CLIPPER: A Low Latency Online Prediction Serving System

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

Hello!

I am Raja Haseeb

I am a researcher at RIT lab in Electrical department.

You can find me at:
raja@rit.kaist.ac.kr

Contents

For more details, check the original paper at https://arxiv.org/abs/1612.03079

1.

MOTIVATION

Learning

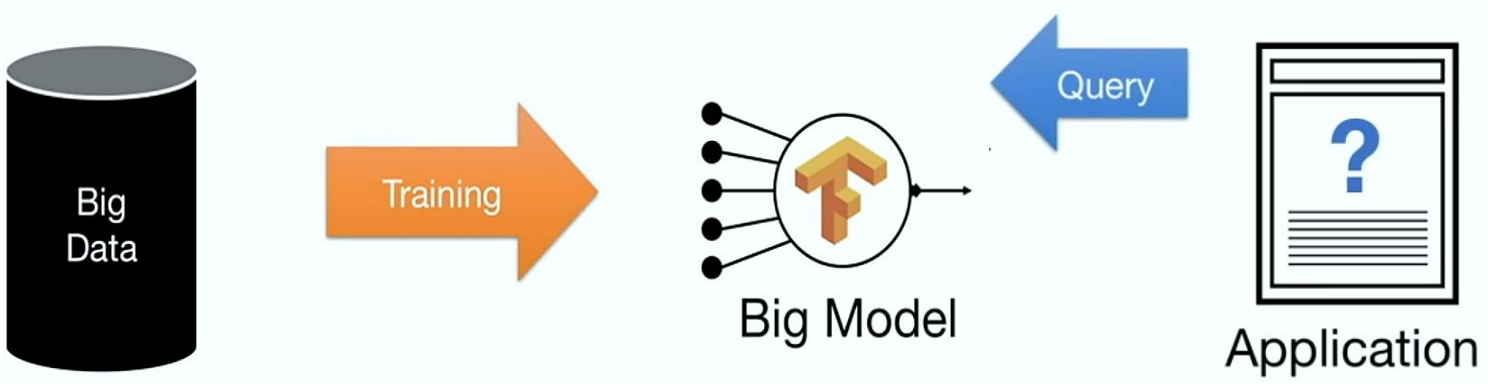
Learning

Timescale: minutes to days

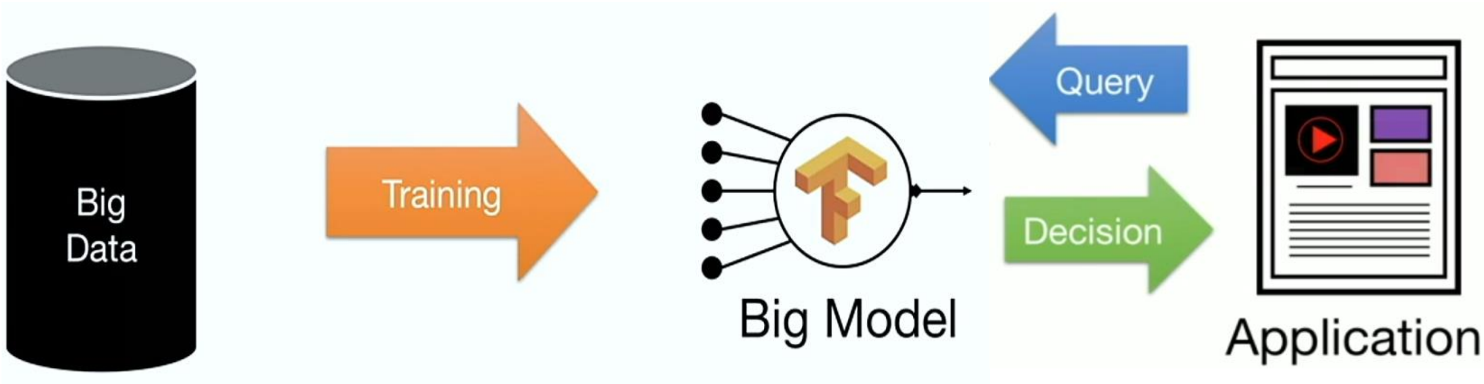
Systems: offline and batch optimized

(*Heavily studied* … major focus of the **AMPLab**)

Learning Inference



Learning Inference

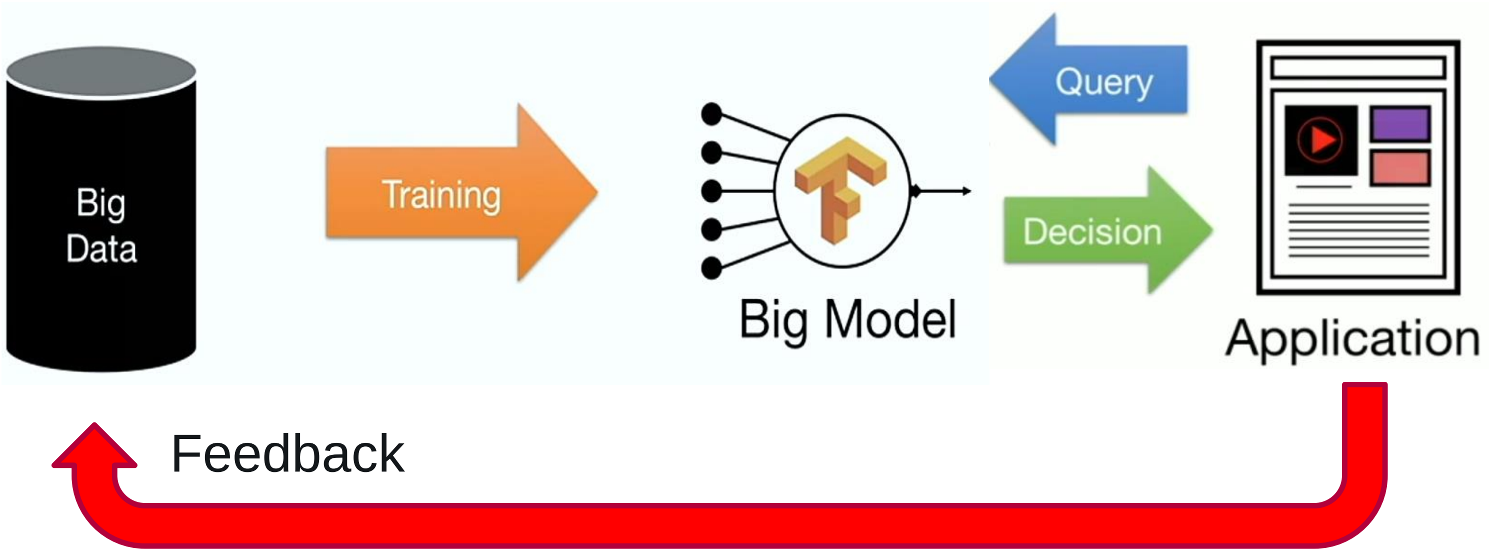


Timescale: ~20 milliseconds

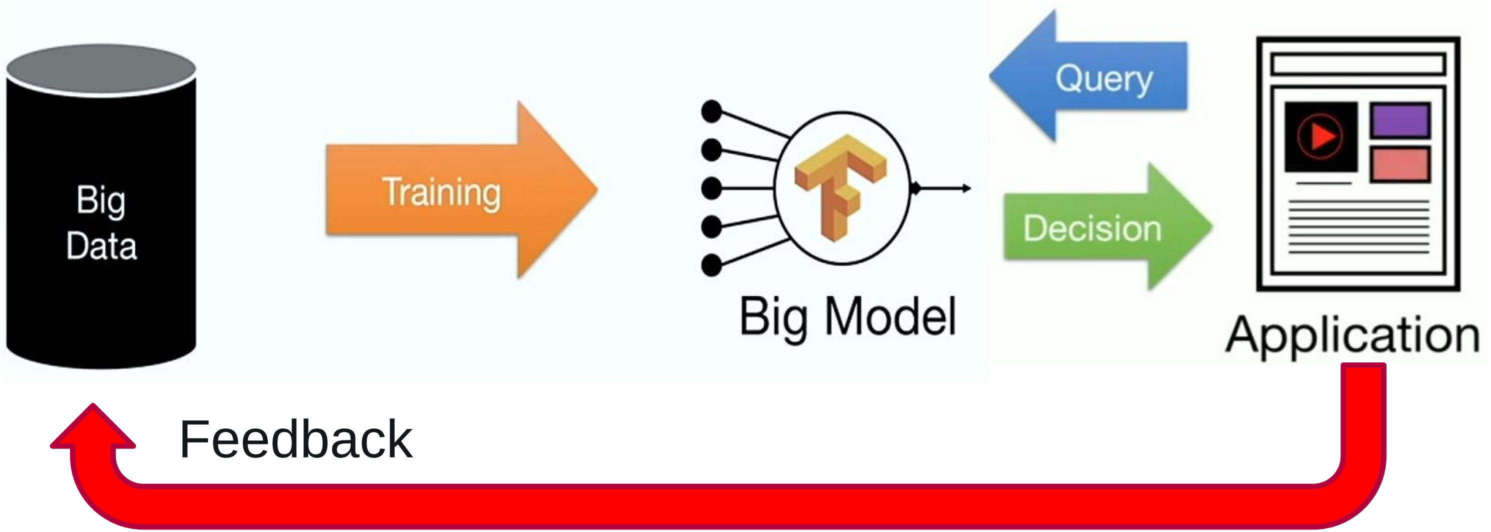
Systems: online and latency optimized

(Less studies ...)

Learning Inference



Learning Inference



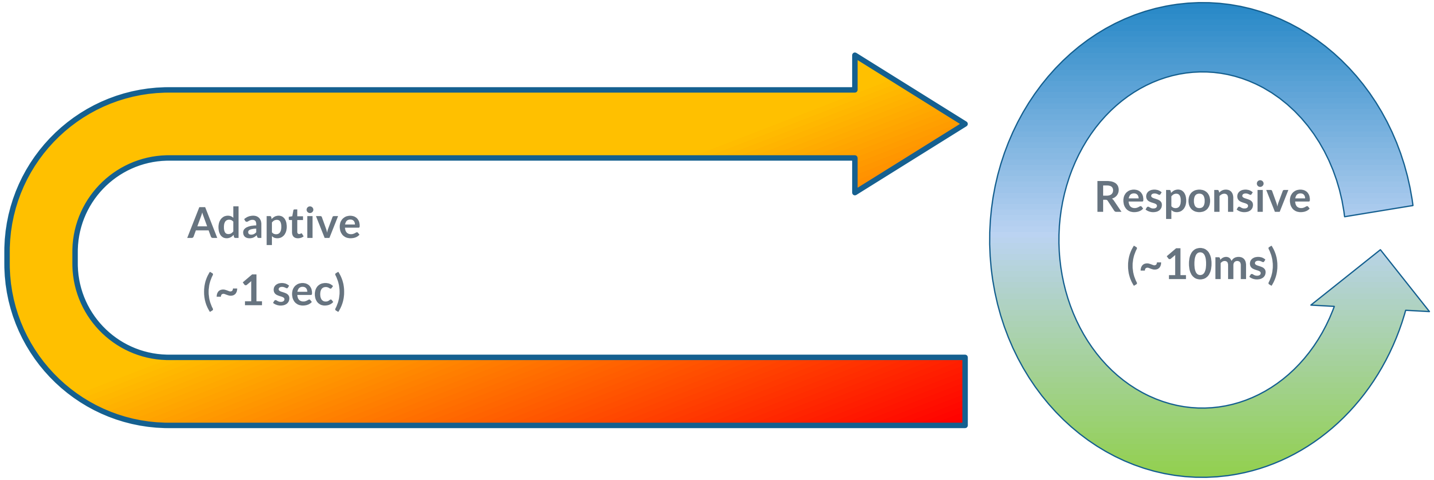
Timescale: hours to weeks

Systems: combination of systems

(Less studies …)

Learning

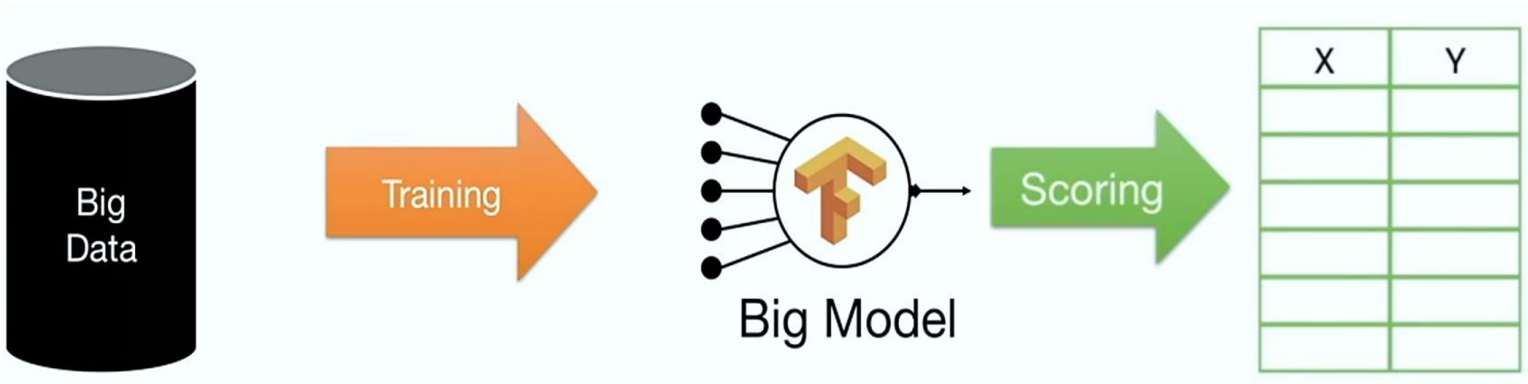
Inference



Serving Predictions Today: Offline Scoring

Offline Batch System

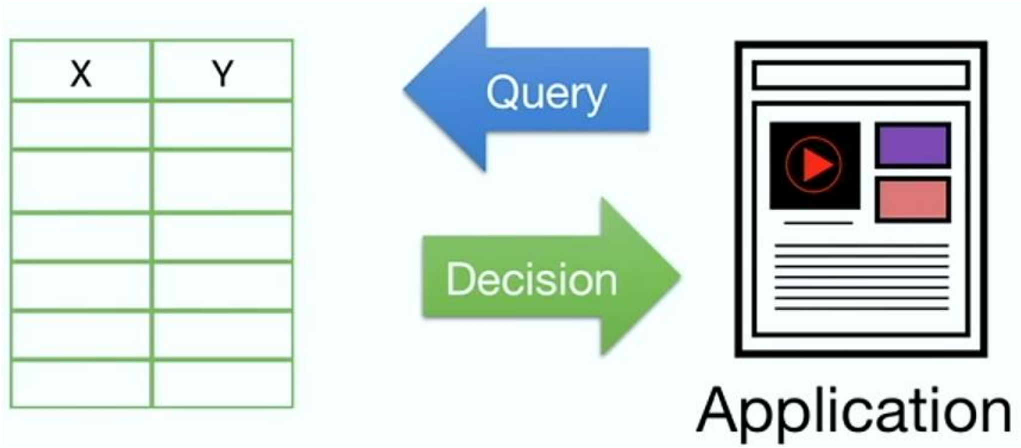
Serving Predictions Today: Offline Scoring



Offline Batch System

Serving Predictions Today: Offline Scoring

Lookup decision in KV-store



Offline Serving System

Serving Predictions Today: Offline Scoring

Problems

Serving Predictions Today: Offline Scoring

Problems

- Requires full set of queries ahead of time
  - Small and bounded input domain

Serving Predictions Today: 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

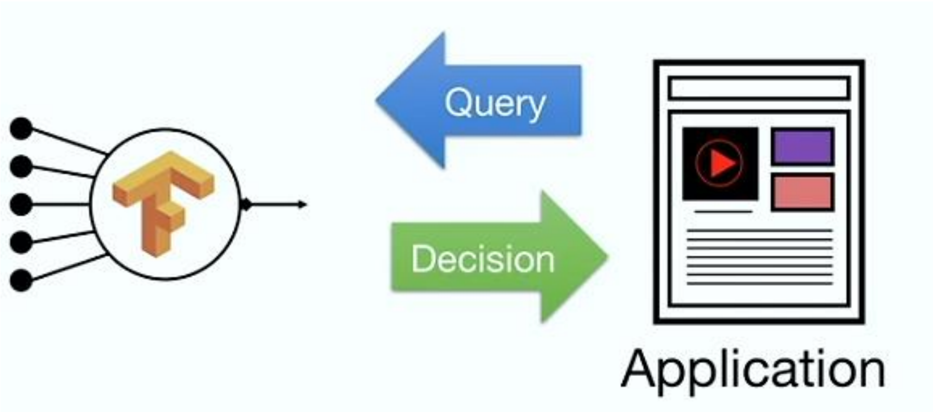
Serving Predictions Today: Offline Scoring

Problems

- Requires full set of queries ahead of time
  - Small and bounded input domain
- Wasted computation and space
  - Can render and store unneeded predictions
- No feedback and costly to update

Serving Predictions Today: Online Scoring

Render prediction with model in real-time

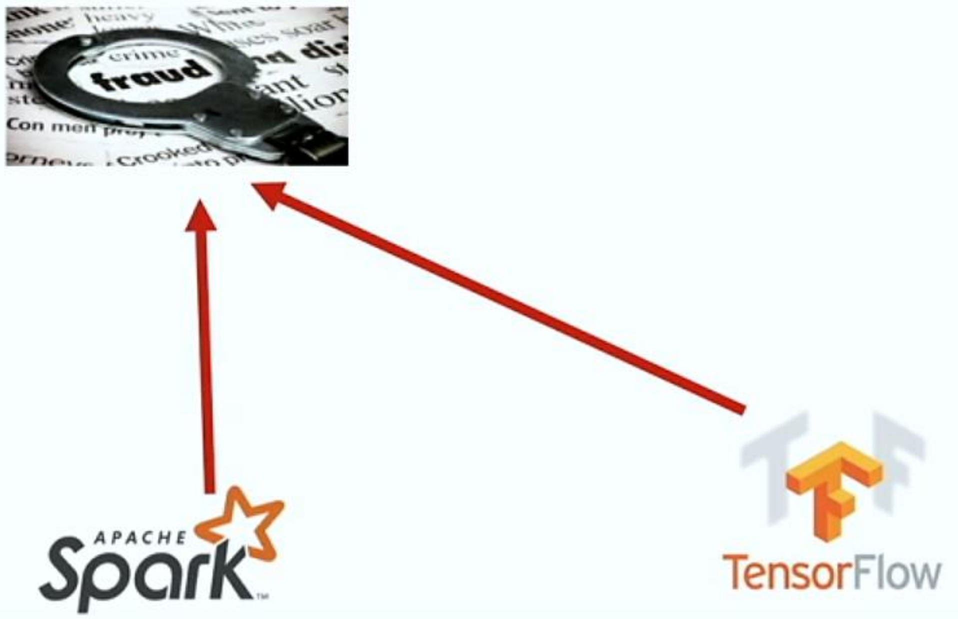


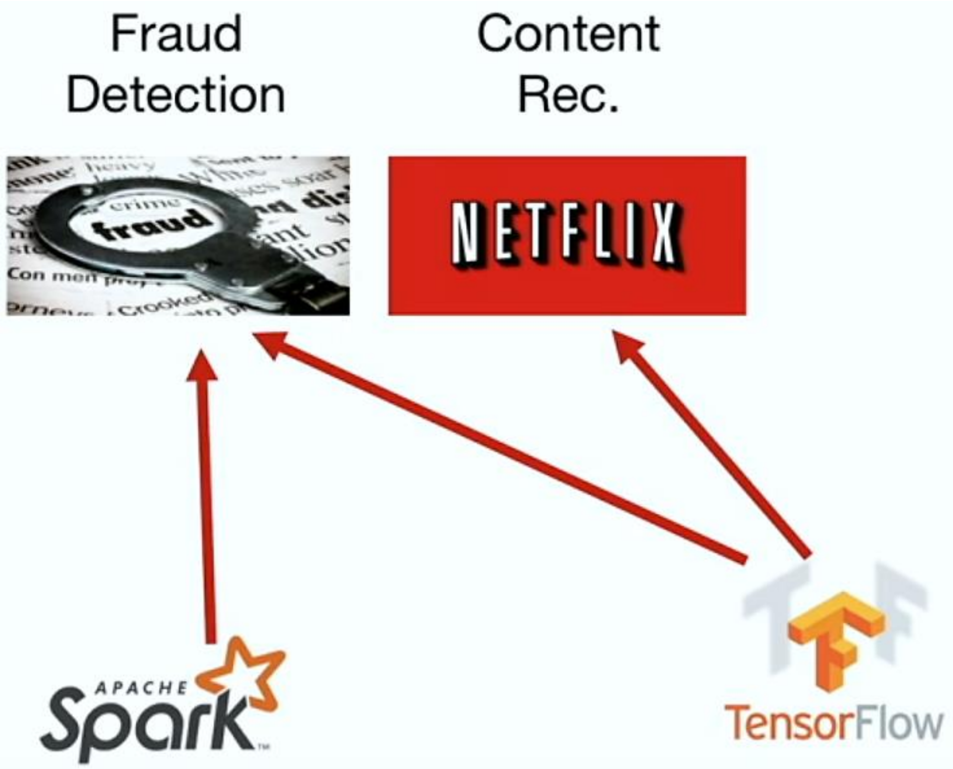
Online Serving System

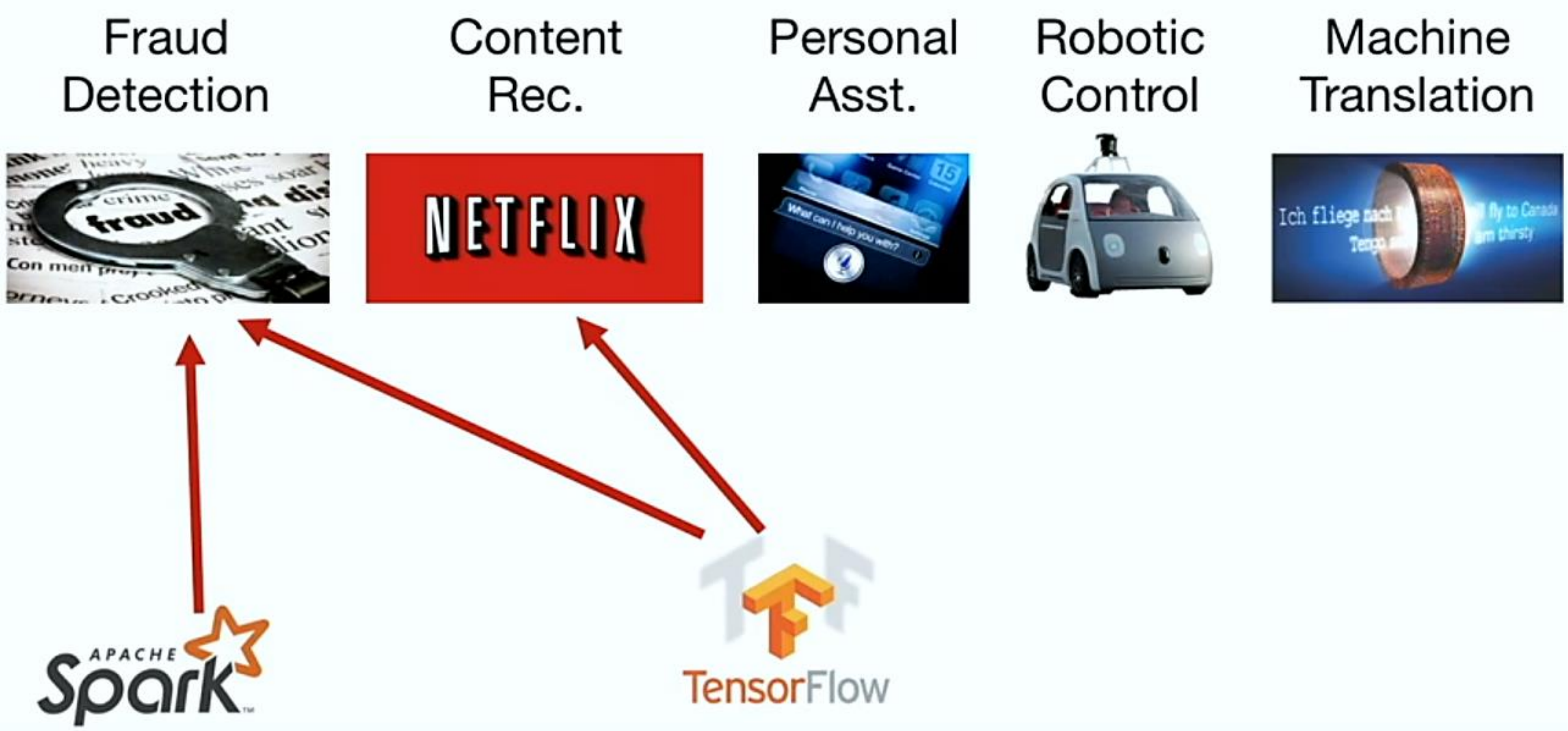
Fraud Detection

APACHE Spark

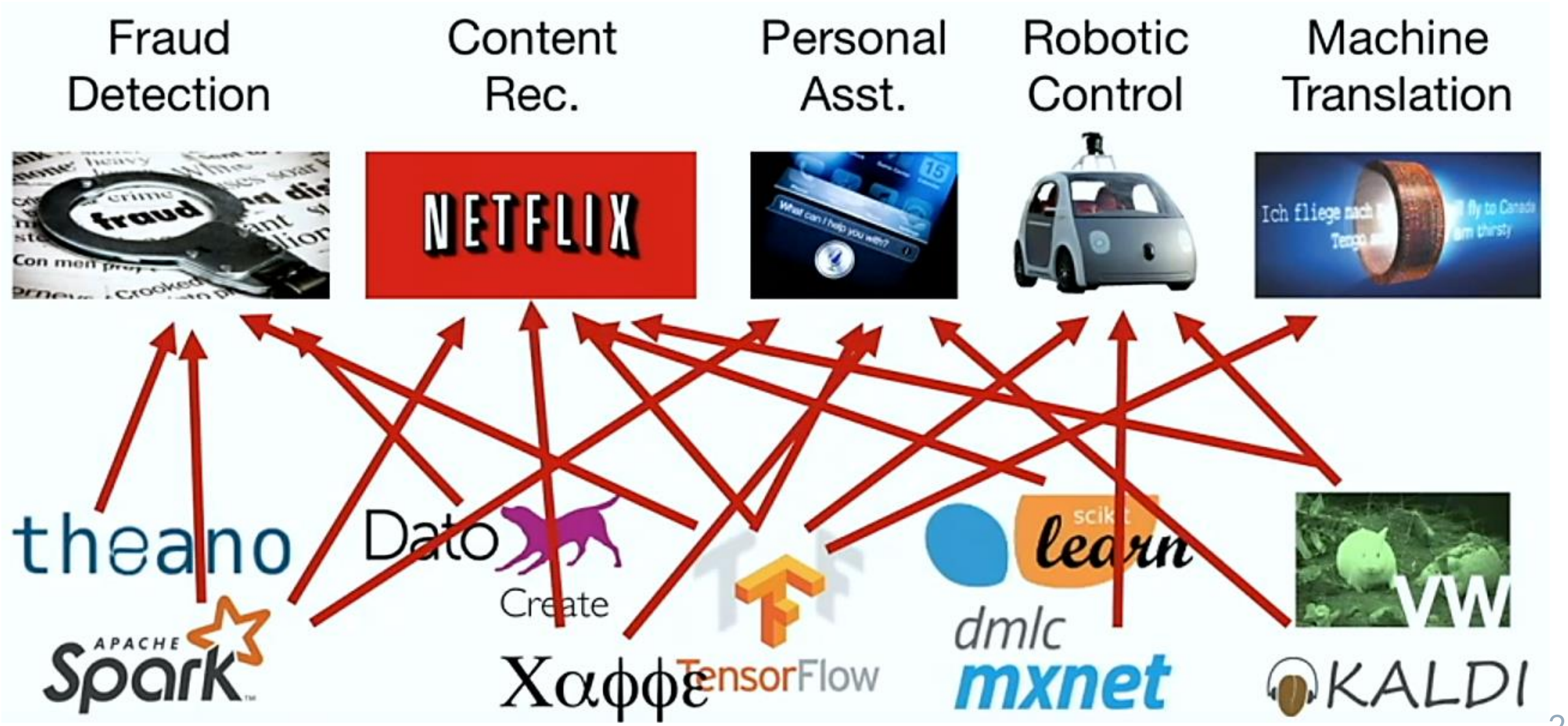
Fraud Detection



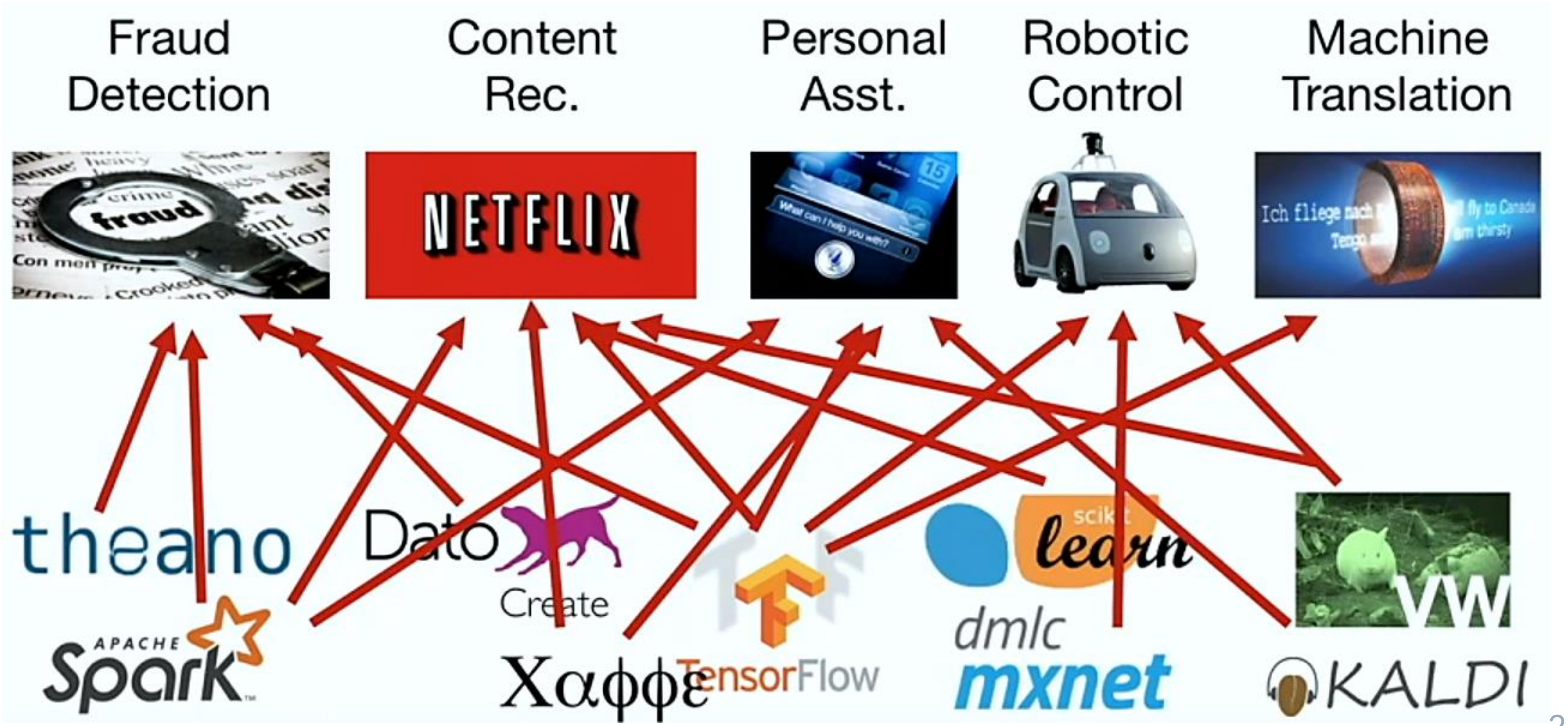

Fraud
Detection
Content
Rec.
fraud
NETFLIX
APACHE
Spark
TensorFlow

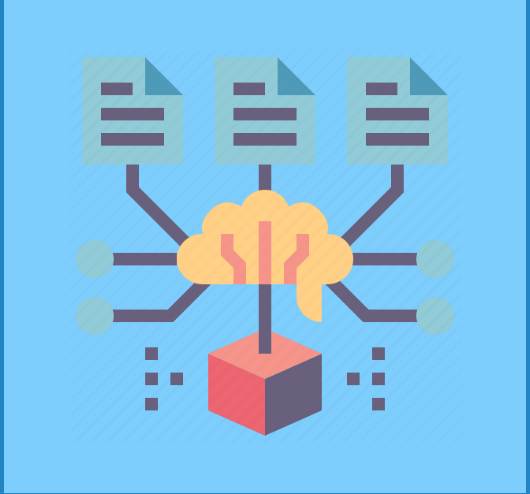

Fraud Detection
Content Rec.
Personal Asst.
Robotic Control
Machine Translation
fraud
NETFLIX
What can I help you with?
APACHE Spark
TensorFlow

Many applications and many models



Can we **decouple** models and applications?





Prediction Serving System

A decoupling system for models and applications

Requirements

Requirements

- Decouple applications from models and allow them to evolve independently from each other

Requirements

- Decouple applications from models and allow them to evolve independently from each other
- For the frontend developer perspective: focus on bilding reliable, low latency applications

Requirements

- Decouple applications from models and allow them to evolve independently from each other
- For the frontend developer perspective: focus on bilding reliable, low latency applications
  - Provide stable, reliable, performant APIs to meet SLAs (Service Level Agreements)
    - Scale system, hardware to meet application demands

Requirements

- Decouple applications from models and allow them to evolve independently from each other
- **For the frontend developer perspective**: focus on bilding reliable, low latency applications
  - Provide stable, reliable, performant APIs to meet SLAs (Service Level Agreements)
    - Scale system, hardware to meet application demands
  - Oblivious to the implementations of underlying models

Requirements

- Decouple applications from models and allow them to evolve independently from each other
- **For the frontend developer perspective**: focus on bilding reliable, low latency applications
  - Provide stable, reliable, performant APIs to meet SLAs (Service Level Agreements)
    - Scale system, hardware to meet application demands
  - Oblivious to the implementations of underlying models
- **The Data Scientist perspective:** focus on making accurate predictions

Requirements

- Decouple applications from models and allow them to evolve independently from each other
- **For the frontend developer perspective**: focus on bilding reliable, low latency applications
  - Provide stable, reliable, performant APIs to meet SLAs (Service Level Agreements)
    - Scale system, hardware to meet application demands
  - Oblivious to the implementations of underlying models
- **The Data Scientist perspective:** focus on making accurate predictions
  - Support many models and frameworks simultaneously

Requirements

- Decouple applications from models and allow them to evolve independently from each other
- **For the frontend developer perspective**: focus on bilding reliable, low latency applications
  - Provide stable, reliable, performant APIs to meet SLAs (Service Level Agreements)
    - Scale system, hardware to meet application demands
  - Oblivious to the implementations of underlying models
- **The Data Scientist perspective:** focus on making accurate predictions
  - Support many models and frameworks simultaneously
  - Simple deployment and online experimentation

Requirements

- Decouple applications from models and allow them to evolve independently from each other
- **For the frontend developer perspective**: focus on bilding reliable, low latency applications
  - Provide stable, reliable, performant APIs to meet SLAs (Service Level Agreements)
    - Scale system, hardware to meet application demands
  - Oblivious to the implementations of underlying models
- **The Data Scientist perspective:** focus on making accurate predictions
  - Support many models and frameworks simultaneously
  - Simple deployment and online experimentation
  - Oblivious to system performance and workload demands

2.

CLIPPER

From the Frontend Dev perspective

Clipper

From the Frontend Dev perspective

From the Frontend Dev perspective

- Management REST API

From the Frontend Dev perspective

- Management REST API

  create_application

  deploy_model()

  replicate_model()

  inspect_instance()

From the Data Scientist perspective

From the Data Scientist perspective

- Implement Model API

From the Data Scientist perspective

- Implement Model API

```
class ModelContainer:
    def__init__(model_data)
```

From the Data Scientist perspective

- Implement Model API

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

From the Data Scientist perspective

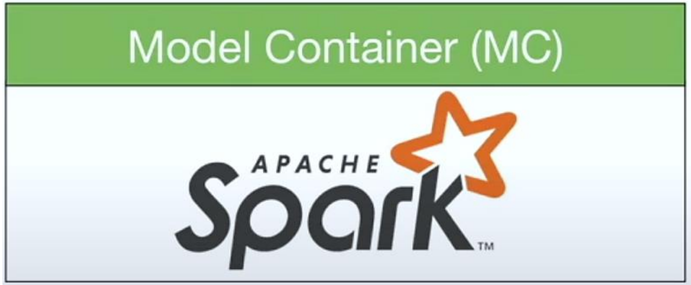
- Implement Model API

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

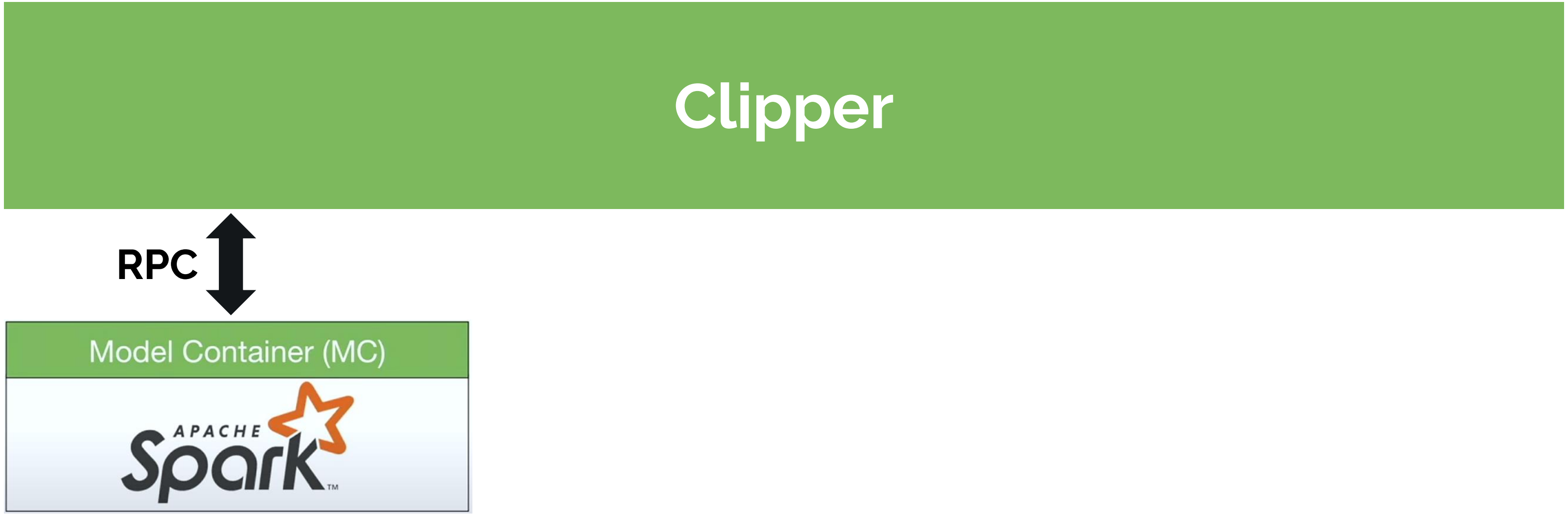
- Implemented in many languages
  - Python
  - Java
  - C/C++
  - R
  - ...

From the Data Scientist perspective

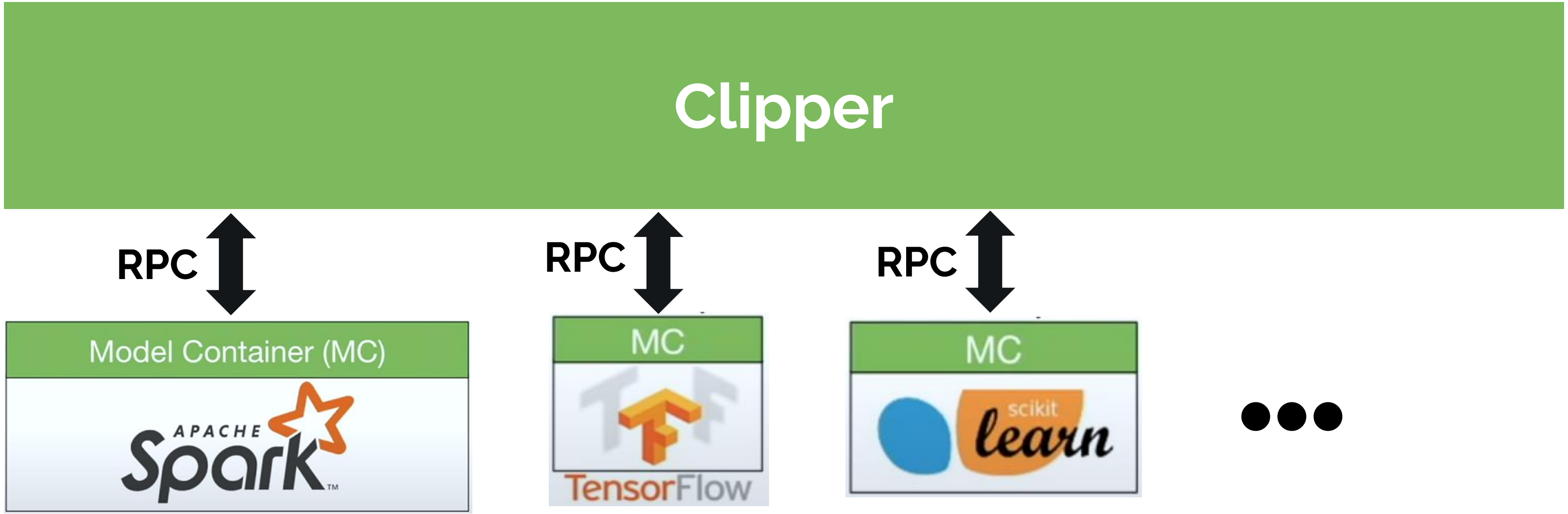
- Model implementation packaged in a container

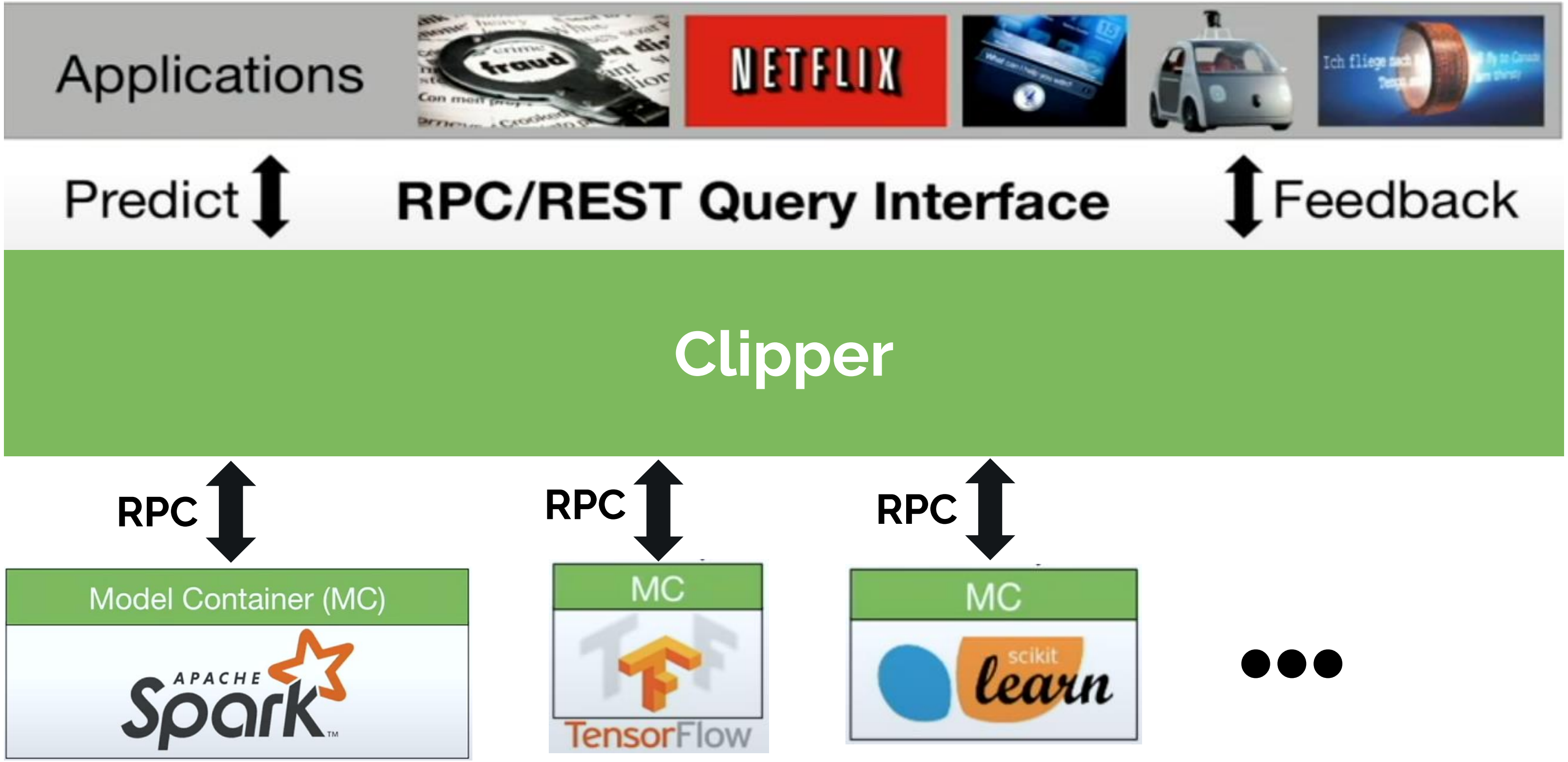


From the Data Scientist perspective



From the Data Scientist perspective




Applications
NETFLIX
Predict
RPC/REST Query Interface
Feedback
Clipper
RPC
RPC
RPC
Model Container (MC)
APACHE Spark
MC
TensorFlow
MC
scikit learn

Challenges

Challenges

- Managing heterogeneity everywhere
  - Different type of models (different software, different resource requirements) in a production environment

Challenges

- Managing heterogeneity everywhere
  - Different type of models (different software, different resource requirements) in a production environment
  - Different workloads and latencies etc.

Challenges

- Managing heterogeneity everywhere
    - Different type of models (different software, different resource requirements) in a production environment
    - Different workloads and latencies etc.
- Scheduling (space-time resource management)
    - where and when to send prediction queries to models

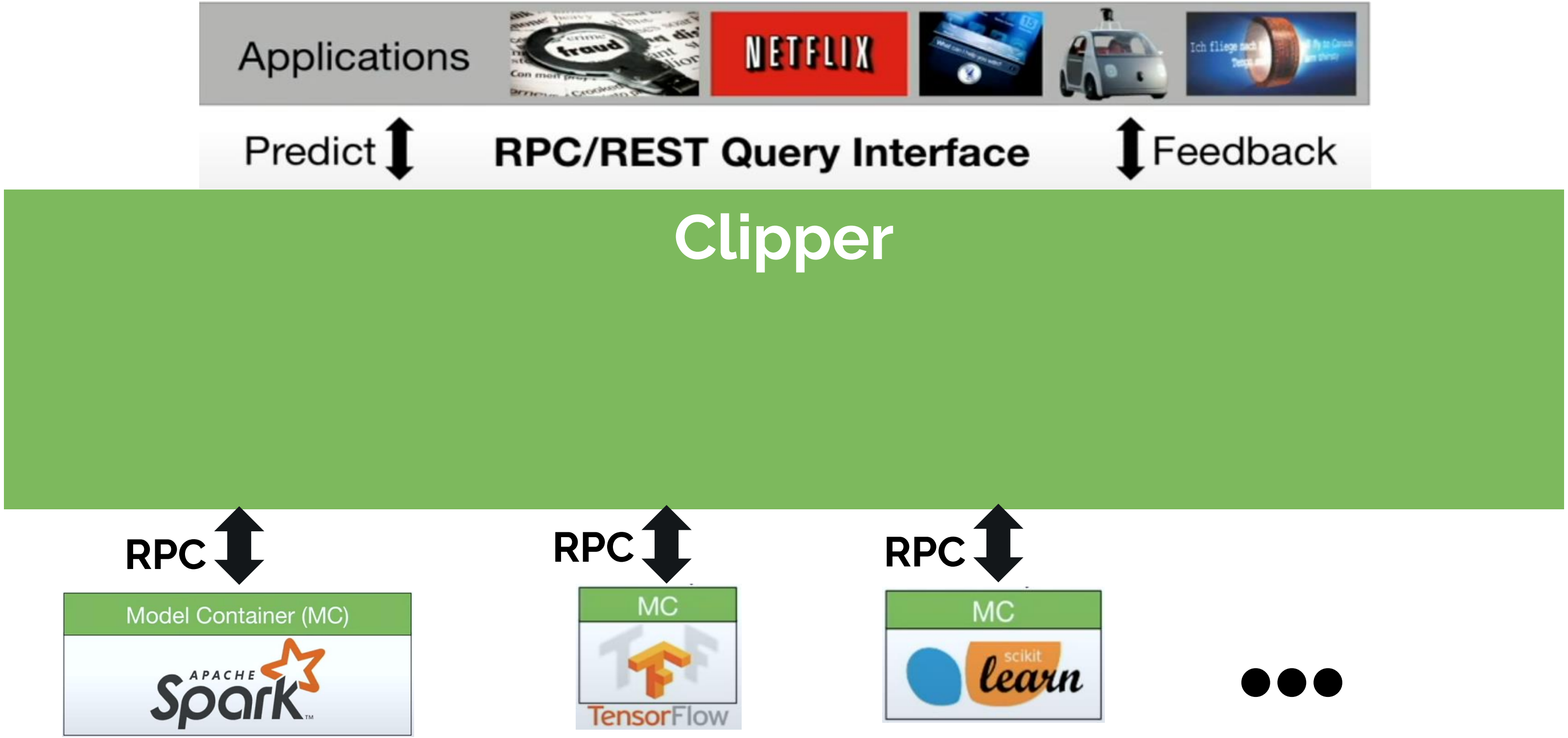
Challenges

- Managing heterogeneity everywhere
  - Different type of models (different software, different resource requirements) in a production environment
  - Different workloads and latencies etc.
- Scheduling (space-time resource management)
  - where and when to send prediction queries to models
- Latency-accuracy tradeoffs
  - Marginal utility of allocating additional resources

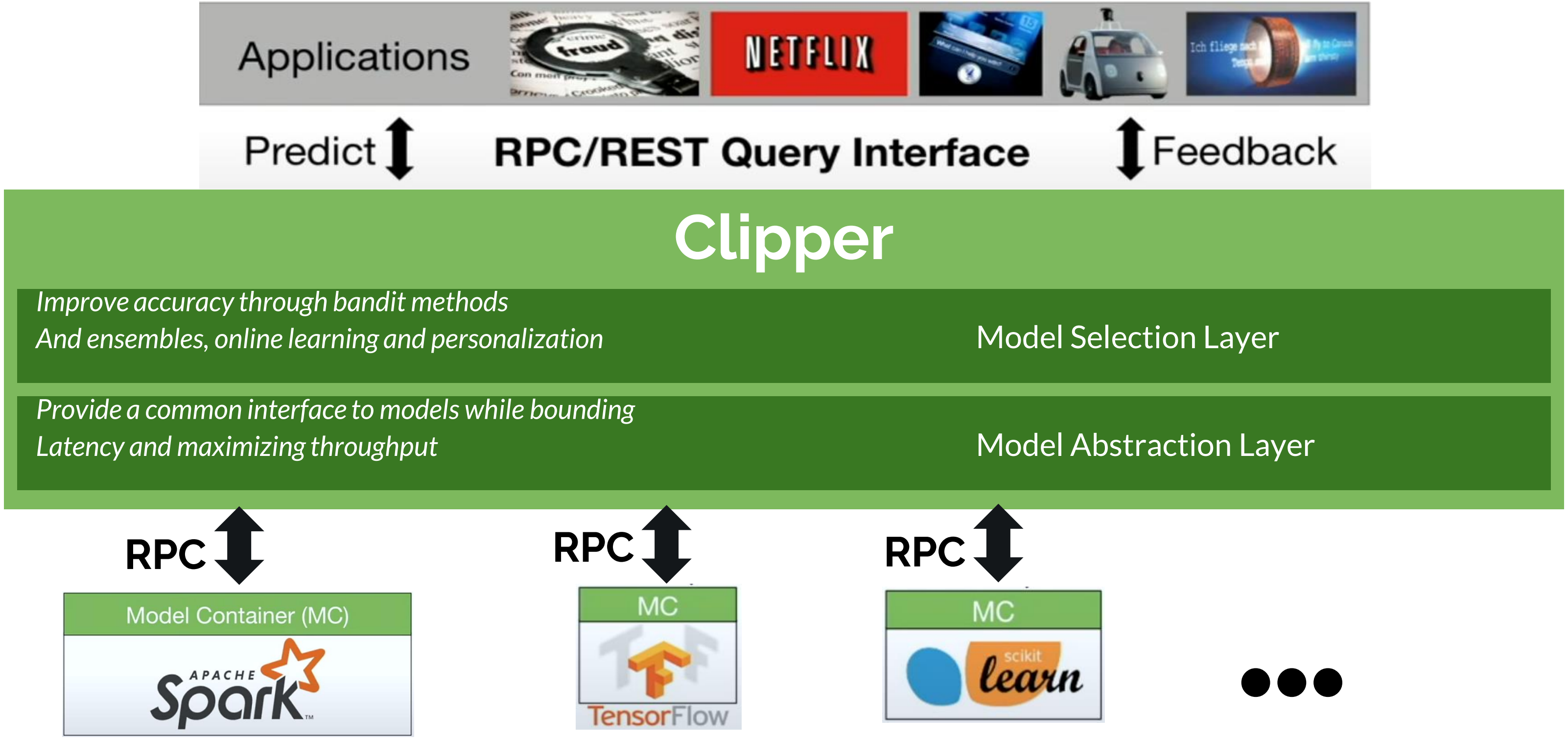
Challenges

- Managing heterogeneity everywhere
  - Different type of models (different software, different resource requirements) in a production environment
  - Different workloads and latencies etc.
- Scheduling (space-time resource management)
  - where and when to send prediction queries to models
- Latency-accuracy tradeoffs
  - Marginal utility of allocating additional resources
- How to use feedback to improve accuracy in real-time

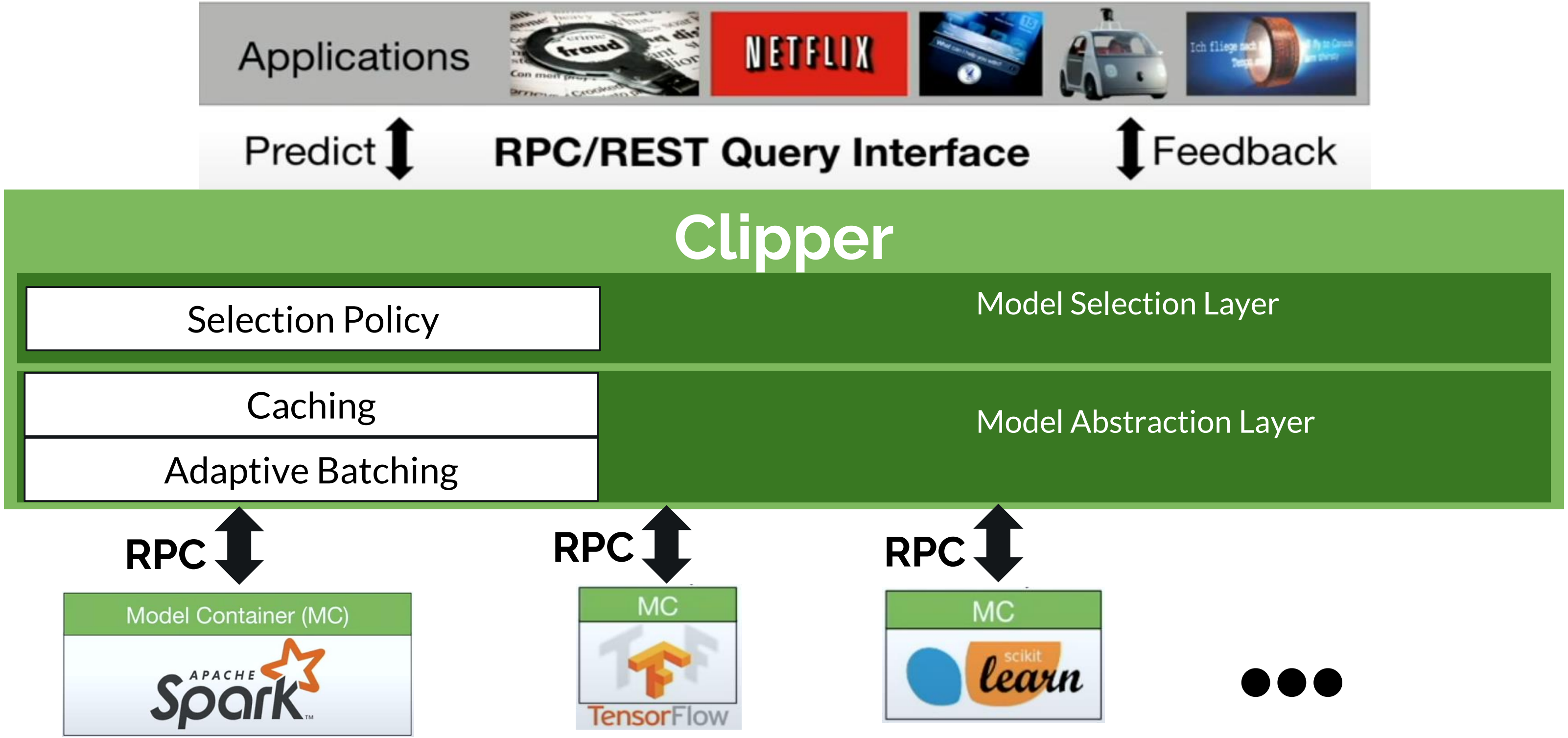
Clipper Architecture

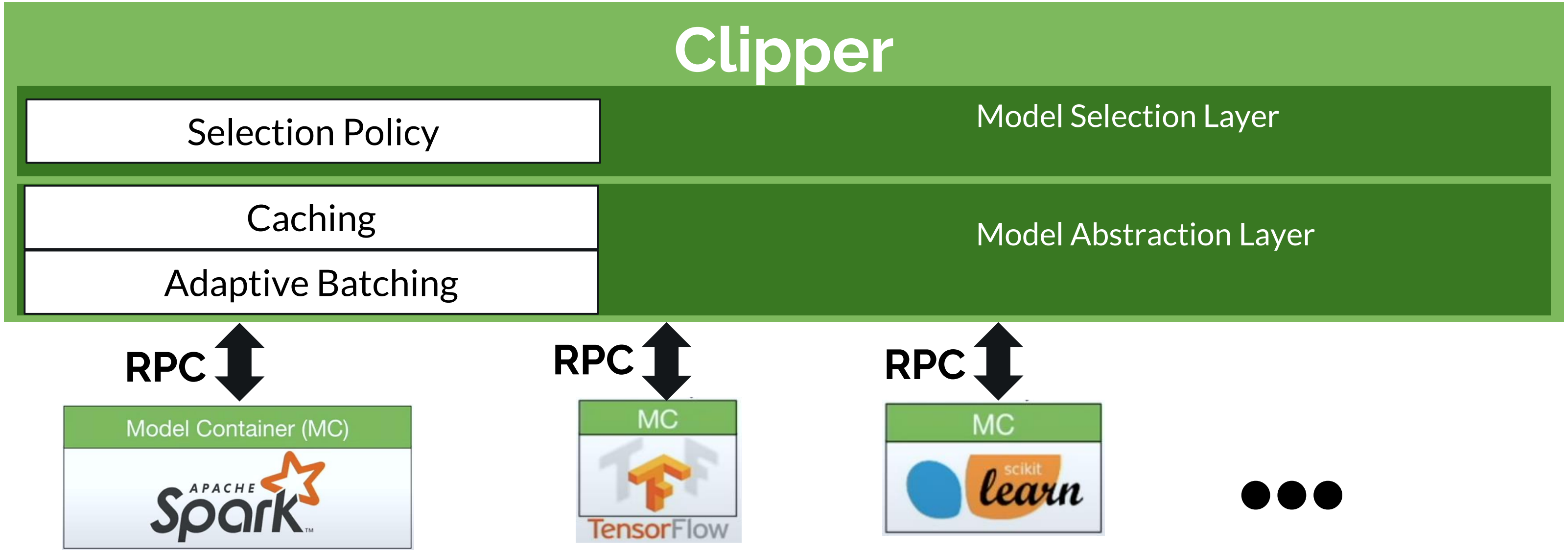


Clipper Architecture



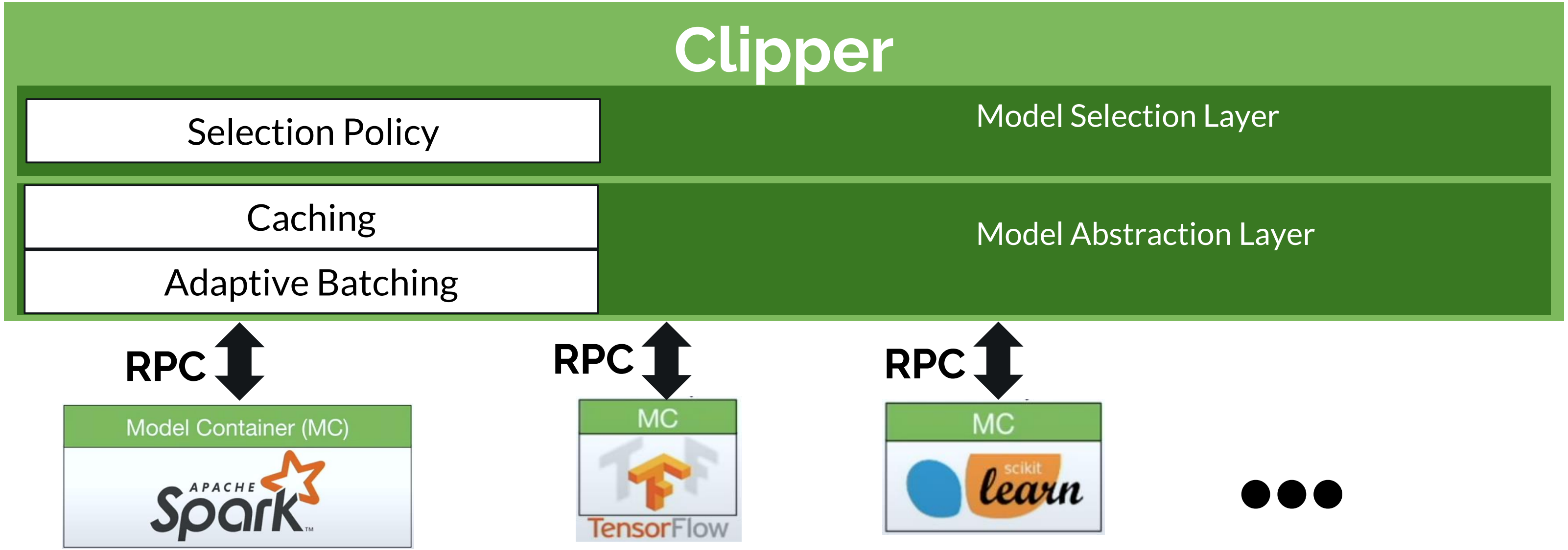
Clipper Architecture





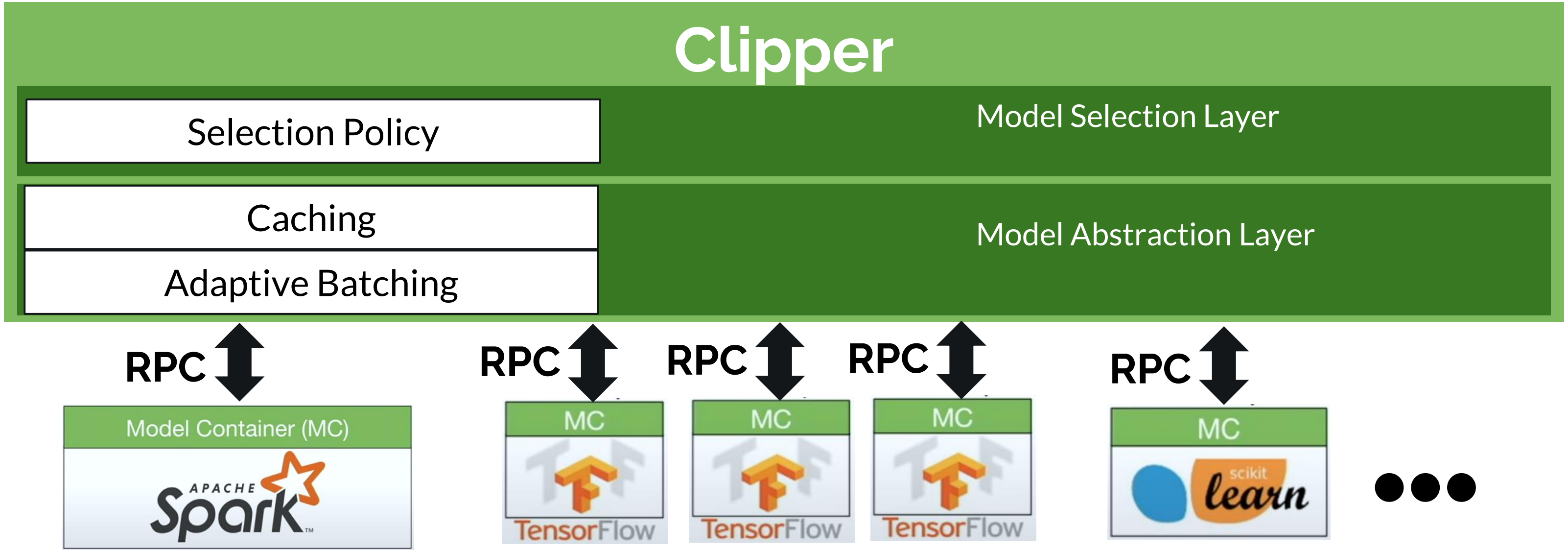
Common interface simplifies deployment:

- Evaluate models using original code and systems



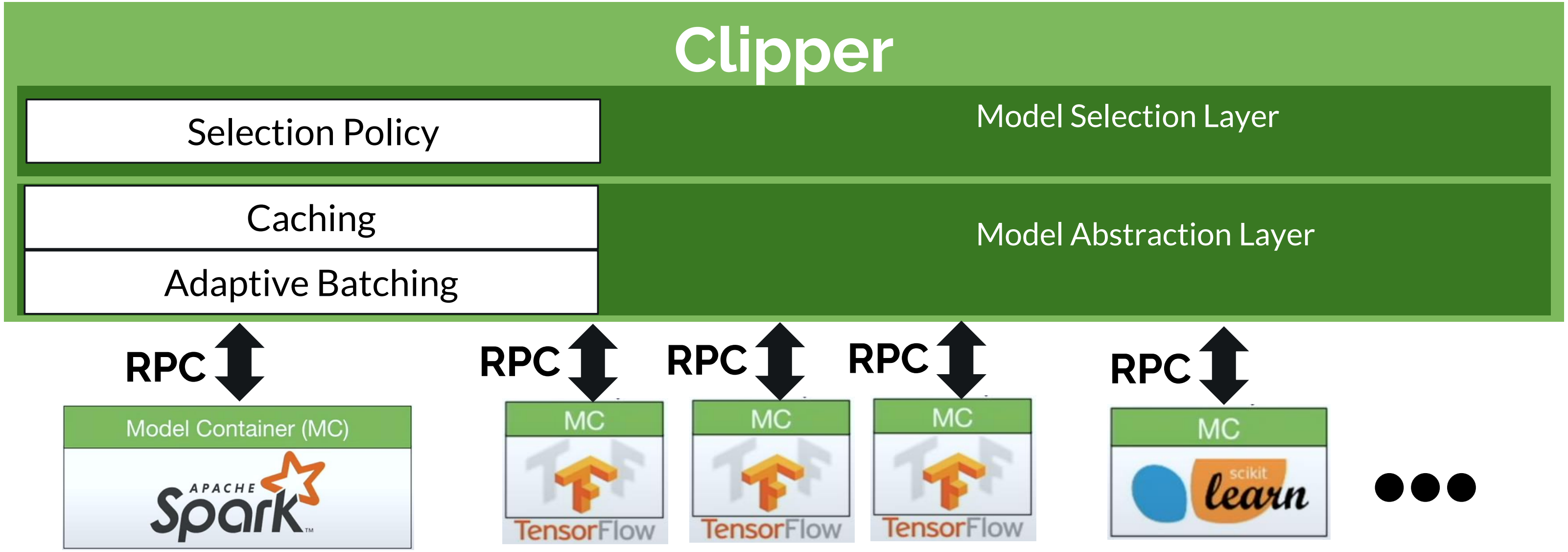
Common interface simplifies deployment:

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



Common interface simplifies deployment:

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



Common interface simplifies deployment:

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

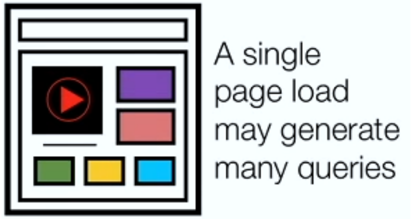
Problem: frameworks optimized for batch processing not latency

Adaptive Batching to improve throughput

- Why batching helps:

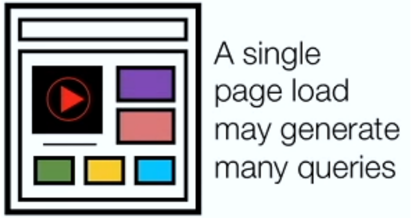
Adaptive Batching to improve throughput

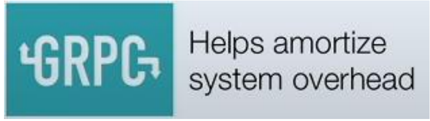
- Why batching helps:



Adaptive Batching to improve throughput

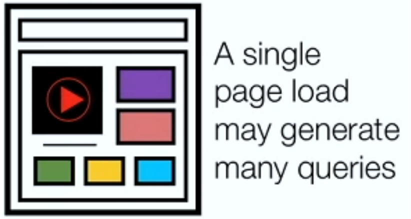
- Why batching helps:

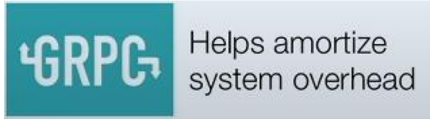




Adaptive Batching to improve throughput

- Why batching helps:

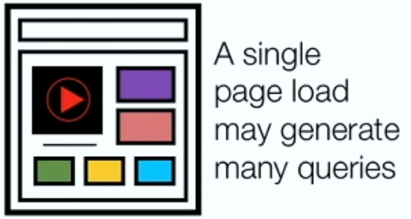




- Optimal batch depends on:
    - Hardware configuration
    - Model and framework
    - System load

Adaptive Batching to improve throughput

- Why batching helps:



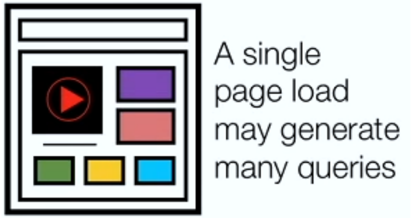
- Optimal batch depends on:
  - Hardware configuration
  - model and framework
  - System load

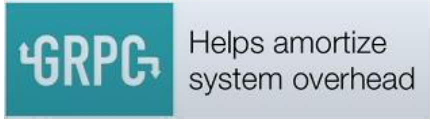
Clipper Solution:

*be as **slow** as **allowed**...*

Adaptive Batching to improve throughput

- Why batching helps:





- Optimal batch depends on:
  - Hardware configuration
  - model and framework
  - System load

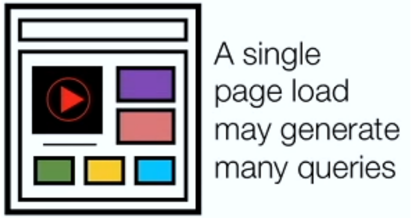
Clipper Solution:

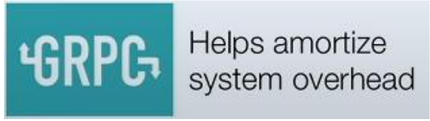
*be as **slow** as **allowed**...*

- Increase the batch size until the latency objective is exceeded

Adaptive Batching to improve throughput

- Why batching helps:





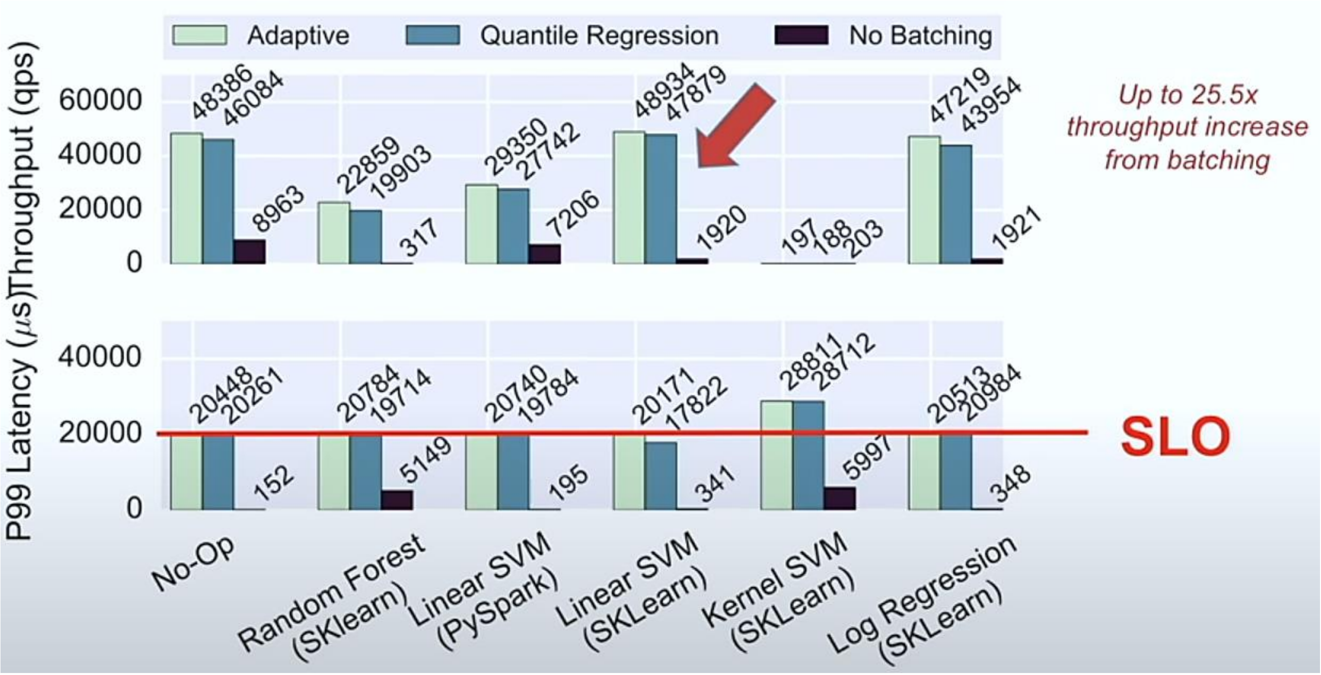
- Optimal batch depends on:
  - Hardware configuration
  - model and framework
  - System load

Clipper Solution:

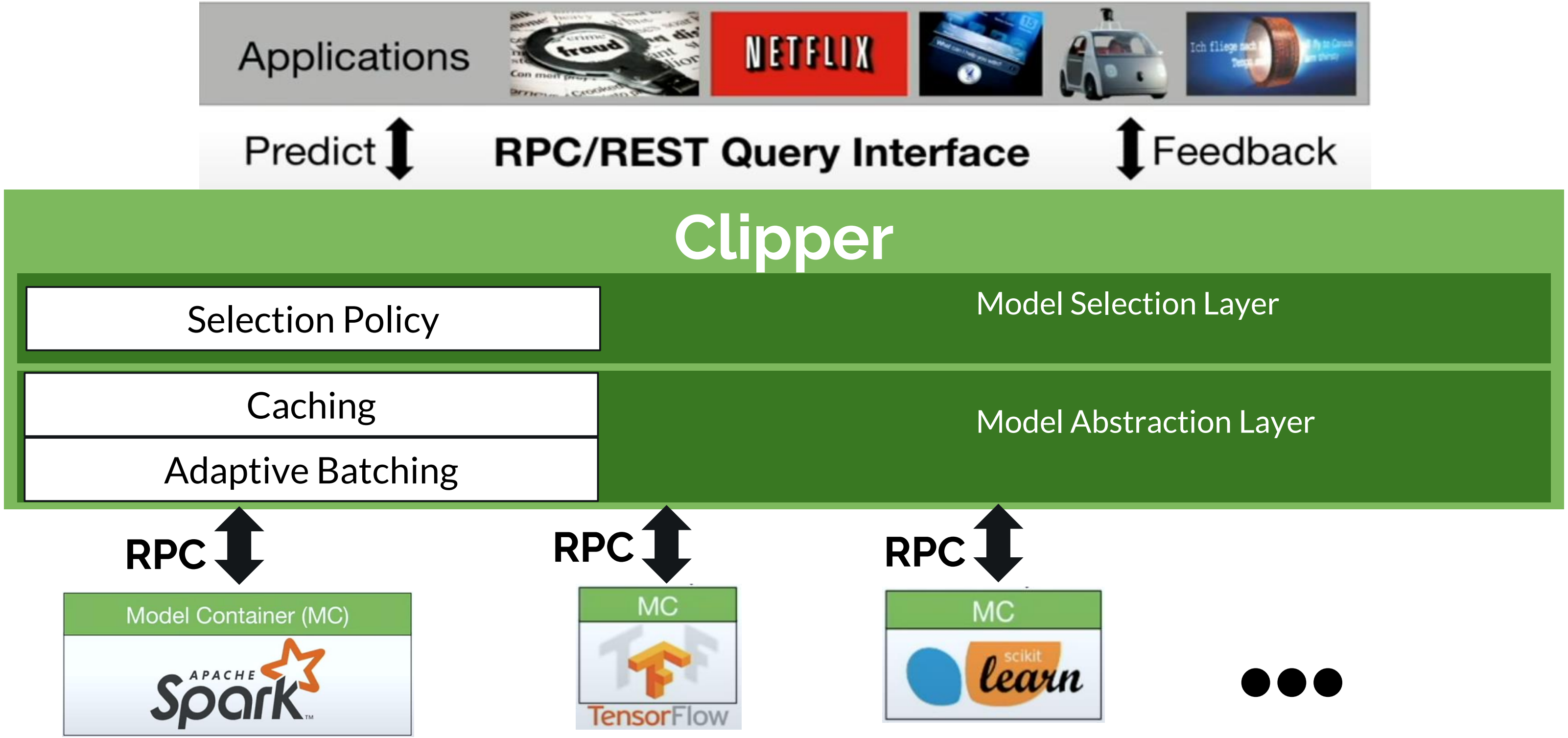
*be as **slow** as **allowed**…*

- Increase the batch size until the latency objective is exceeded
- If latency exceeds SLO cut batch size by a fraction

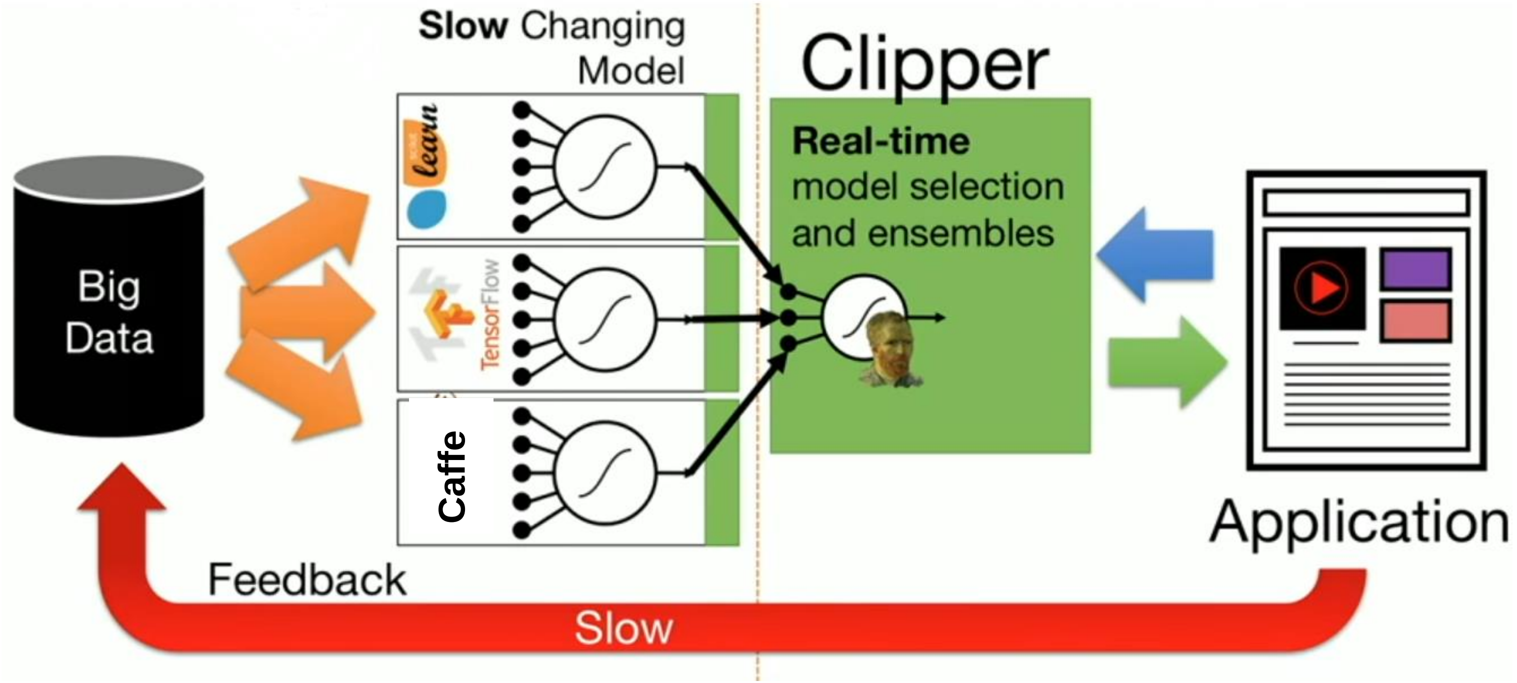
Batching Results



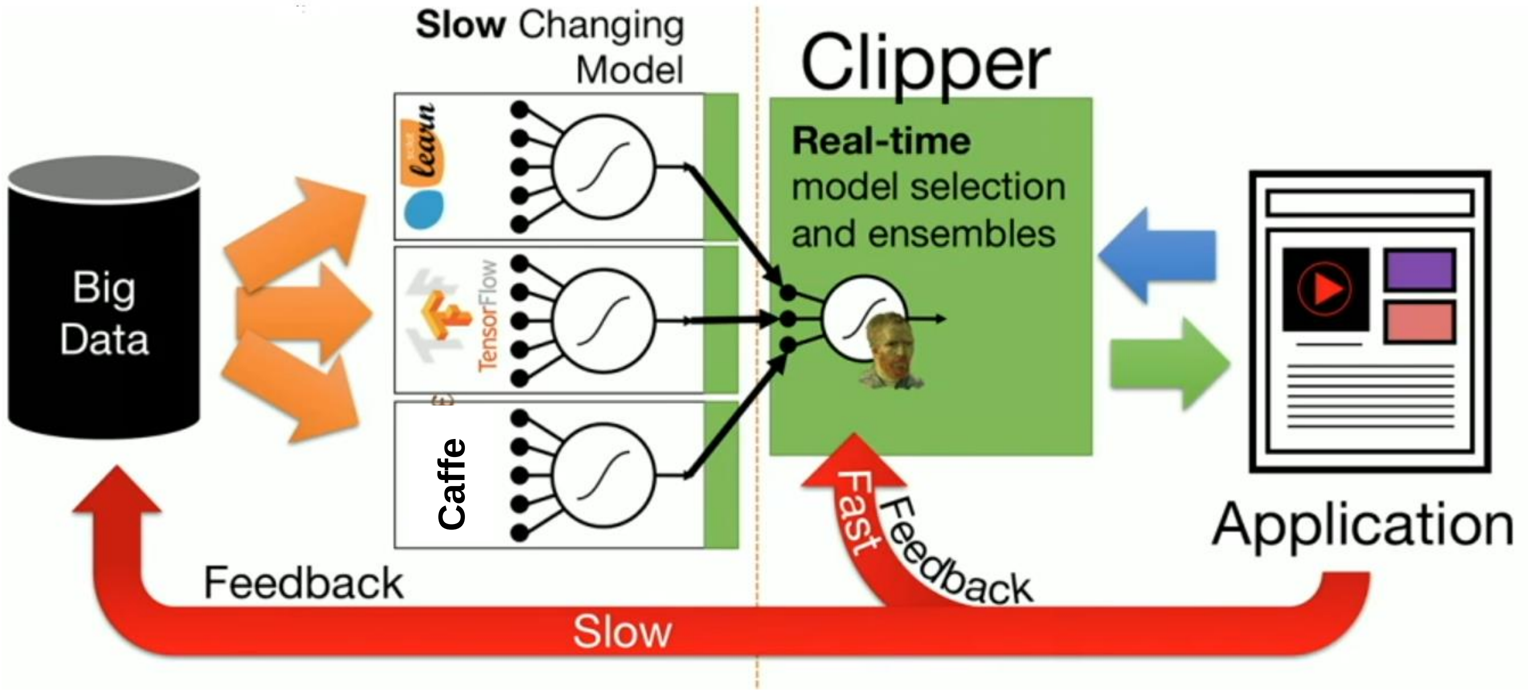
Clipper Architecture



Learning Inference



Learning Inference

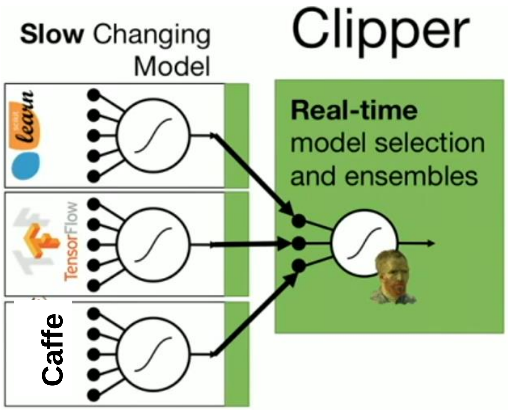


Bring leaning into the Serving Tier

Clipper

Selection Policy

Model Selection Layer

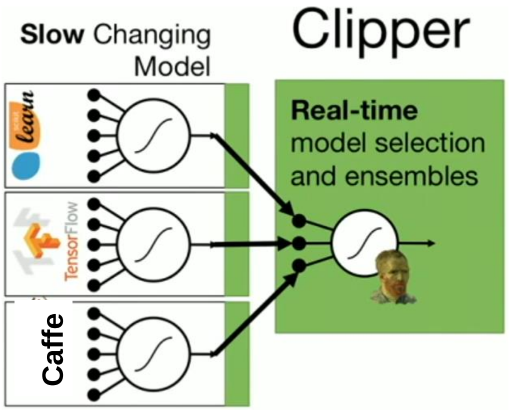


Bring leaning into the Serving Tier

Clipper

Selection Policy | Model Selection Layer

What can we learn?



Bring leaning into the Serving Tier

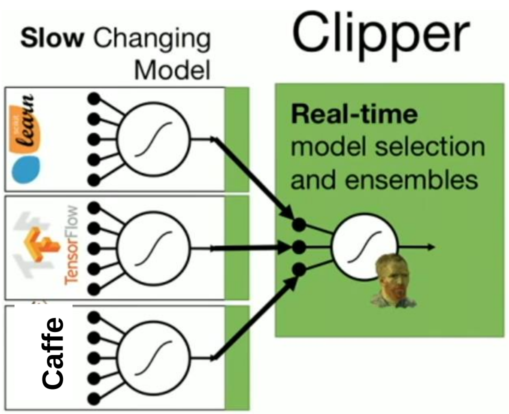
Clipper

Selection Policy

Model Selection Layer

What can we learn?

- Dynamically weight mixture of experts



Bring leaning into the Serving Tier

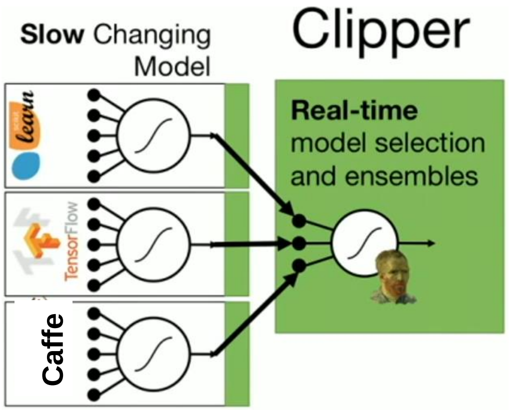
Clipper

Selection Policy

Model Selection Layer

What can we learn?

- Dynamically weight mixture of experts
- Select best model for each user



Bring leaning into the Serving Tier

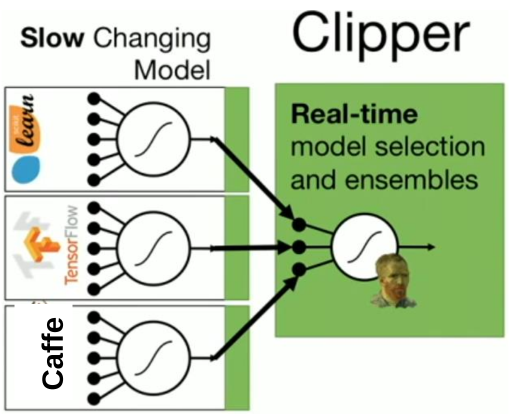
Clipper

Selection Policy

Model Selection Layer

What can we learn?

- Dynamically weight mixture of experts
- Select best model for each user
- Use ensemble to estimate prediction confidence



Bring leaning into the Serving Tier

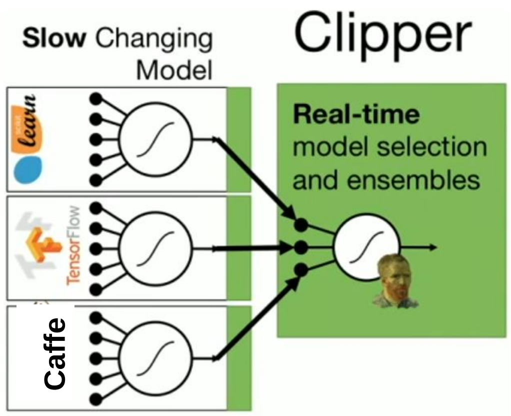
Clipper

Selection Policy

Model Selection Layer

What can we learn?

- Dynamically weight mixture of experts
- Select best model for each user
- Use ensemble to estimate prediction confidence
- Don't try to retrain models



3.
RELATED WORK

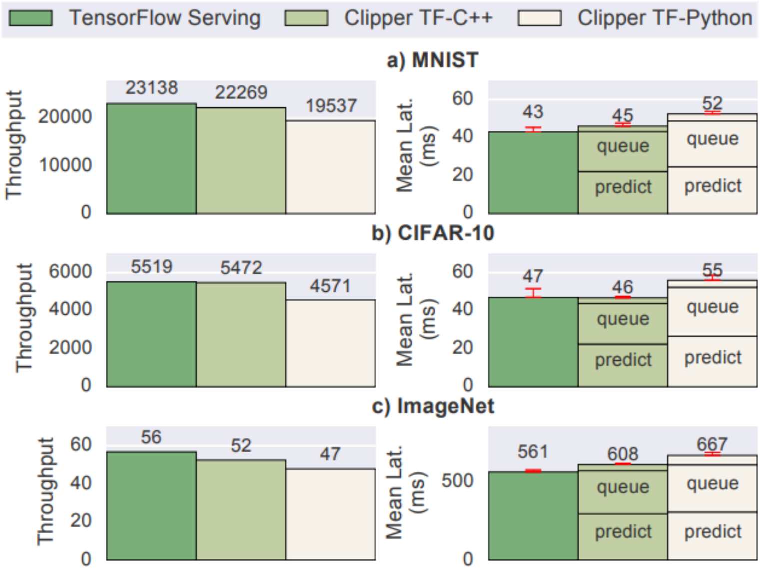
Related Work

- The closest project to clipper are LASER
- Velox is a UC Berkley research project to study personalized prediction serving with Spark
- TensorFlow Serving is the open-source prediction serving system
- Author only compare with TensorFlow, since LASER is not publically available, and the current prototype of Velox has very limited functionality

Comparison with TensorFlow Serving

- TensorFlow Serving also employs batching
- Batch sizes in TensorFlow Serving are static
- TF-serving does not explicitly incorporate prediction latency objectives
- TF-serving does not directly support feedback, dynamic mode selection and composition

Comparison with TensorFlow Serving



By achieving comparable performance across this range of models, we have demonstrated that through careful design and implementation of the system, the modular architecture and substantially broader set of features in Clipper do not come at a cost of reduced performance on core prediction-serving tasks.

▷ Despite Clipper's modular design, we are able to achieve comparable throughput to TensorFlow Serving

4.

CONCLUSION

Final Remarks

- Identified three key challenges of prediction serving: latency, throughput and accuracy
- Proposed a new layered architecture to address these challenges
- New ML frameworks and models can be introduced without modifying end-user applications
- The model abstraction layer lifts caching and adaptive batching strategies above the machine learning frameworks to achieve up to a 26x improvement in throughput

Final Remarks

- The model selection layer enables many models to be deployed concurrently and then dynamically selects and combines predictions from each model to render more robust, accurate, and contextualized predictions
- Compared Clipper to Google's TensorFlow Serving system and achieved parity on throughput and latency performance,

Limitations

- Clipper does not optimize the execution of the models within their respective machine learning frameworks
- Similarly, Clipper does not manage the training or retraining of the base models within their respective frameworks

Most of these limitations follow directly from the design of the Clipper architecture which assumes models are below Clipper in the software stack, and thus are treated as black-box components.

Thanks!

Any questions?

You can find me at:

@username

raja@rit.kaist.ac.kr