Deep Learning Workloads Scheduling in GPU Clusters

Qingwei Ji
qingweiji1217@gmail.com
University of Electronic Science and Technology of China, Chengdu, P. R. China

April 7, 2023

Outline

- ***Introduction***
- ***Various Schedulers for DL Training Workloads***
- ***Future Work And Discussion***
- ***References***

Introduction

Deep Learning: An important cloud workload

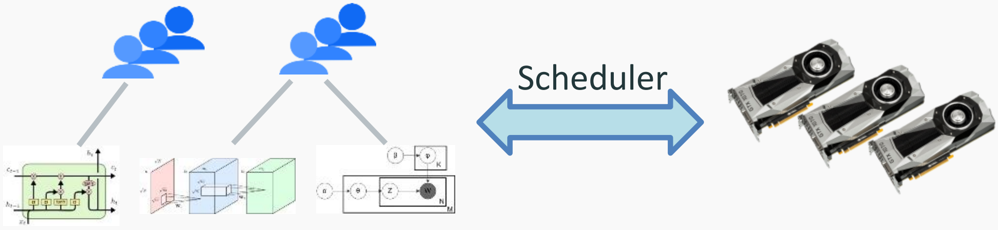
- Deep Learning is ***ubiquitous***: CV, NLP, Recommend...

- DL jobs are ***compute-intensive***, so need expensive hardware
  - Dominant platform today: GPUs
  - Large company runs DL in shared GPU clusters***(billions of $)***

Deep Learning Training (DLT)

- Build a model for an end-to-end application
  - Select best model architecture, invent new architectures, tune accuracy, ...
  - Key to DL Innovation
- DLT is mostly ***trial-and-error***: Little theoretical understanding
  - Will a model architecture work?
    - *Don't know — Train it And Measure!*
  - Lots of trials => high cost:
    - *Training = Significant Fraction of GPU Usage!*

DL Training in GPU Clusters



Many users and DLT jobs

Shared Compute Cluster

Cluster scheduler *decides how to allocate resources to jobs in order to minimize* ***training time****, maximize* ***cluster utilization****, or ensure* ***fairness***

Q: How are DL training jobs scheduled in the existing ML systems?

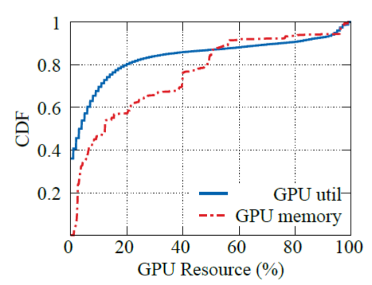
Like Borg(Google), Yarn(Hadoop), Mesos(Apache)...

Cluster Schedulers Today

- Treat DLT jobs as generic big-data jobs(*error*)
  - DLT jobs exhibit certain ***unique features*** distinct from big data jobs[1]
- Expect users to specify the number of resources for each job(*sounds not good*)
  - Rely heavily on the engineering experience of users = > Often leading to ***inefficient resource use***[2]
- Schedule a DLT job on a GPU exclusively, and job holds it until completion (*sounds not good*)
  - ***Static resource allocation*** to jobs may prevent the best training performance[3]

Some Motivations or Problems?

Problem #1: ***Low resource utilization[4]***

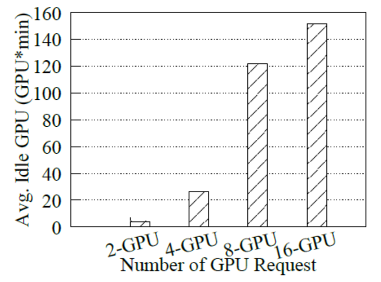


(a)GPU resource statistic on a GPU production cluster[4]

- A DLT job usually can only use parts of a GPU
  - Model training often involves many different steps, such as data preprocessing, etc. Some steps are not suitable for the GPU

Some Motivations or Problems?

Problem #1: ***Low resource utilization[4]***

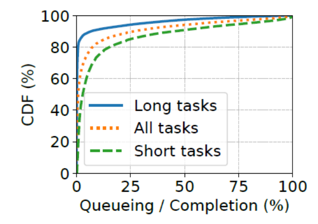


(b)Average GPU idle waiting waste from gang-schedule[4]

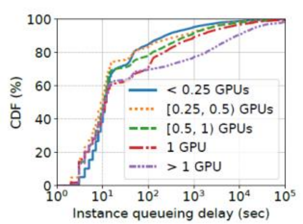
- Idle waiting for gang-schedule
  - ***Gang-schedule***: DL training requires all the GPUs to be allocated simultaneously in an ***all-or-nothing*** manner
  - Multi-GPU training jobs require gang-scheduling
    - *A job will not start training unless all required GPUs are simultaneously available*

Some Motivations or Problems?

Problem #2: ***High Latency(head-of-line Blocking)[5]***



(a)CDF of normalized instance queueing delays[5]



(b)CDF of queueing delays w.r.t. GPU requests per instance[5]

- Long queueing delays for short-running jobs
  - Long DLT job Runtime: Several days!
  - GPU sharing

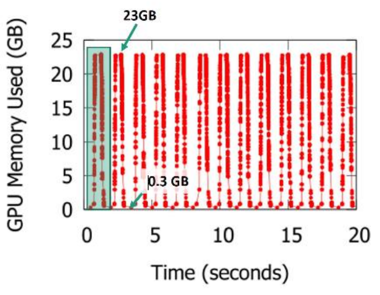
Don't worry,

let's break it down!

DLT Workloads' Unique Characteristics

A series of studies have characterized training workloads from the production GPU datacenters, including Alibaba [5], Microsoft [6] and SenseTime [7]. The characteristics are summarized as below.

- Domain knowledge: ***Intra-job predictability[6,8]***



ResNet50 training on ImageNet data

- Each job performs repetitive iterations with constant behaviors and duration
- To predict future GPU memory usage and job completion time

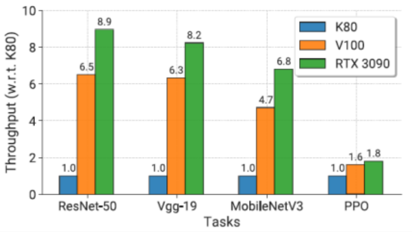
DLT Workloads' Unique Characteristics

- Domain knowledge: ***Feedback-driven exploration[8]***
  - Training a DL model is a typical ***trial-and-error*** process
  - Early-feedback on DLT jobs is critical, especially in the initial stages of training
    - *Select the model structure*
    - *specify the hyper-parameters, including:*
      - *the number of layers/weights in the model*
      - *minibatch size*
      - *learning rate*
      - …

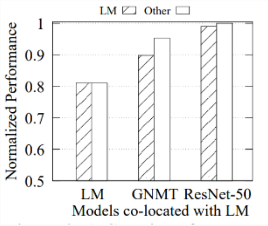
These are typically chosen by the user based on domain knowledge and trial-and-error, and can sometimes even result in early training failure.

DLT Workloads' Unique Characteristics

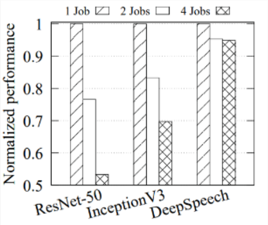
- ***Inherent Heterogeneity[1](sounds not accurate) + Interference Sensitivity[8](✓)***



(a)Heterogeneous Affinity[1]



(b)1-GPU interference[8]

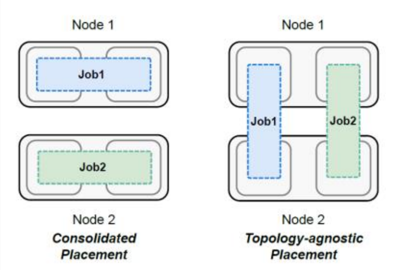


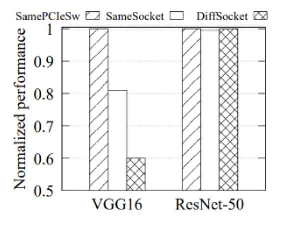
(c)NIC interference[8]

- When running in a shared execution environment, DLT jobs might interfere with each other due to resource contention
- Jobs widely differ in terms of memory usage, GPU core utilization, sensitivity to interconnect bandwidth, and/or interference from other jobs

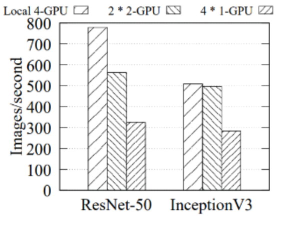
DLT Workloads' Unique Characteristics

- ***Placement Sensitivity[1,8]***





(b)Intra-server locality



(c)Inter-server locality

(a)Job Placement[1]

- The runtime speed of some distributed DL jobs are bounded by device-to-device communication
- The communication sensitivity of training jobs depends on the inherent property of the model structure

Review the Cluster Schedulers Today

- Treat DLT jobs as generic big-data jobs(*error*)
  - ***DLT jobs exhibit certain unique features distinct from big data jobs[1] => get(✓)***

Causing some scheduling challenges?

Making cluster scheduling of DLT jobs inefficient?

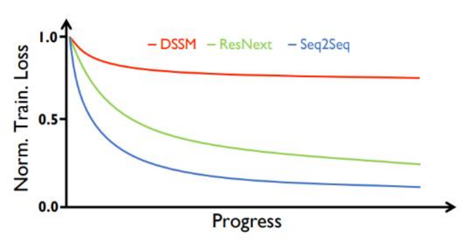
Expect users to specify the number of resources for each job(*sounds not good*)

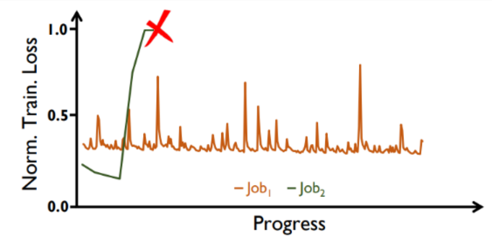
Schedule a DLT job on a GPU exclusively, and job holds it until completion (*sounds not good*)

==> Maybe it's a process of mutual influence

Challenges for Scheduling DL Training Jobs

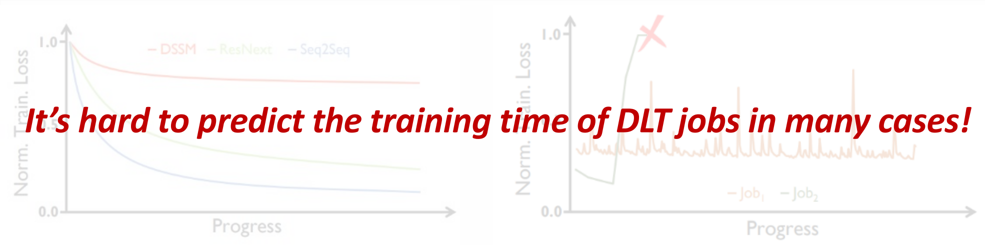
- ***Resource Scheduling:* Unpredictable Training Time[9]**
  - Unknown execution time of DL training jobs
    - *Job execution time is useful when minimizing JCT(Job Completion Time)*
  - Predict job execution time
    - *Use the smooth loss curve of DL training jobs(Optimus[10])*





Challenges for Scheduling DL Training Jobs

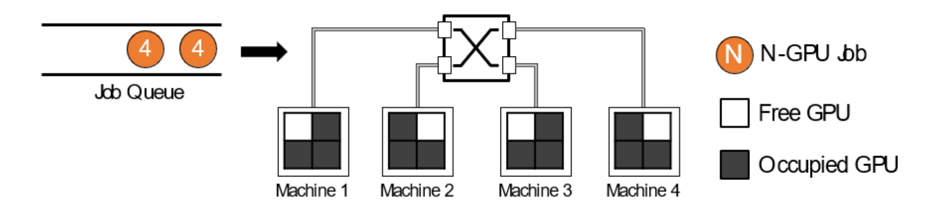
- ***Resource Scheduling:* Unpredictable Training Time[9]**
  - Unknown execution time of DL training jobs
    - *Job execution time is useful when minimizing JCT*
  - Predict job execution time
    - *Use the smooth loss curve of DL training jobs(Optimus[10])*



It's hard to predict the training time of DLT jobs in many cases!

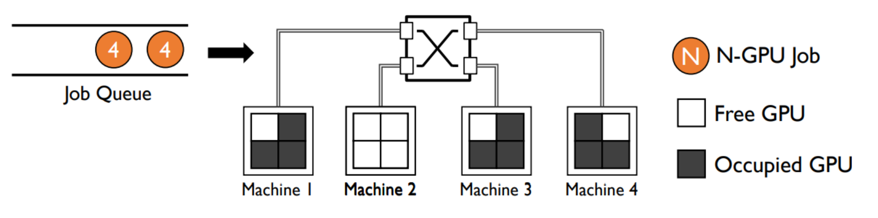
Challenges for Scheduling DL Training Jobs

- ***Job Placement:*** ***Over-Aggressive Job Consolidation[9]***
  - Network overhead in DL training



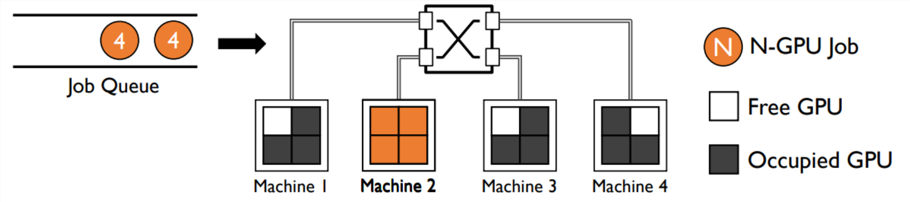
Challenges for Scheduling DL Training Jobs

- ***Job Placement:* *Over-Aggressive Job Consolidation[9]***
  - Network overhead in DL training
  - ***Consolidated placement*** for good training performance



Challenges for Scheduling DL Training Jobs

- ***Job Placement:* Over-Aggressive Job Consolidation[9]**
  - Network overhead in DL training
  - ***Consolidated placement*** for good training performance
    - *Fragmented free GPUs in the cluster*
    - *Longer queuing delay*



Well,

everything seems to fit together!

~~Review the Cluster Schedulers Today~~

- Treat DLT jobs as generic big-data jobs(*error*)
  - *DLT jobs exhibit certain unique features distinct from big data jobs[1] => get(✓)*

Causing some scheduling challenges?

Making cluster scheduling of DLT jobs inefficient?

Customize: An Effective and Efficient GPU Cluster Scheduler for Distributed Deep Learning Training Jobs

Expect users to specify the number of resources for each job(*sounds not good*)

Schedule a DLT job on a GPU exclusively, and job holds it until completion (*sounds not good*)

==> Maybe it's a process of mutual influence

Existing Work

Different Scheduling Objectives

Different Scheduling Objectives

- Reduce the average queuing and ***execution time*** of training jobs. ★★★
- Reduce power ***consumption***. ★
- Guarantee the ***fairness*** among different entities (user-level, job-level). ★
- Maximize the ***utilization*** of resource. ★
- Ensure the job can be done before the ***specified deadline***.★
- ...

Reduce JCT(Job Completion Time)

- ***Optimus[10] (Elastic scheduling)***
  - Approach: Performance Modelling
  - Advantages: JCT Reduction
- ***Tiresias[9]***
  - Approach: Gittins index; Least-Attained Service (LAS)
  - Advantages: Information-agnostic
- ***Aonline[11] (Elastic scheduling)***
  - Approach: Integer Linear Programming
  - Advantages: JCT Reduction

Other Objectives

- ***ANDREAS[12] (reduce energy consumption)***
  - Approach: Randomized Greedy Algorithm
  - Advantages: Energy Cost Reduction
- ***Themis[13] (guarantee fairness)***
  - Approaches: Finish-Time Fairness; Auction Bid
  - Advantages: Better Fairness
- ***Gandiva[8](maximize the utilization of resource)(Elastic scheduling)***
  - Approaches: Time-slicing; Migration; Grow-shrink
  - Advantages: Better GPU Utilization
- ***Chronus[14](guarantee ddl)***
  - Approach: Linear Programming; Local Search Allocation
  - Advantages: SLO Guarantee

Future Work And Discussion

Some Ideas? => Maybe they are enabled

- ***Objectives***
  - *SLO guarantee*
  - *Energy consumption optimization*
- ***Considerations***
  - *Resource heterogeneity*
  - *Predicted job's information inaccuracy*
- ***Methods***
  - *GPU sharing*
  - *Elastic scheduling*

References

[1] Gao W, Hu Q, Ye Z, et al. Deep Learning Workload Scheduling in GPU Datacenters: Taxonomy, Challenges and Vision[J]. arXiv preprint arXiv:2205.11913, 2022.

[2] Qiao A, Choe S K, Subramanya S J, et al. Pollux: Co-adaptive Cluster Scheduling for Goodput-Optimized Deep Learning[C]//OSDI. 2021, 21: 1-18.

[3] Bao Y, Peng Y, Wu C, et al. Online job scheduling in distributed machine learning clusters[C]//IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2018: 495-503.

[4] Xiao W, Ren S, Li Y, et al. AntMan: Dynamic Scaling on GPU Clusters for Deep Learning[C]//OSDI. 2020: 533-548.

[5] Weng Q, Xiao W, Yu Y, et al. MLaaS in the wild: Workload analysis and scheduling in Large-Scale heterogeneous GPU clusters[C]//19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22). USENIX Association, 2022: 945-960.

[6] Jeon M, Venkataraman S, Phanishayee A, et al. Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads[C]//USENIX Annual Technical Conference. 2019: 947-960.

[7] Hu Q, Sun P, Yan S, et al. Characterization and prediction of deep learning workloads in large-scale gpu datacenters[C]//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2021: 1-15.

[8] Xiao W, Bhardwaj R, Ramjee R, et al. Gandiva: Introspective cluster scheduling for deep learning[C]//13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18). 2018: 595-610.

[9] Gu J, Chowdhury M, Shin K G, et al. Tiresias: A GPU Cluster Manager for Distributed Deep Learning[C]//NSDI. 2019, 19: 485-500.

[10] Peng Y, Bao Y, Chen Y, et al. Optimus: an efficient dynamic resource scheduler for deep learning clusters[C]//Proceedings of the Thirteenth EuroSys Conference. 2018: 1-14.

[11] Zhou R, Pang J, Zhang Q, et al. Online scheduling algorithm for heterogeneous distributed machine learning jobs[J]. IEEE Transactions on Cloud Computing, 2022.

[12] Filippini F, Ardagna D, Lattuada M, et al. ANDREAS: Artificial intelligence traiNing scheDuler foR accElerAted resource clusterS[C]//2021 8th International Conference on Future Internet of Things and Cloud (FiCloud). IEEE, 2021: 388-393.

[13] Mahajan K, Balasubramanian A, Singhvi A, et al. Themis: Fair and efficient GPU cluster scheduling[C]//17th USENIX Symposium on Networked Systems Design and Implementation. 2020.

[14] Gao W, Ye Z, Sun P, et al. Chronus: A novel deadline-aware scheduler for deep learning training jobs[C]//Proceedings of the ACM Symposium on Cloud Computing. 2021: 609-623.