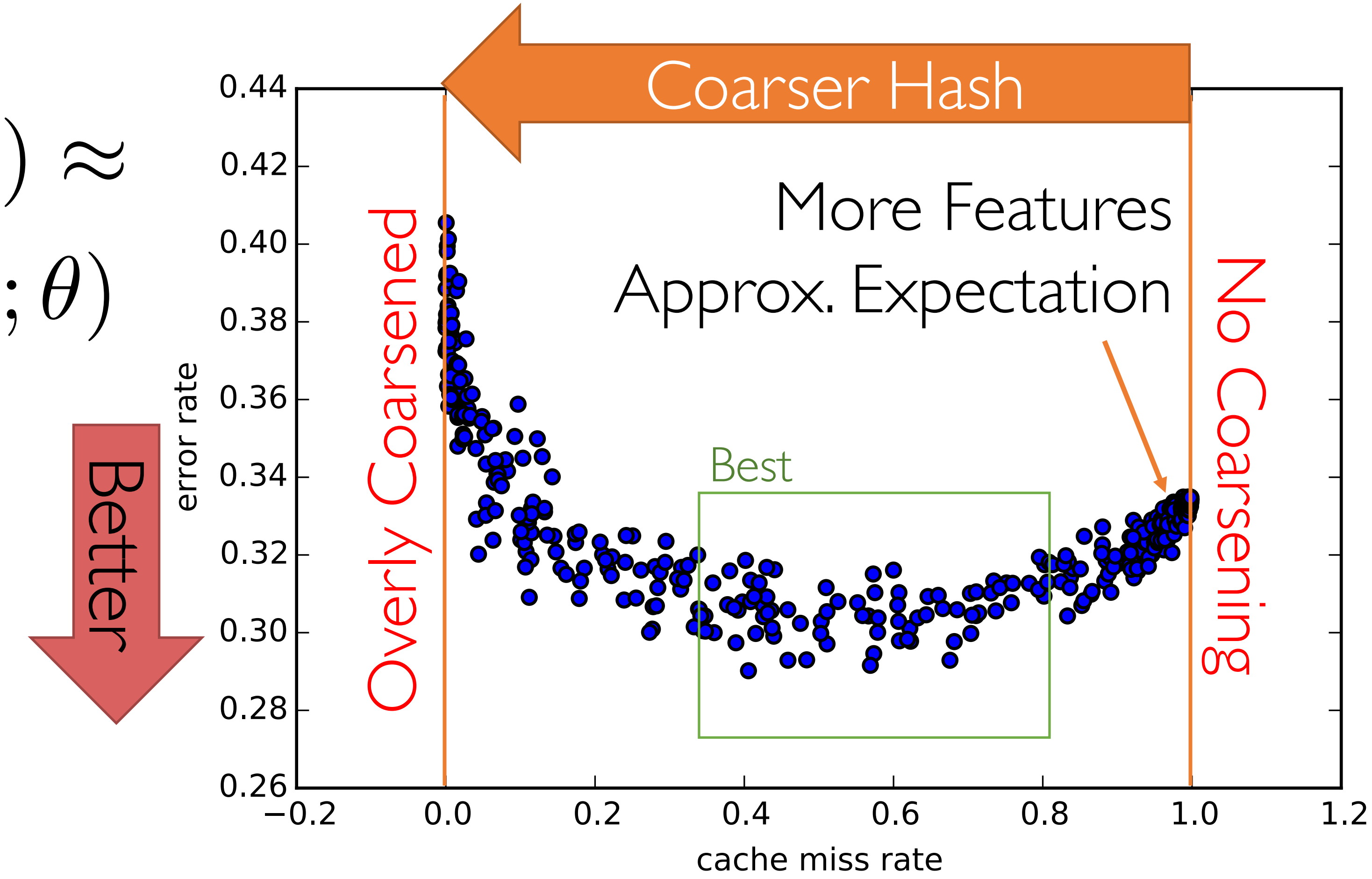

Big Data
Training
Model
Query
Decision
Application
Feedback
VELOX
NETFLIX
Clipper
theano
Dato
Caffe
dmlc
scikit learn
VW
KeystoneML
TensorFlow
mxnet
KALDI

Ongoing & Future Research Directions

- Serving and updating RL models
- Bandit techniques in correction policies
- Splitting inference across the cloud and the client to reduce latency and bandwidth requirements
- Secure model evaluation on the client (model DRM)

Coarsening + Anytime Predictions

$$f_i(x;\theta) \approx f_i(z;\theta)$$



$$f_i(x;\theta) \approx \mathbb{E}\left[f_i(x;\theta)\right]$$