Analysis of GPU Cluster Scheduling Frameworks for DDL Training Workloads

Anurag Gupta
New York University
ag7662@nyu.edu

Arnab Arup Gupta
New York University
ag7654@nyu.edu

Jaya Mundra
New York University
jm8834@nyu.edu

Aurojit Panda
New York University
apanda@cs.nyu.edu

***Abstract*—This paper discusses the evaluation of the performance of GPU Cluster schedulers from two research paper - Gandiva and Tiresias - that work with Distributed Deep Learning training workloads and analyzed them on the Job Completion Time and GPU Utilization since these are important metrics to job owners and cluster managers. We have presented our results with appropriate reasoning and compiled a comparative study.**

I. Introduction

Distributed Deep Learning (DDL) is a mechanism used to carry out training in parallel using multiple GPUs because this training is characteristically compute-intensive and data sets used for training are increasing in size by the day. GPU clusters, provided by platform providers, are shared by users who can leverage these large collections of GPUs for their DDL jobs.

There are two factors behind proper allocation and optimal usage of these GPUs which help bring down the Job Completion Time (JCT) cluster-wide and boost the GPU Utilization of the cluster. While the first of these consequences concerns the job, the second one concerns the cluster manager as it plays a role in optimising costs. Since a GPU VM costs about ten times as much as a regular VM in the cloud, the efficient utilization of cluster becomes that much more important.

Deep Learning Training Jobs are characterised by (1) Sensitivity to locality, (2) sensitivity to inference, and (3) intra-job predictability, which can be exploited to increase GPU utilization while making sure not to affect average JCT. The two key tasks that most research in the area focuses on are efficient job scheduling and smart GPU allocation. In this report, we shall be discussing two algorithms - Gandiva and Tiresias - developed 3 years apart, that attempt at resolving issues faced by traditional cluster schedulers. They are explained in brief in the following section.

We have carried out a systematic analysis of these algorithms and evaluated them based on average JCT and cluster-wide GPU utilization. In the following sections, we present our approach, our results along with appropriate supporting justification and our conclusions and learning from conducting these experiments.

II. Related Work

In this section, we present short overviews for Gandiva [3] and Tiresias [2]. They shall be further discussed in Section III.

Gandiva focuses on increasing the cluster efficiency while also improving latency for jobs. It exploits intra-job predictability for time slicing GPUs efficiently across multiple jobs to ensure low latency. Gandiva is information agnostic and uses feedback-driven exploration to prioritize jobs. Additionally, it leverages migration of jobs, suspend-resume mechanisms, grow-shrink algorithms and efficient packing to improve cluster efficiency and reduce fragmentation.

The authors of Tiresias propose two scheduling algorithms - Discretized Two-Dimensional Gittins index which relies on partial information and Discretized Two-Dimensional LAS which is information-agnostic. The algorithms aims to minimize the average JCT by employing priority discretization and loosening the consolidated placement constraint by leveraging a simple, externally-observable, model specific criteria.

Other schedulers face issues dealing with such workloads due to (1) the unavailability of a job's remaining execution time, (2) over-simplified assumptions that include predictions about loss curves, and (3) over-aggressive consolidation during placement. Sometimes, schedulers event treat DDL jobs as big-data jobs that are allocated GPUs once and hold onto them exclusively till they complete their execution.

III. Algorithms

A. Gandiva

Gandiva was implemented to exploit features characteristic of Deep Learning Training (DLT) jobs: intra-job predictability to implement application aware time slicing, to perform profile driven introspection. Gandiva comprises of a handful of mechanisms to achieve high GPU utilization. Therefore, they focus on providing early feedback to jobs and maximising cluster efficiency.

1) Suspend Resume: This is used to remove exclusivity of a set of GPUs to a DLT job and is carried out when GPU memory usage is at its lowest.
2) Packing: Alternatively, multiple DLT jobs can be run on a GPU simultaneously, letting the GPU time-share jobs.
3) Migration: This changes the set of GPUs assigned to a DLT job. This helps to move interfering jobs away from each other and, more importantly, helps reduce fragmentation of the cluster.
4) Grow Shrink: Grow the number of GPUs available to a job opportunistically during idle times and correspond-

ingly also shrink the number of GPUs available when load increases.

5) Profiling: Gandiva introspects DLT jobs in an application aware manner to estimate rate of progress and determines whether packing is effective to make packing decisions.

B. Tiresias

Tiresias prioritizes reducing the average JCT and works with the fact that a DL job's execution time is often unpredictable. It also determines when the consolidated placement constraint can be relaxed. It utilizes priority discretization to avoid aggressive job preemption. The Gittins index is a representation of the ratio of the probability that the job will complete within an additional service quantum to the expected service required for completion.

1) 2D Scheduling: Tiresias reviews both the time and size-based heuristics i.e. the temporal and spatial requirement, i.e. required duration and GPU count in prioritizing.
2) LAS: Least Attained Service assigns each job a priority based on its attained service, which is a product of the number of GPUs it uses and the amount of time it has been running so far.
3) Profiling: The amount of skew in tensor distributions are identified to consolidate the job accordingly.
4) Avoids Starvation by allowing priority queue jumps.

IV. Experiments

A. Job Traces

We run our experiments on a trace obtained from Deep Neural Network (DNN) training workloads on Microsoft's internal Philly clusters [1]. The job trace log contains information about each job, including each individual successful scheduling attempt. The trace spans over 4 months including information of about 117325 jobs. While all successful scheduling attempts were listed, we filtered out those which ended in either a FAILURE or was KILLED (where a FAILURE is due to node or system crashes, and KILLS are called manually by users), and selected those attempts which resulted in a completion of the job. The reason behind this filtering was our selection of metrics (JCT and GPU Utilization) which require the execution times and log of GPUs used at each time step and including these unnecessary attempts would only blur the true trends of our graphs for both metrics.

We subsampled the trace to generate two different traces and performed our experiments on both traces. The CDF graph for both the traces are shown in Figure 1 and Figure 2. The original trace does not provide information about the models but Gandiva uses models for its placement algorithms. Therefore we profile and analyze model-wise GPU requirements from the job trace provided by the authors of Tiresias to understand model distribution. We accordingly assign the following models - vgg19, vgg16, vgg11, alexnet, resnet101, resnet50, inception4, inception3, alexnet152 - based on a job's GPU requirements.

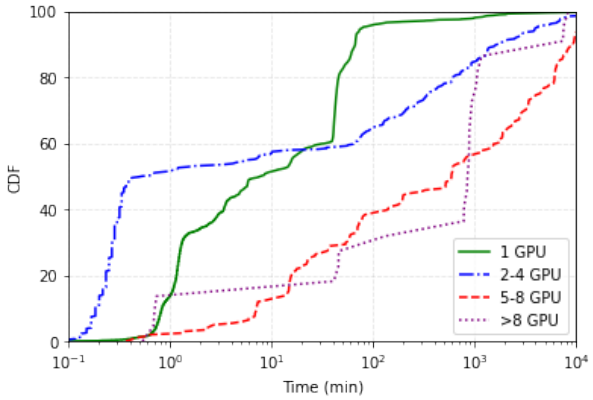


Fig. 1. Job Trace 1

The first trace is sampled by taking the first 10000 successful jobs spanning over 2 months with the same GPU distribution as the original trace. The generated trace accounts for 90% of jobs requiring a single GPU.

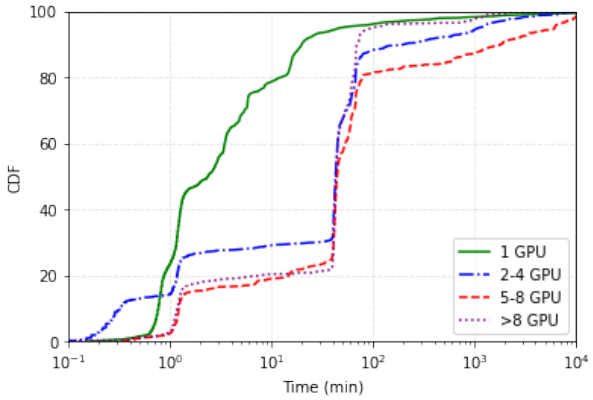


Fig. 2. Job Trace 2

The second trace is generated to understand GPU utilization by decreasing one-GPU requirement jobs to 50% and the rest of the 50% jobs requiring 2 to 64 GPUs. The job arrival pattern for this trace was kept similar to the first 10000 jobs of the original trace.

B. Implementation

The simulator and algorithms were implemented in Python2.7 and executed on the NYU Snappy server with Two Intel Xeon E5-2680 (2.80 GHz) (20 cores) CPUs with the operating system as CentOS 7 and 128GB memory.

Along with 2D-LAS, 2D-Gittins and Gandiva, we also implement three naive algorithms - FIFO (First In First Out), SJF (Smallest Job First) in terms of GPU requirements and SRTF (Shortest Remaining Time First).

For Gandiva, we do not implement grow-shrink strategy which increases/decreases allocated GPUs to a job during idle/busy times. The information needed to device grow-shrink strategy was not available in Philly traces. Early feedback

and profiling was also not possible to account for in the simulations. Instead, we profiled the jobs ourselves manually and the model used these labels in its placement algorithm.

We assume non-sharable GPUs and hence, left out the packing strategy discussed in the Gandiva in our implementation. Packing is the mechanism where more than one job is assigned to a single GPU and shared the resource to optimize GPU utilization.

For our simulations, we could not gauge preemption overheads, network communication delays, or latency that arises due to hardware-related constraints.

V. Results

This section talks about the observations from the simulations carried out on both the traces.

A. Job Completion Time

For trace 1, we see that SJF and SRTF gives the best results because they utilize the most information to schedule the job. For DDL jobs, this information is generally not available beforehand and hence, SJF and SRTF cannot be applied to use in practical settings. SRTF can be assumed as the best case in our scheduling.

TABLE I
Job Completion Time For Trace 1

Algorithm	Average JCT (s)	FOI*
2D-LAS	3952.88	1.0
2D-Gittins	3867.927	0.979
Gandiva	12146.911	3.073
FIFO	4822.813	1.22
SJF	3936.838	0.996
SRTF	3857.419	0.976

It is observed that 2D-Gittins performs almost as well as SRTF and it is justified because the calculation of Gittins index requires information about job distribution and is not completely information agnostic, so the algorithm has an expectation about the job's execution time.

However, we see very high average JCT for Gandiva. Gandiva functions in a round robin pattern and as seen Figure 3.b, though most of the jobs complete very early on in the time frame, there are few straggler jobs which take a large amount of time to complete increasing the overall average JCT.

TABLE II
Job Completion Time For Trace 2

Algorithm	Average JCT (s)	FOI
2D-LAS	21957.588	1.0
2D Gittins	19084.053	0.869
Gandiva	40124.706	1.827
FIFO	44924.197	2.046
SJF	23002.992	1.048
SRTF	17284.951	0.787

For Trace 2 (refer to Table II), trends similar to Trace 1 are witnessed in the sense that all these algorithms perform just as well when ranked amongst themselves as they did for Trace 1. On a whole, the average JCT is much higher for all of them which can be attributed to the increase in GPU requirement of Trace 2 over Trace 1, while the arrival times stay the same. This is because lesser jobs can be accommodated at the same time, causing more jobs to have to wait longer. However, the FOIs are more drastic. Since we have a more uniform distribution of GPU requirements, the profiling benefits of Gandiva kick in and it outperforms FIFO. The performance of SJF degrades due to waiting time increase for large jobs. A point to note is that all algorithms that use some information perform better because of the increase in variance of the GPU sizes in Trace 2.

*Factor of Improvement

B. GPU Utilization

For our purposes, we have limited the analysis of GPU utilization to only the three algorithms (from Gandiva and Tiresias) we focused on in the paper.

For trace 1 (Figure 1.c), high utilization is observed for all the algorithms. Average GPU Utilizations are composed in Table III. Gandiva shows low average GPU utilization as compared with 2D-LAS and 2D-Gittins because as shown in the Figure 3.b, few straggler jobs are running for a long time during which majority GPUs are idle and decreasing the cluster efficiency towards the tail. As against this, the two Tiresias algorithms complete sooner and therefore maintain their high average GPU utilizations.

TABLE III
GPU Utilization for Trace1

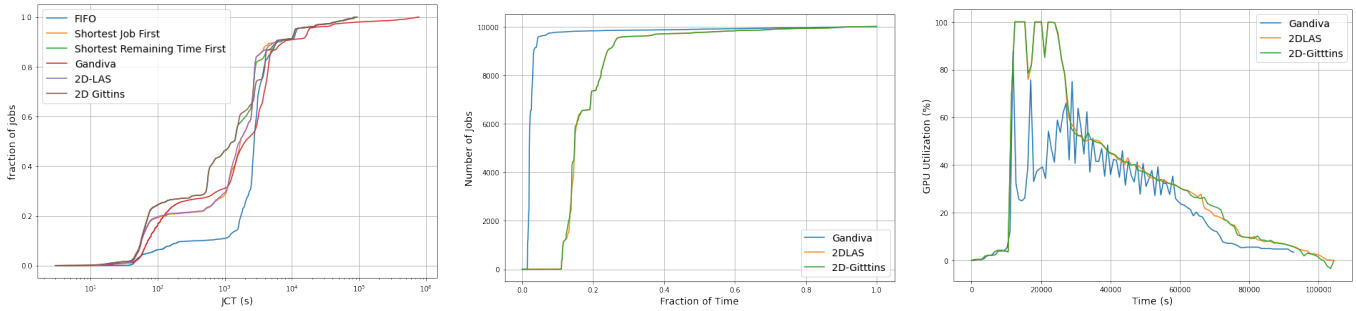
Algorithm	GPU Utilization
2D-LAS	88.368
2D-Gittins	87.892
Gandiva	24.114

It is noticed that the utilization for Gandiva is lower compared to 2D-LAS and 2D-Gittins during the peak job execution time even though Gandiva is optimized for high GPU utilization. We believe the shown trend is because of the memory constraints due to which even though nodes are assigned to job but are unable to run leading to lower GPU usage. For trace 1, 90% of jobs require single GPU and more jobs are placed without considering their affinity because nodes with single GPUs gets exhausted quicker leading to fragmentation of the cluster.

TABLE IV
GPU Utilization for Trace2

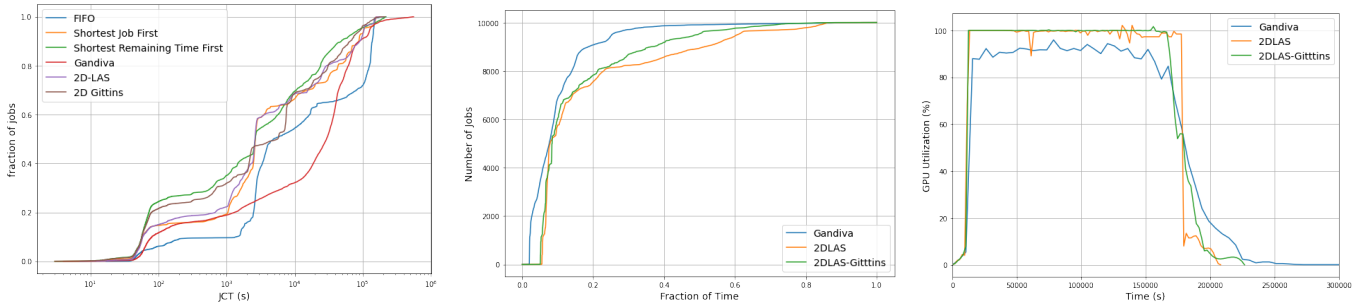
Algorithm	GPU Utilization
2D-LAS	97.51
2D-Gittins	97.309
Gandiva	56.654

For trace 2, there has been an increase in GPU utilization for all the algorithms and significantly for Gandiva. Average GPU utilizations are composed in Table IV. Trace 2 has job distribution with equal number of jobs with high GPU



(a) Job Completion Time (b) Job Completion Rate (c) GPU Utilization

Fig. 3. Metrics for Trace 1



(a) Job Completion Time (b) Job Completion Rate (c) GPU Utilization

Fig. 4. Metrics for Trace 2

requirement. Therefore, the scheduler is able to assign jobs according to their node affinity and reduces fragmentation of the cluster. This leads to more nodes being used on average instead of being idle. Although Gandiva still takes longer to finish, and has lower average GPU Utilization due to the low-valued long tail, this tail is shorter than in Trace 1 and we have a better average GPU utilization for this trace.

VI. Conclusion

From the experiments on both the traces, it is observed that 2D-LAS and 2D-Gittins performs better in terms of average JCT compared to Gandiva. The discrete priortization and LAS algorithm helps achieve faster completion time for jobs. Moreover, as the Gittins index is introduced, it accounts for some information content and the scheduler takes advantage of this information to outperform 2D-LAS.

The GPU utilization is also high for 2D-LAS and 2D-Gittins and hence they achieve good cluster efficiency. The GPU utilization for Gandiva is unexpectedly lower given Gandiva focuses on increasing GPU utilization and maximizing cluster efficiency. This observation could be due to varied factors including the distribution of job requirements and negligence of hardware optimization in the simulations.

Job Completion Time and GPU Utilization are good metrics to compare schedulers with since they look at the temporal as well as spatial factors of a GPU cluster. We have learnt that more the information, better the scheduler as there is higher potential optimization (part-hardware-oriented) to leverage.

References

[1] Jeon, Myeongjae and Venkataraman, Shivaram and Phanishayee, Amar and Qian, Junjie and Xiao, Wencong and Yang, Fan, “Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads” In 2019 USENIX Annual Technical Conference (USENIX ATC 19), pp. 947–960.

[2] Gu, J., Chowdhury, M., Shin, K. G., Zhu, Y., Jeon, M., Qian, J., ... & Guo, C. (2019). Tiresias: A GPU cluster manager for distributed deep learning. In 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19) (pp. 485-500).

[3] Xiao, Wencong, Romil Bhardwaj, Ramachandran Ramjee, Muthian Sivathanu, Nipun Kwatra, Zhenhua Han, Pratyush Patel et al. ”Gandiva: Introspective cluster scheduling for deep learning.” In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), pp. 595-610. 2018.