Good morning!

CS 744: CLIPPER

Shivaram Venkataraman

Fall 2020

ADMINISTRIVIA

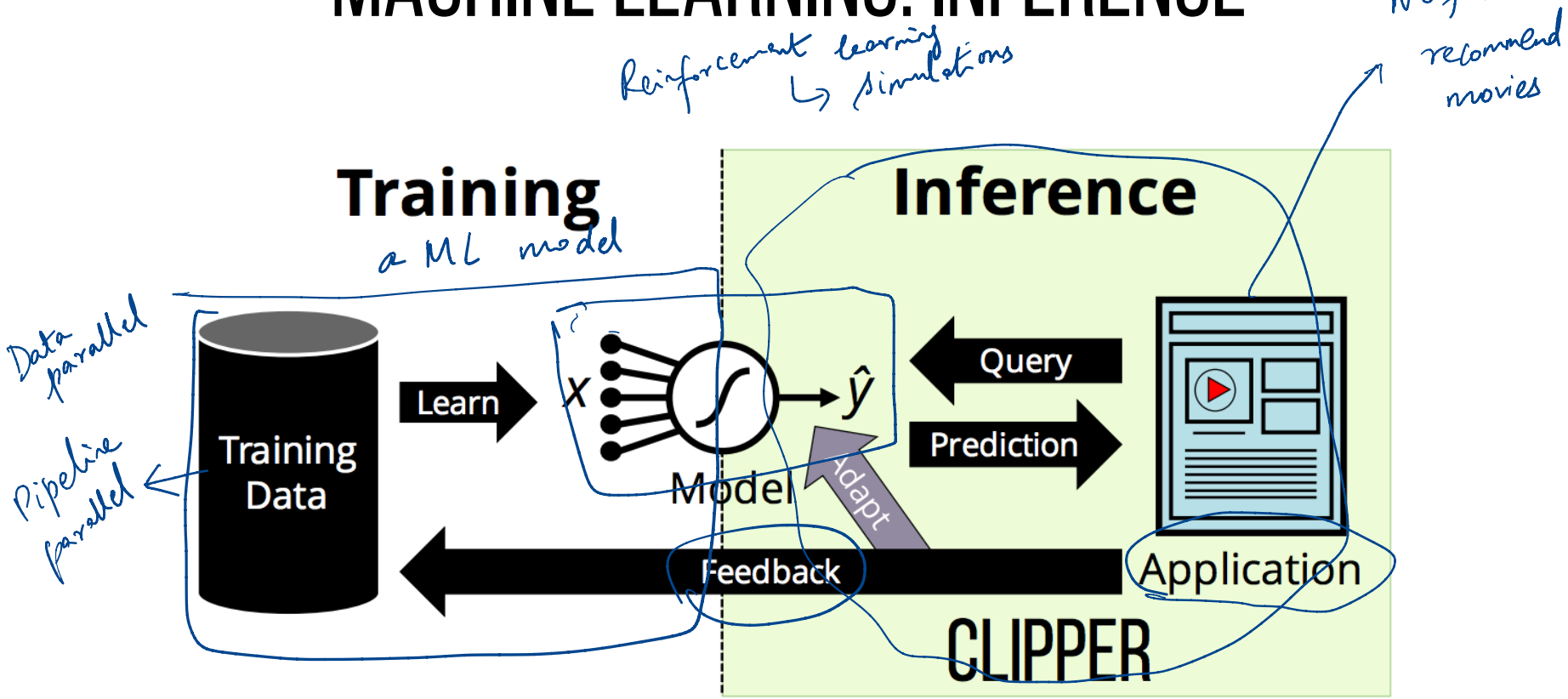
Course Project Proposals

- Due on Friday! →
- See Piazza for template
- Submission instructions soon

Midterm details → Oct 22 → Up to ML section | sample midterms on Piazza

- Open book, open notes
- Held in class time 9.30-10.45am Central Time
- Type / Upload photos (extra 15 mins)

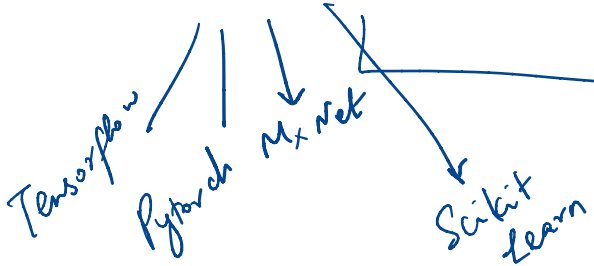
MACHINE LEARNING: INFERENCE



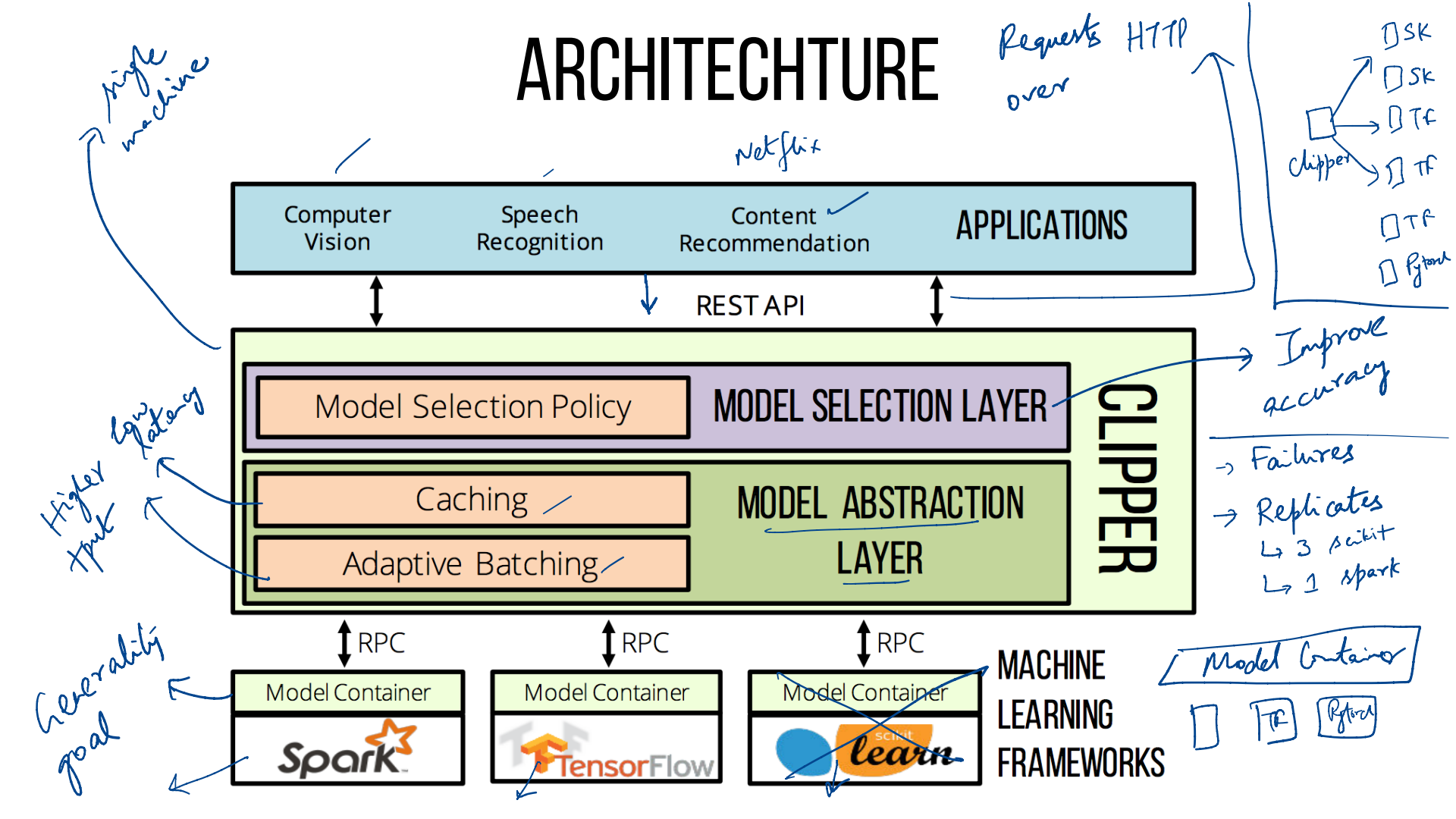
GOALS

web server ↳ chat

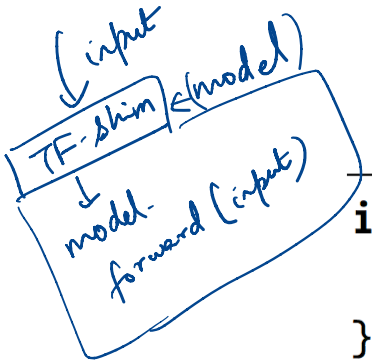
- Interactive latencies (tail latency < 100ms) → 99 percentile or 99.9th percentile → low latency inference
- High throughput to handle load → many users or many requests that need to be made
- Improved prediction accuracy → ML specific
- Generality (?) → handle as many ML models / frameworks as possible



ARCHITECHTURE



MODEL CONTAINERS



simple API

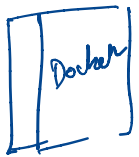
```
interface Predictor<X,Y> {
  List<List<Y>> pred_batch(List<X> inputs);
}
```

Interface is implemented once per framework

Run using Docker containers

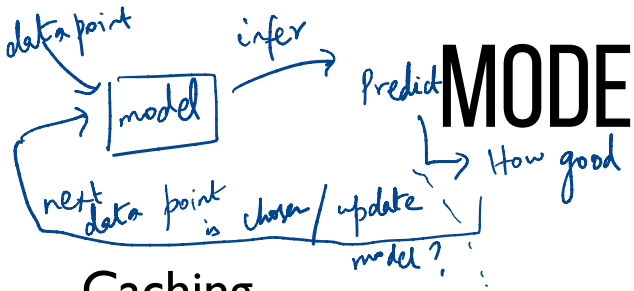
Can be replicated across machines

instantiate TF slim with Resnet-18



instantiate TF slim with Resnet-50

Model frameworks are a black box to clipper!



MODEL ABSTRACTION LAYER

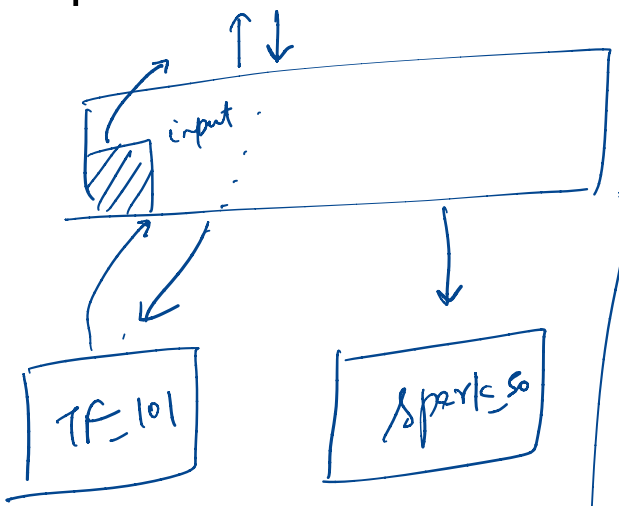
Caching

- Improve performance for frequent queries
- LRU eviction policy
- Important for feedback

high dim inputs
↳ cache hits are less likely!

spec provided by the dev.

→ Predict (1) movies for user_id_1 movies are TF_101 or Spark_50



Feedback
(1) Predict Movies for user_1
(2) User rates movie_1
Feedback and Predictions → Model update

To do an RPC
↳ fixed cost overhead
→ amortize

BATCHING, QUEUING

batch size that max. tput while within latency SLO.

each model container has diff overheads, parallelism

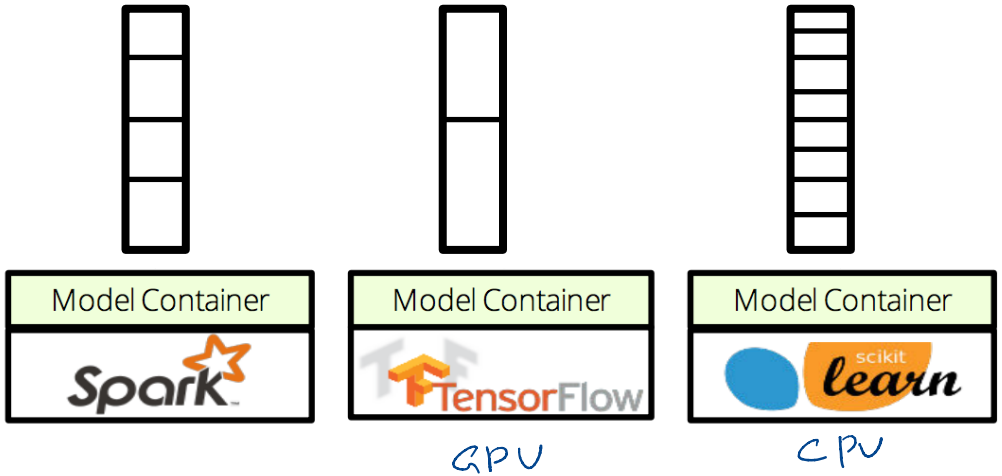
Optimal batch size could vary for each model!

Goals, Insight

- Increase latency (within SLO) for improved throughput → batches use hardware parallelism
- Reduce RPC overheads
- GPU / BLAS acceleration

Approach

- Per container queues.
- Why?



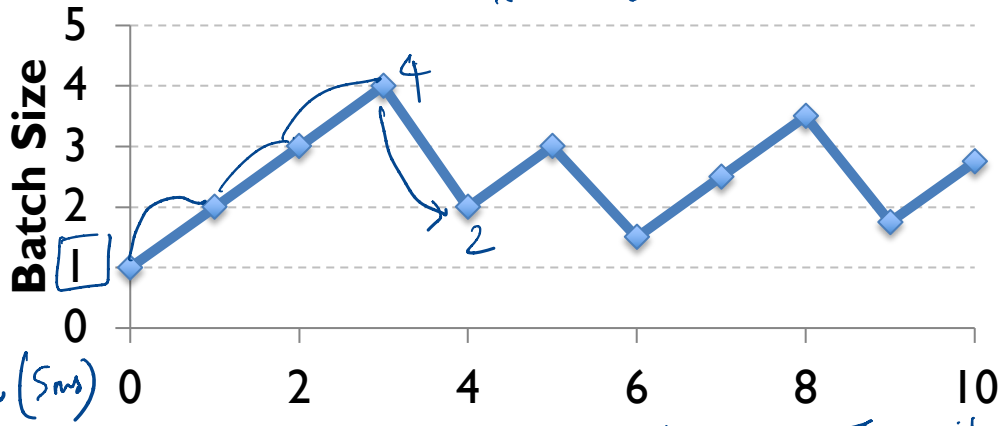
ADAPTIVE BATCHING

AIMD: Additive Inc Multiplicative Dec (AIMD)

Why ?

Increase batch size carefully & latency quickly comes down if its above SLO !

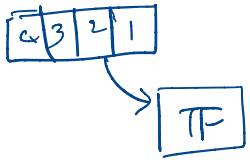
observe latency while lat < SLO inc. batch



Delayed: Wait until batch exists

Why?

Collect examples upto a certain time & then dispatch



(inc lat) Should I wait?

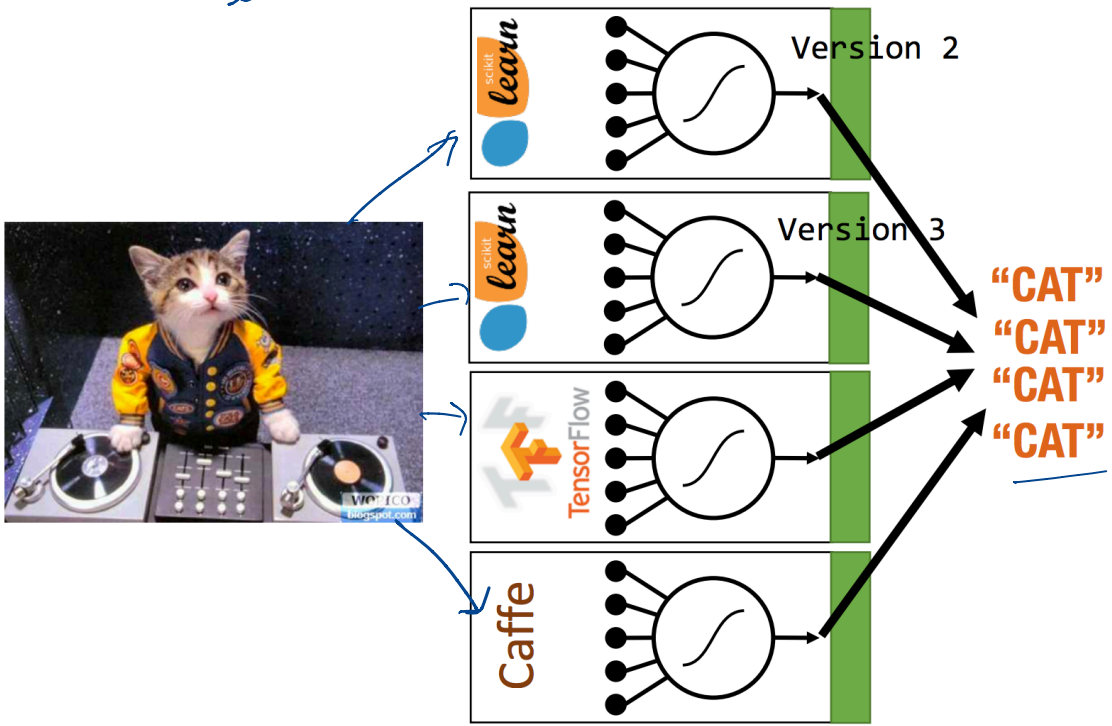
or

(dec util) Should I send?

MODEL SELECTION

→ Improve Accuracy

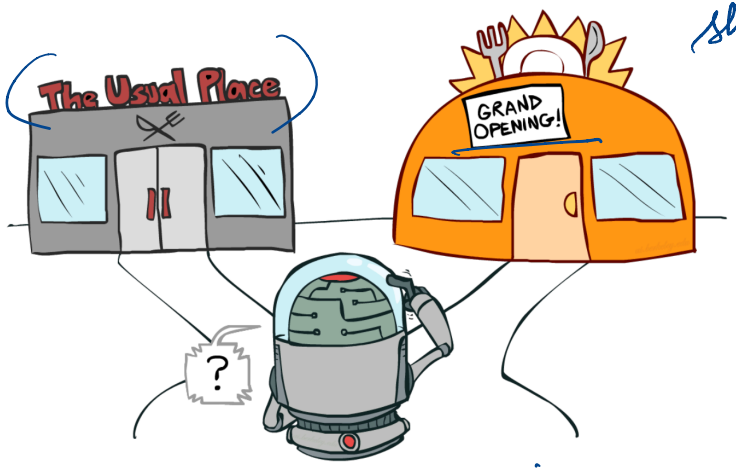
ensembles



SINGLE MODEL SELECTION

Given you have N models
↳ which one should I use?

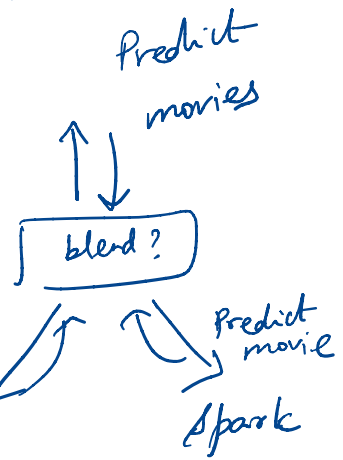
Multi-Arm Bandit formulation
- Explore vs Exploit
- Regret: Loss by not picking optimal action
- Goal: Minimize regret



Clipper
- Exp3 algorithm → associate a weight with each option
- Single evaluation →
- Scales to more models
- update weights based on feedback

model selection
model 1 model 2

MULTI MODELS → ensembles



Ensemble

- Combine output from models (weighted average)
- How do we get the weights ?

Robust Prediction

- React to model changes
- Output confidence score

linear combination

$= \alpha\, y_i + \beta\, z_i$

Binary

classifier →	cat	dog
C1	0.25	0.75
C2	0.4	0.6

Exp4 → update α & β

↳ combine & threshold > 0.5 → cat / dog

STRAGGLER MITIGATION

Why do stragglers occur?

99th pc latency

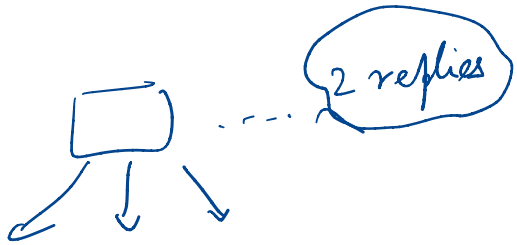
↳ If we wait for N model containers to reply, some of them might be slow?

↳ more replicas / scaling?

Approach

↳ Approx result based on whatever has finished

→ Better approx than late! → ML specific



SUMMARY

- Clipper: ML inference Workloads + Requirements
- Layered architecture provides generality
- Caching, Batching, Replication to improve latency, throughput
- Multi-Arm bandits to improve accuracy

DISCUSSION

https://forms.gle/FCVhPURqz7HSbDtg6

Consider a scenario where you run a model serving service that hosts a number of different applications. The traffic for some applications is sporadic (e.g. only a few hours where they are used). What are some advantages / disadvantages of using Clipper for such a service?

Advantages

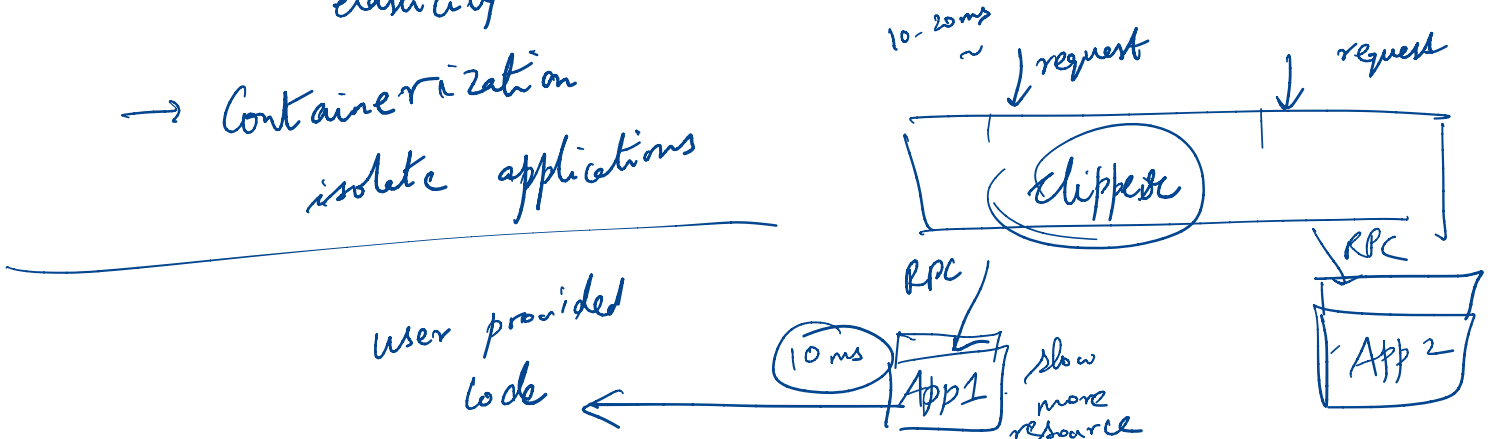
→ Adaptive batching + delayed → tune

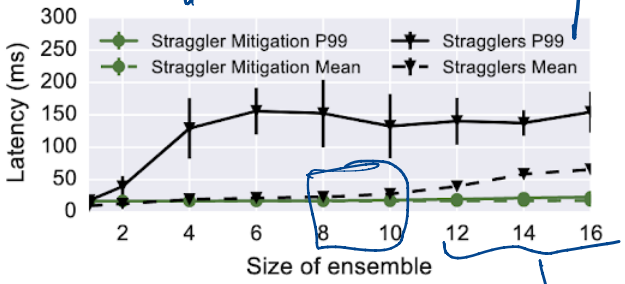
→ multiple replicas → elasticity

→ Containerization → isolate applications

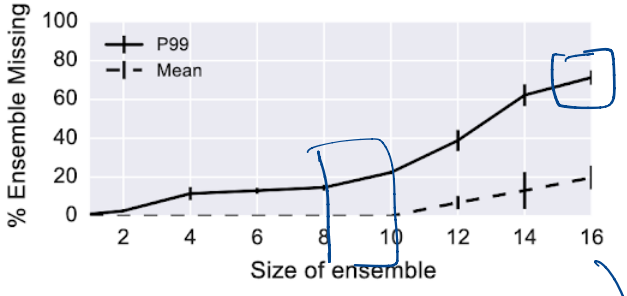
Disadvantages

→ Cache might be contented → how to do this online & fashion?

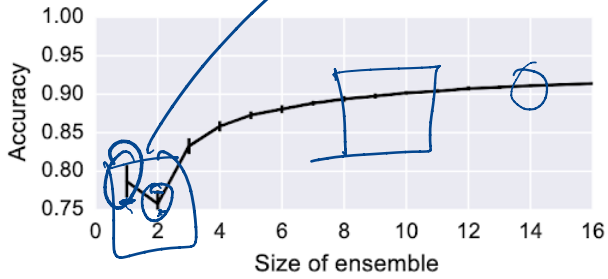




(a) Latency



(b) Missing Predictions



(c) Accuracy

Larger ensembles use a lot more resources

8-10 seems to a good size - balancing bunch of different things?

Whats going on here?

16 ensembles latency inflation is very low

reasonable accuracy