2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW) | 979-8-3503-0208-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/CCGridW59191.2023.00077

Privacy-preserving Job Scheduler for GPU Sharing

Aritra Ray[1], Kyle Lafata[1], Zhaobo Zhang[2], Ying Xiong[2], and Krishnendu Chakrabarty[3]

[1]Duke University, *NC, USA*
[2]Futurewei Technologies, *CA, USA*
[3]Arizona State University, *AZ, USA*
[1]{*aritra.ray, kyle.lafata*}*@duke.edu*
[2]{*zzhang1, yxiong*}*@futurewei.com*
[3]{*krishnendu.chakrabarty*}*@asu.edu*

***Abstract*—Machine learning (ML) training jobs are resource intensive. High infrastructure costs of computing clusters encourage multi-tenancy in GPU resources. This invites a scheduling problem in assigning multiple ML training jobs on a single GPU while minimizing task interference. Our paper introduces a clustering-based privacy-preserving job scheduler that minimizes task interference without accessing sensitive user data. We perform ML workload characterization, made available publicly [1], and do exploratory data analysis to cluster ML workloads. Consequently, we build a knowledge base of inter and intra-cluster task interference to minimize task interference.**

I. Introduction

Enterprise solutions encourage multi-tenancy [2]–[4] of GPU resources for ML training jobs. In multi-tenancy, the *competitor job* plays a significant role in the training time of the concerned ML job as they compete for the same GPU resources. For efficient scheduling of ML training jobs on the same resources, existing schedulers for GPU sharing [5] [6] does not address preserving user privacy. User-sensitive model information can be defined as accessing sensitive hyper-parameters like batch size, model architecture, and the number of epochs to make scheduling decisions. Our paper proposes a privacy-preserving job scheduler that does not access user-sensitive information to make efficient scheduling decisions.

II. Design Methodology

We perform workload characterization for five different ML training jobs over 66 metrics like network and memory usage, CPU and GPU utilization, and monitor several system statistics. To preserve user privacy, we remove all user-sensitive data like model architecture, epoch, batch size, the dataset used, and the task ID. Next, for our two target variables, maximum GPU utilization percent (*MGUP*) and maximum GPU time spent accessing memory percent (*MGMAP*), we evaluate their correlation to the other attributes through exploratory data analysis tools. Consequently, to bolster the size of the dataset regarding the target variables and selected attributes, we perform interpolation concerning the batch size. The enlarged dataset is then fed into a clustering algorithm to group the ML jobs. Fig. 1 shows the synthetic dataset generation methodology.

For the clustered jobs, we perform extensive experiments to determine the inter and intra-cluster task interference through

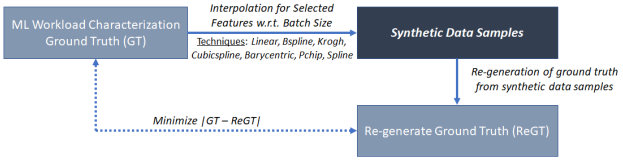

Fig. 1. Synthetic Data Generation Methodology

two metrics of *Individual Slow Down (ISD)* and *Packing Saving (PS)*. *ISD* is calculated as the percent of additional time required to execute a job in parallel to its competitor task to the time required for it to run individually in the GPU. *PS* is calculated as the percent of time saved when two jobs run concurrently to them executing sequentially. Based on the task interference analysis for a set of unknown jobs, we propose our scheduling algorithm that balances both *PS* and *ISD*.

III. Evaluations

In this section, we discuss our evaluation results in detail.

- **Data Collection**: We characterize ML workload over 66 different attributes for five ML training jobs. The five ML training jobs include image classification, image segmentation, Reinforcement Learning (RL), Generative Adversarial Networks (GAN), and Natural Language Processing (NLP). In regards to image classification, workload characterization is done for MobileNet [7], ResNet-50 [8], InceptionV3 [9], EfficientNetV2 [10] and NasNet-mobile [11] on the GTSRB dataset [12]. For image segmentation, characterization is done on the oxford-IIIT pet dataset[1]. We implement the cart-pole environment from OpenAI for characterizing a task from the RL domain. We implemented standard Deep Convolutional GAN architecture for MNIST [13] and five small BERT [14] models for sentiment analysis on IMDB movie review dataset for ML workload characterization in GAN and NLP domain respectively. The 49 collected data points featuring 66 attributes are released publicly [1].
- **Feature Selection**: We perform exploratory data analysis to study the impact of the 66 collected attributes over our two target variables, namely, (*MGUP*) and (*MGMAP*).

[1]https://www.robots.ox.ac.uk/vgg/data/pets/

979-8-3503-0208-0/23/$31.00 ©2023 IEEE
DOI 10.1109/CCGridW59191.2023.00077

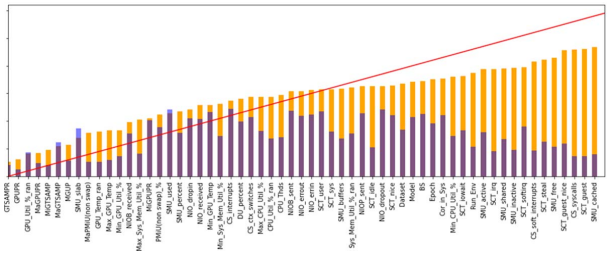

Fig. 2. Shows the relative ranking of all the attributes for our target variable, *MGUP*. Mean ranks are represented by orange bars, with the standard deviation of the attributes for all the feature selection techniques marked by blue bars. With every new attribute, the red line, which refers to the mean average rank, grows by one.

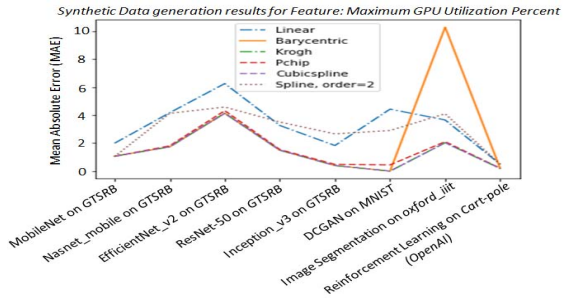

Fig. 3. Reconstruction error when synthetic data is used to re-generate ground truth. Low error across several cases demonstrates successful reconstruction of ground truth from generated synthetic data.

Correlation to the attributes is analyzed through Pearson's correlation coefficient, while feature extraction is done through random forests, ridge, and lasso regression. Implementation details regarding the construction of the decision trees and feature space exploration can be found in our GitHub repo. [1] Feature extraction results for *MGMAP* are presented in Fig. 2.

- **Synthetic Data Generation**: Various interpolation techniques like linear, spline, krogh, cubic, barycentric, and pchip concerning batch size are used to generate synthetic data to increase the dataset size four times to its original. To estimate the quality of the synthetic samples, we used the synthetic samples as pseudo-ground truth to reconstruct the ground truth. The re-constructed error for target variable *MGMAP* across the dataset is as shown in Fig. 3.
- **Task Interference Analysis through Clustering**: We perform K-means clustering on our two target variables, alongside the selected feature, maximum GPU time spent accessing memory percent (*MGTSAMP*). Computing the distortion score through the elbow method yielded the presence of 3 clusters, which are presented in Fig. 4. We run extensive experiments based on the clustering results to build a knowledge base of inter and intra-cluster task interference regarding *ISD* and *PS*. The results are as presented in Fig. 5.
- **Scheduler Analysis**: Based on our inference in terms of *ISD* and *PS* for the inter and intra-cluster task interference, we design our scheduling algorithm. From Fig. 5, we infer that jobs from cluster 1, when executed in parallel with jobs from cluster 2, yield the best *PS* while encountering the least *ISD*. For verification, we scheduled 24 ML training jobs, with 8 jobs from each cluster, randomly sampled from the five categories of ML jobs. The detailed results of the specific jobs that were scheduled can be found in our GitHub repo [1]. The scheduling results are demonstrated in Fig. 6.

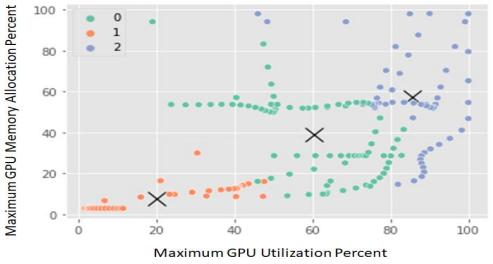

Fig. 4. Clustering results based on MGUP, MGMAP, and MGTSAMP

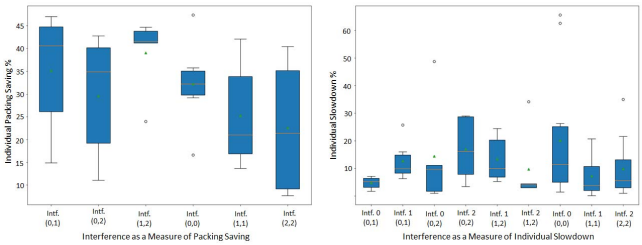

Fig. 5. Inter and Intra class task interference measured as *PS* percent on the left, followed by *ISD* percent on the right. For fig. on the left, Intf. (0,1) indicates the interference (*PS*) of job from cluster 1 when executed in parallel with job from cluster 0, on NVIDIA v100 GPU. For fig. on the right, Intf.1 (0,1) indicates the interference (*ISD*) of job from cluster 1 when executed in parallel with job from cluster 0, on NVIDIA v100 GPU.

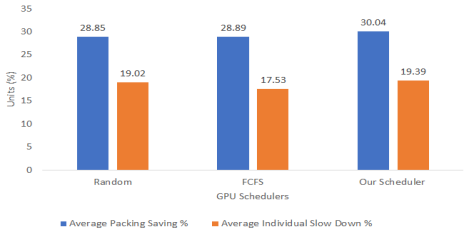

Fig. 6. Comparison of our proposed scheduler to random and First Come First Serve (FCFS) ones in terms of individual slowdown percent, and packing saving percent.

IV. Conclusion

Our proposed scheduler handles the trade-off between *ISD* and *PS* efficiently. We see a 1.19 percent gain in *PS* with 0.37 percent fall in the *ISD* compared to the random scheduler. Similar trade-off is achieved in comparison to FCFS scheduler also. To schedule unknown jobs, we can execute the jobs for one epoch, and using the attribute values of *MGUP*, *MGMAP*, and *MGTSAMP*, can assign it to a cluster. Based on our knowledge base, we can then schedule the jobs that efficiently to balance the trade-off between *ISD* and *PS* while addressing user privacy.

REFERENCES

[1] 2023. [Online]. Available: https://github.com/CentaurusInfra/alnair/tree/main/open-data/gpu-sharing

[2] M. Jeon, S. Venkataraman, A. Phanishayee, J. Qian, W. Xiao, and F. Yang, "Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, 2019, pp. 947–960.

[3] K. Ranganath, J. D. Suetterlein, J. B. Manzano, S. L. Song, and D. Wong, "Mapa: Multi-accelerator pattern allocation policy for multi-tenant gpu servers," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–14.

[4] F. Yu, D. Wang, L. Shangguan, M. Zhang, C. Liu, and X. Chen, "A survey of multi-tenant deep learning inference on gpu," *arXiv preprint arXiv:2203.09040*, 2022.

[5] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang *et al.*, "Gandiva: Introspective cluster scheduling for deep learning," pp. 595–610, 2018.

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

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[10] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[11] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[12] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark," in *International Joint Conference on Neural Networks*, no. 1288, 2013.

[13] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.