Hi!
Course project assignments
- Emails will go out today (Oct 14th)
- Introductions due Oct 25th

Midterm Exam
- In class on Oct 28th
- Includes everything from beginning to the end of ML

includes Pollux, Nexus
MACHINE LEARNING: TRAINING

How do I parallelize this?

What is the API?

From one job training to multiple jobs!
WORKLOAD CHARACTERISTICS

- Gang scheduling
- Long running tasks: 150 epochs ~ few minutes → many hours
- All the processes for a job need to run at the same time
- GPU Sharing?
  - Isolation primitives like containers to split cores, memory
Run job Resnet18
With BatchSize = 64
on Num GPUs = 4

Scheduler

Goals:
Maximize throughput
Fairness
Minimize JCT
…
ResNet18 on CIFAR-10

Batch size vs throughput

\[
\text{Time}_{\text{iter}} = T_{\text{grad}} + T_{\text{sync}}
\]

The way to get better throughput with more GPUs:
- Increase batch size!
BATCH SIZE VS STATISTICAL EFFICIENCY

Statistical Efficiency: Amount of progress made per training example

- **GPU parallelism in your compute**
- Each SGD goal to minimize loss
- **Actual statistical efficiency**
- **“Perfect” statistical efficiency**
- Larger batch sizes lead to lower statistical efficiency
- Ideal scaling = double batch size then we take 1/2 num steps
- **Steps to Reach 3.9 Validation Cross Entropy**

Graph showing the relationship between batch size and statistical efficiency.
Scale batch size *during* training based on gradient noise

The optimal batch size can vary during training. GNS is correlated with optimal batch size (for stat. efficiency).

Initial iterations $\rightarrow$ smaller batch size

Towards end of training $\rightarrow$ larger batch size
Maximize goodput across all jobs in the cluster

\[
\text{GOODPUT}_t(\star) = \text{THROUGHPUT}(\star) \times \text{EFFICIENCY}_t(M(\star)),
\]

Approach

1. Job submitted with initial batch size, num GPUs

2. Profile jobs during execution to model throughput, efficiency

3. Tune batch size/GPU and num GPUs based on resource availability
MODELING STATISTICAL EFFICIENCY

For a batch size $M$, initial batch size $M_0$

$$\text{EFFICIENCY}_{t}(M) = \frac{\Phi_t + M_0}{\Phi_t + M}.$$ 

$\Phi_t$: Computed using gradients noise (GNS) using a specific batch size. 

Run the job using $M'$ as batch size. 

Measure $\Phi_t$ at iteration $t$ using $M'$. 

Use the formula above to infer eff. at other batch sizes.
MODELING SYSTEM THROUGHPUT

\[ \text{THROUGHPUT}(a, m, s) = M(a, m, s) / T_{\text{iter}}(a, m, s). \]

\[ T_{\text{grad}}(m) = \alpha_{\text{grad}} + \beta_{\text{grad}} \cdot m, \]

\[ T_{\text{iter}}(a, m, 0) = (T_{\text{grad}}(a, m)^{\gamma} + T_{\text{sync}}(a)^{\gamma})^{1/\gamma}. \]

Account for overlap

\[ \gamma = 1 \rightarrow \text{no overlap} \Rightarrow T_{\text{grad}} + T_{\text{sync}} \]

\[ \gamma = \infty \rightarrow \text{perfect overlap} \Rightarrow \max (T_{\text{grad}}, T_{\text{sync}}) \]

Takeaway: Learn gamma, alpha, beta etc. by running the job
Predict throughputs!
POLLUX: ARCHITECTURE

Talks to agents & determines resources for each job
That manages every job
profiles job
computes $d_1, \beta, \gamma$
GNS
set batch size & LR for this job
Maximize fitness across all jobs

\[ \text{FITNESS}_p(A) = \left( \frac{1}{J} \sum_{j=1}^{J} \text{SPEEDUP}_j(A_j)^p \right)^{1/p} \]

\( p \rightarrow \) fairness knob

- \( p = 1 \) then fitness is just the sum
- \( p = \infty \) minimum speedup \( \Rightarrow \) max-min fairness on speedup
- \( p = -1 \) used in eval.

Approach: Use a search algorithm (genetic algorithm) to find allocation
SUMMARY

DL Workloads Scheduling: Batch Size, Num GPUs
Pollux: Optimal batch size, num GPUs varies
   Across jobs and during a job’s execution
Models for system throughput, statistical efficiency
Cluster-wide optimization
DISCUSSION

https://forms.gle/hQTrk53W3wwkEu9A8
What are some similarities or differences between Mesos and DL schedulers like Pollux?

**Similarity**
- Goals like resource utilization
- Flexibility of workloads
  - **Mesos**: MPI, Spark
  - **Pollux**: only DL
    - Short jobs, long jobs.
- Decomposition
  - Job-level & Centralized

**Diff**
- Mesos only does allocation.
- Pollux allocation +
  - Job config (batch size)
- General API: Mesos
  - Rejecting offers to the job
  - Pollux centralized schedule makes alloc.
At Point (A) demand is low, so Pollux can adapt. But statistical efficiency is not higher.

Pollux is finishing faster, with higher statistical efficiency.
Next Class: Nexus
Course Project Introductions!
Midterm after that