CS 744: NEXUS

Shivaram Venkataraman

Fall 2022

ADMINISTRIVIA

Course Project Proposals

- Due Oct 26! → Check Canvas
- See Piazza for template

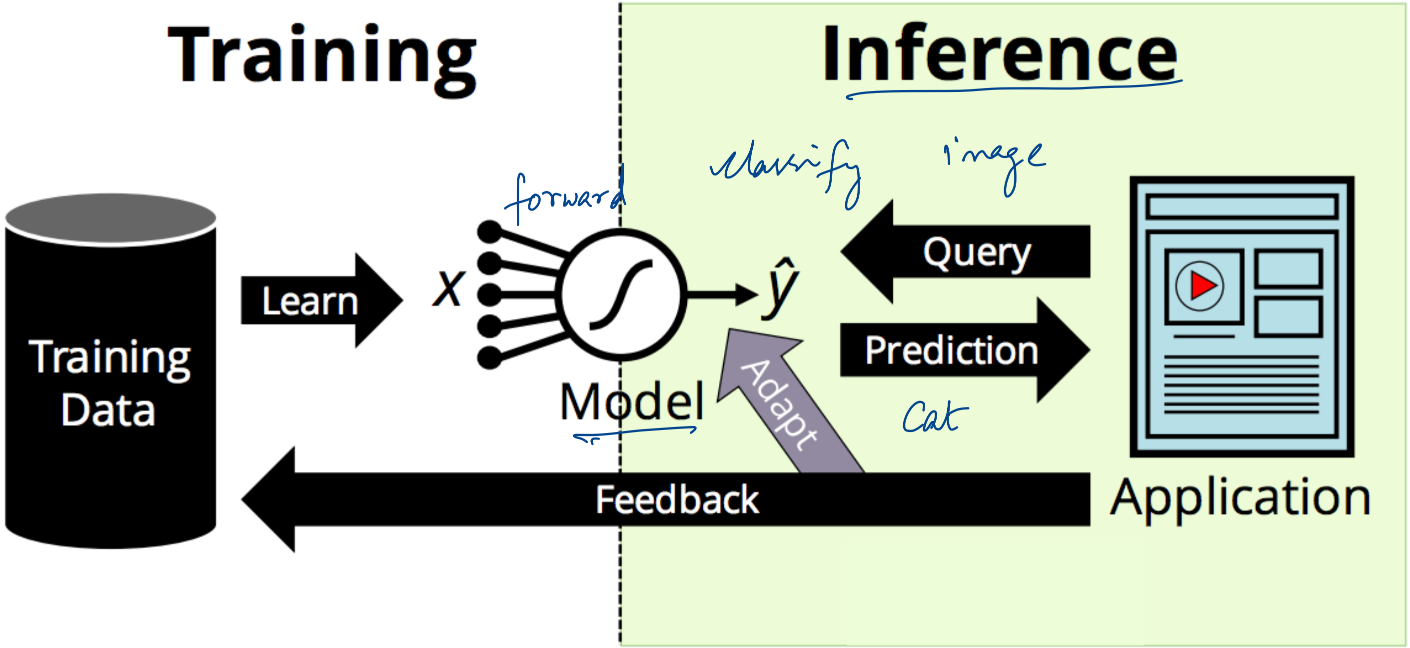
Assignment 1 grades
- Tonight / Tomm.

Midterm details → Prev year Questions on Piazza

- Oct 27th: Includes papers from Datacenter as a Computer to Nexus
- Open book, open notes
- Held in class time ~~9.30-10.45am~~ Central Time

1pm - 2:15pm

MACHINE LEARNING: INFERENCE



EXAMPLE APPLICATION

Video analysis service

- Thousands of streams, thousands of tenants
- Each stream is processed by a DNN-based "query"
- Latency SLOs (10s to 100s of ms)

lots of data
→ arriving continuously

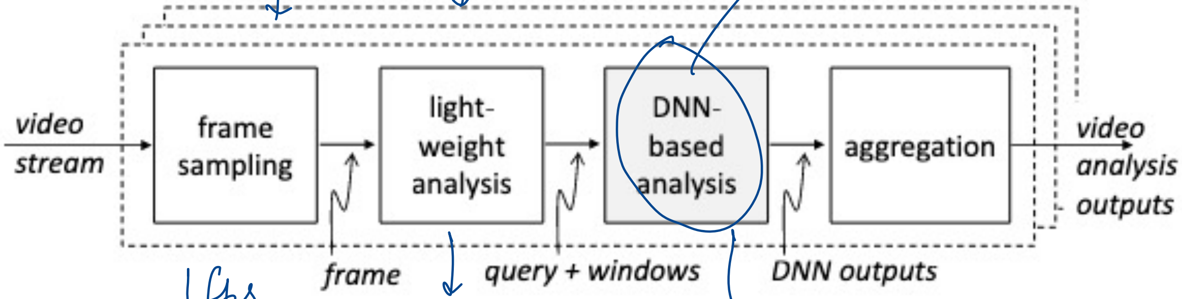
SLO → service-level objectives

↳ Admin / User level decision

input to the system.

Inference pipeline

Pre processing



1fps

throw away empty frames

using GPUs

SCHEDULING GOAL: HIGH GPU UTILIZATION

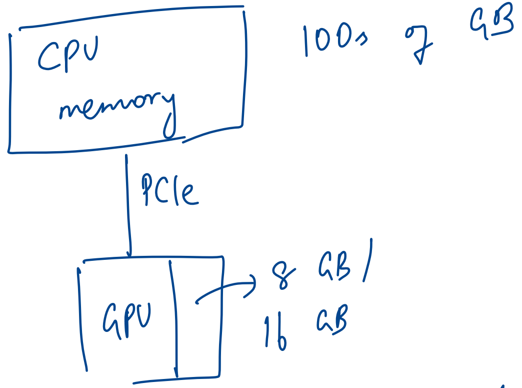
↳ GPUs are expensive

Placement

↳ Which GPUs has which ML model.

↳ Moving ML model to GPU is expensive

↳ GPU memory is limited. Can't store all models.



Batching

$$\text{batch\_lat}(b) = \alpha b + \beta,$$

↳ fixed cost

↳ overhead of launching a GPU kernel

→ memory access etc.

↳ Cost that grows with batch size

Matrix Computations

$\alpha = 1$, $\beta = 5$

$b = 1$ latency $= 5$

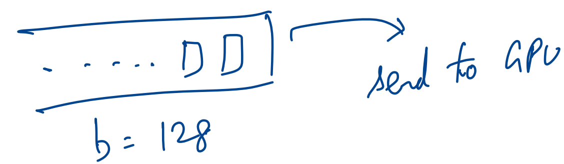
$b = 10$ latency $= 15$

Batch size trade-off

$b = 1$ → low latency

→ Lower utilization

$b = 128$ → High utilization

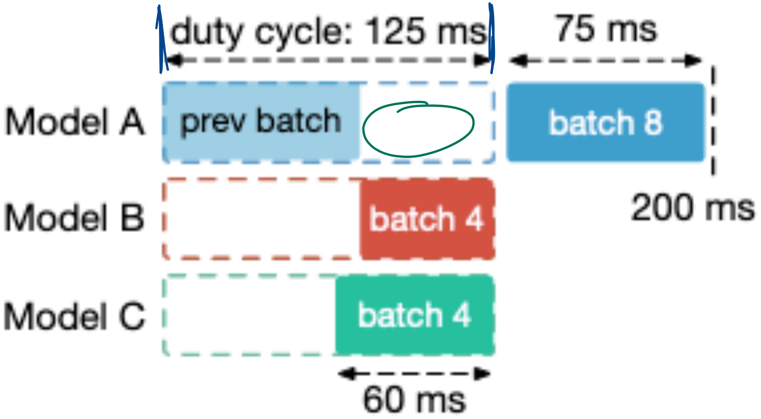


→ Latency SLO miss

↳ to make a batch takes a while

↳ inference of large batch takes longer

SCHEDULING BATCHED EXECUTION



Target tputs $\boxed{\text{A: 64,}}$ B: 32, C: 32 req/sec. SLO: 250ms

Model A			Model B			Model C		
Batch	Lat	Req/s	Batch	Lat	Req/s	Batch	Lat	Req/s
4	50	80	4	50	80	4	60	66.7
8	75	107	8	90	89	8	95	84
16	100	160	16	125	128	16	125	128

None of the models can saturate GPU on their own

When / How can you run multiple models at the same GPU?

Model A & model B colocated

A: b = 8, lat = 75 ms

Model B: b = 4, lat = 50ms → FITS

Model C: b = 4, lat = 60 ms

Does not FIT

BATCH-AWARE SCHEDULING

For each model

All information from prev slide

Inputs: Request rate, SLO for each model, Profiles at batch size

Approach: Allocate "full" GPUs based on load. Handle residuals

Fraction of GPU

Model A req rate = 1100 req/s

Best tput = b = 16, 125 req/s

8 full GPUs = 1000 req/s

Greedy Approximation

Find a packing assignment of k residuals

→ k GPUs to start

→ Merge models to minimize num GPUs

min of two duty cycles

reduce batch size

HANDLING COMPLEX QUERIES

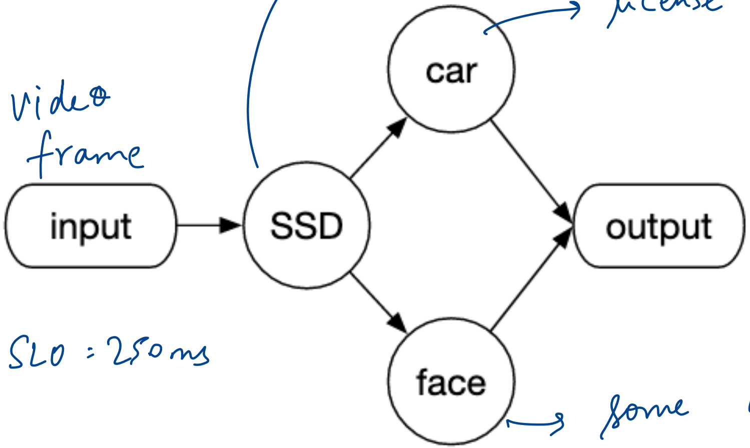
Challenge:

How do we set latency SLOs for complex queries?

type of objects

license plates

video frame



SLO : 250ms

some other model

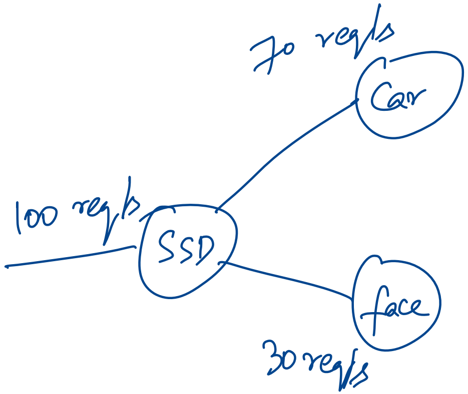
SSD = 100ms

and

car | face = 150ms

Best assignment depends on model properties

SCHEDULING COMPLEX QUERIES



Query Analysis to determine latency SLO splits

Inputs: Models with request rate R_i latency SLO L

$$\underset{\{b_v\}}{\text{minimize}} \quad \sum_{v} R_v l_v(b_v)/b_v$$

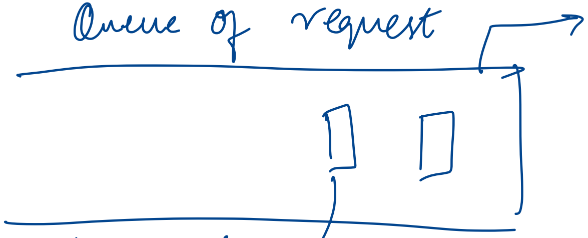
$\rightarrow$ find batch sizes that minimize num GPUs used.

$$\text{subject to} \quad \sum_{u:M_{\text{root}} \rightsquigarrow M_v} l_u(b_u) \leq L \qquad \forall v \in \text{leaf}$$

ADAPTIVE BATCHING

Clipper: Adapt the batch size based on the oldest request in the queue

Queue of request

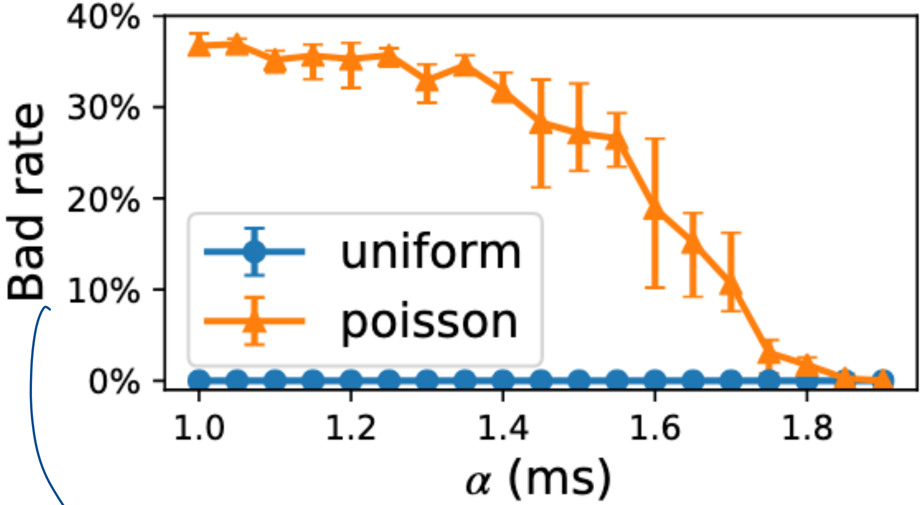


Dropping request that will miss SLO is a good idea

t_arrival + SLO

get time at which this should be executed

$$\text{batch\_lat}(b) = \alpha b + \beta,$$



high fraction of SLO misses

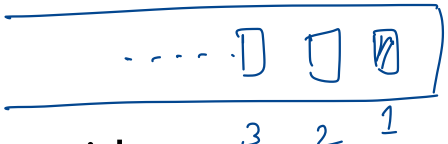
BATCH-AWARE DISPATCH

batch size is given

b = 4

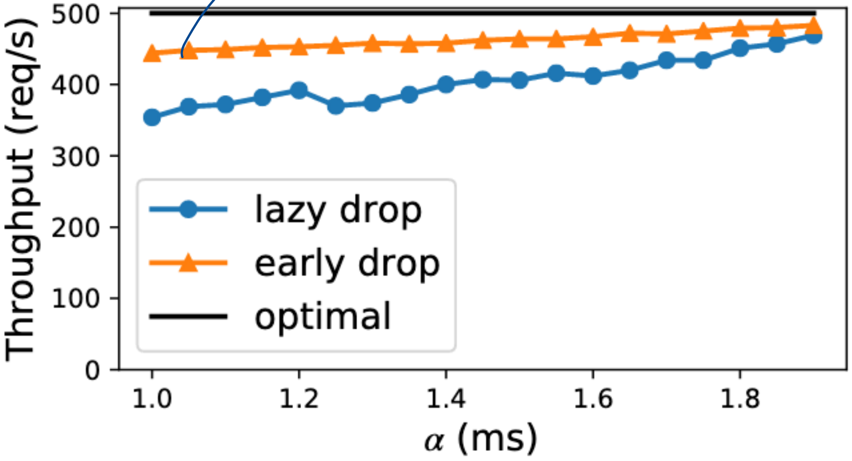
Early-dropping scheme

1. Scans queue using sliding window of batch size



2. Stop at the first request with that can execute *entire window*

reduces num of requests that miss SLO



1 check batch [~~1~~, 2, 3, 4] meet SLO

batch [2, 3, 4, 5] meet SLO

OTHER FEATURES

Prefix Batching

GPU Multiplexing

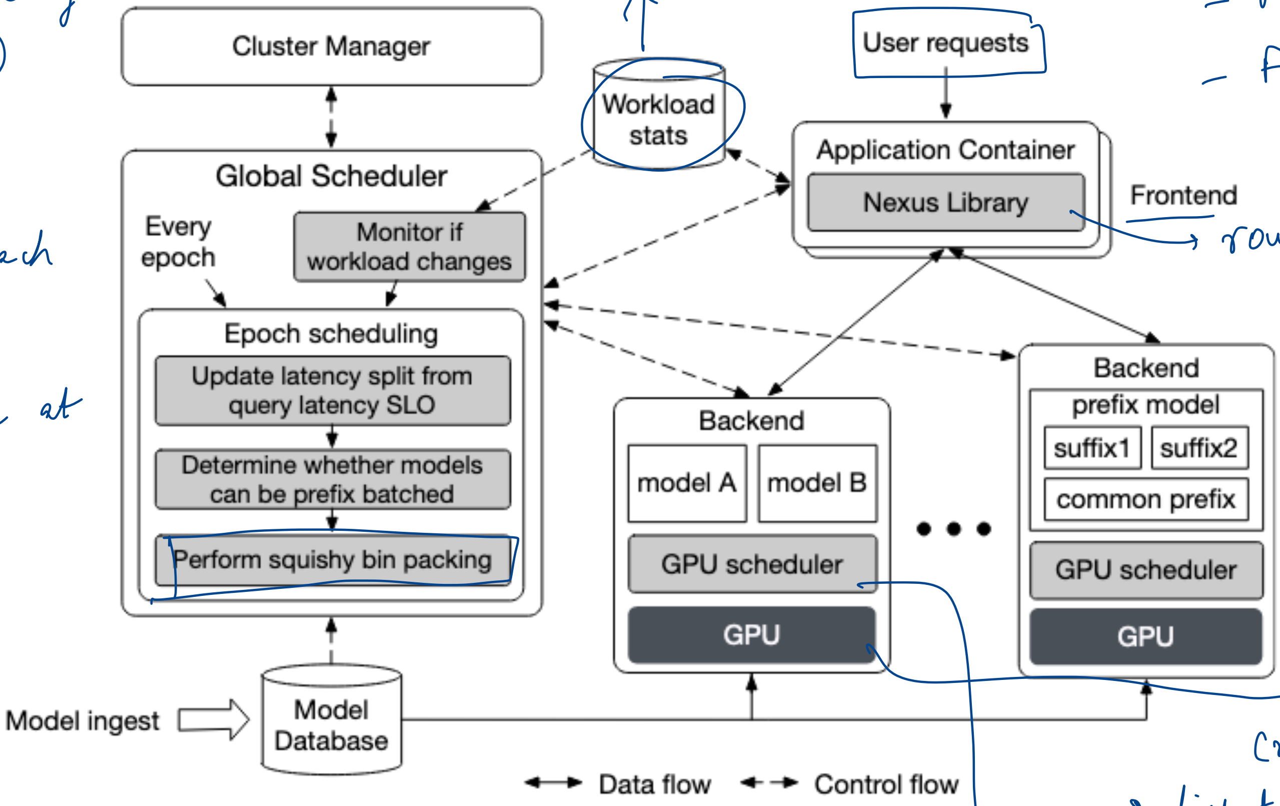
Overlapping CPU and GPU computation

NEXUS ARCHITECTURE

Control Plane

- Run periodically (minute)
- Finds num GPUs for each model
- Batch size at each GPU

throughput



routing table

batches created

dispatched

Data Plane

- Req arrives
- Fwded to backend

SUMMARY

- ML Inference goals: latency SLO, GPU utilization
- Nexus: Handle multiple tenants, multiple DNNs
- Schedule using squishy bin packing
- Breakdown SLO for complex queries, adaptive batching

DISCUSSION

https://forms.gle/PtEaiF4casfZm2JY6

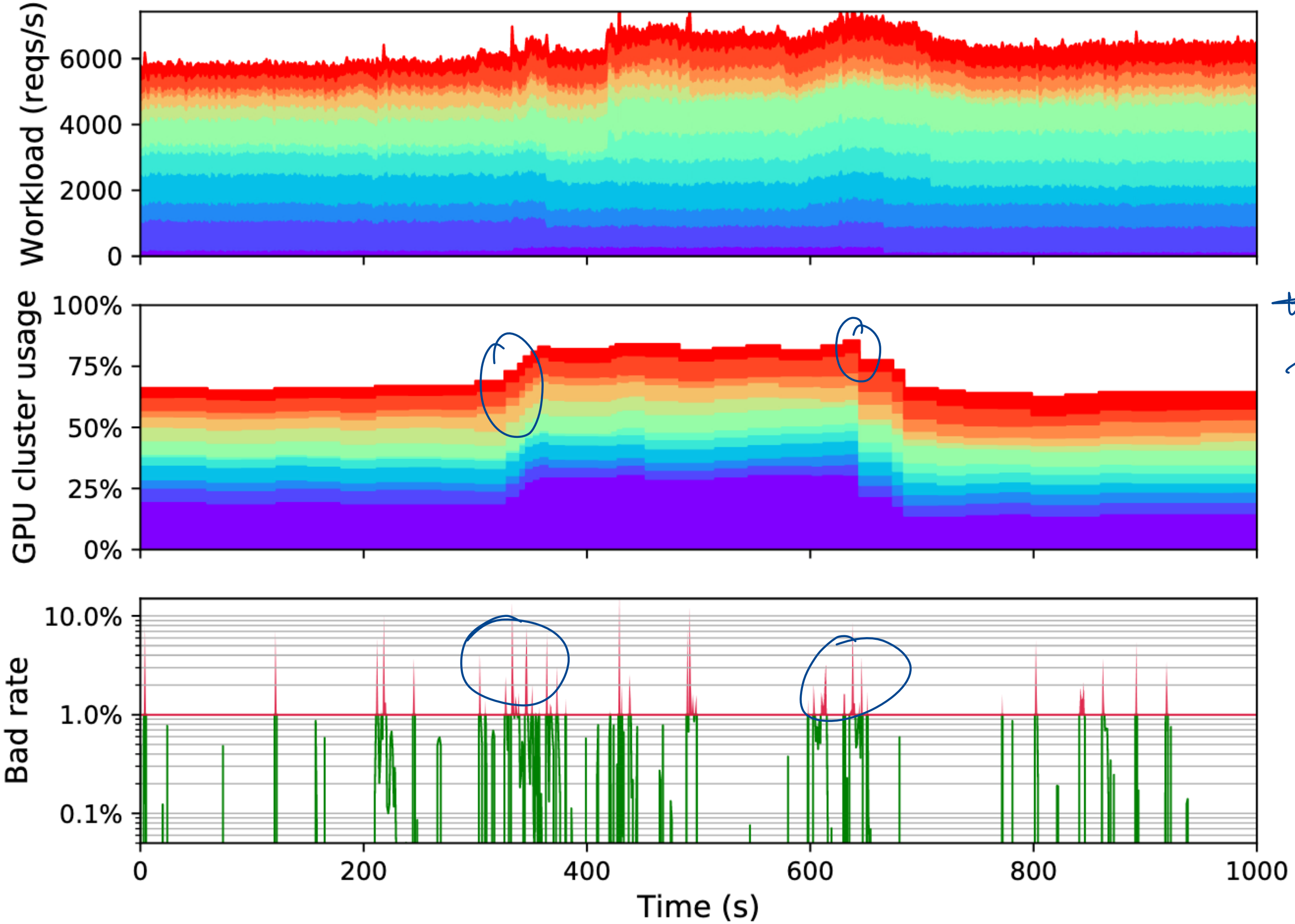
Consider a scenario where you have a model that takes variable amount of time depending on the input. For example if a frame contains 100 cars it takes 250ms to process but if the frame has 1 car then it finishes in 10ms. What could be one shortcoming in using Nexus to schedule this model?

Latency profiling can be hard to do

- Use worst case (~100 cars)
  - lead to underutilization

Profiles

- Dynamic fashion → could pay off if you have patterns
- linear formula? r. number of cars?
  - Pipeline needs a new model to count cars.



~~When~~ At transitions bad rate is high

Next class: SQL

Coming soon

- Project Introductions
- Midterm 1