Poster ID: 06

Fahao Chen, 1st Year Ph. D Student
Computer Organization Lab., The University of Aizu

Poster Session at Graduate School Information Fair

Hare: Exploiting Inter-job and Intra-job Parallelism of Distributed Machine Learning on Heterogeneous GPUs

General Background

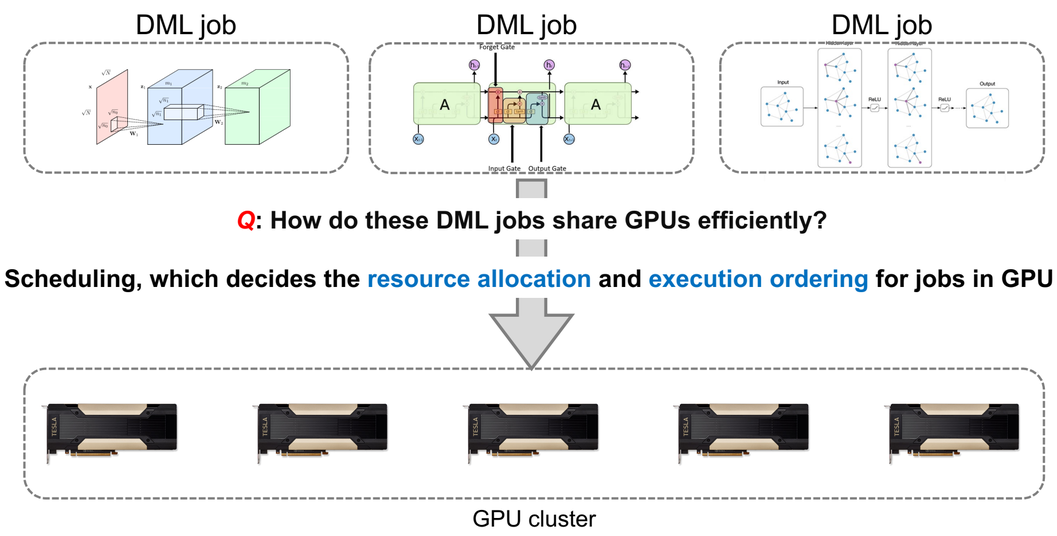


Fig. 1 Distributed machine learning (DML) job scheduling on the GPU cluster

Q: How do these DML jobs share GPUs efficiently?

A: Scheduling, which decides the resource allocation and execution ordering for jobs in GPU cluster.

In practice, it is rare to assign a dedicated GPU cluster to each DML job, due to low resource utilization. Instead, a common practice is to let multiple jobs share these GPUs. A critical research challenge is how to efficiently schedule these jobs on GPUs, which is particularly concerned by public or private cloud data centers that offer learning services while desiring high hardware resource utilization.

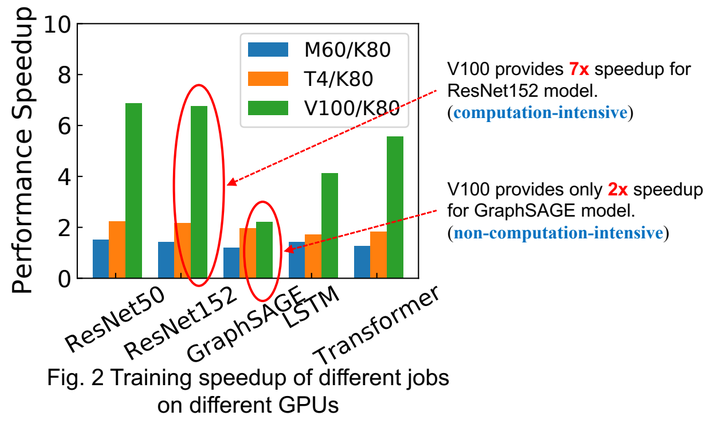


Fig. 2 Training speedup of different jobs on different GPUs

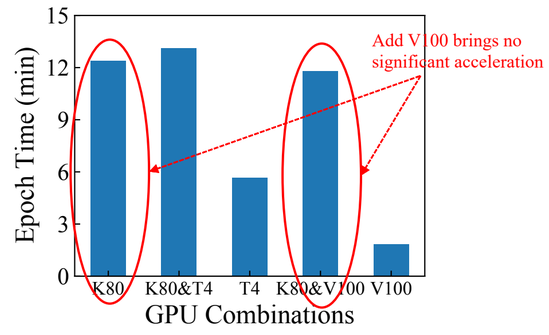


Fig. 3 Epoch time of ResNet152 under different GPU combinations

GPU heterogeneity and inter-job parallelism. We find that different GPUs provide different performance speedup for learning jobs, mainly because of the heterogeneity of model (such as model architecture) and hardware, as shown in Fig. 2.

GPU heterogeneity and intra-job parallelism. We find that mixing different GPUs is not always helpful, as shown in Fig. 3. For example, compared to a pure K80 cluster, adding faster T4 or V100 brings no acceleration. That is because the gradient synchronization impedes early completed GPUs to move to the next-round training.

Task Switching Cost. Exploiting intra-job parallelism on heterogeneous GPU environment is critical for accelerating training and increasing hardware resource utilization. We further find that such an algorithm inevitable generates results with frequent task switching on GPUs.

Hare: A DML job scheduler in the heterogeneous GPU cluster

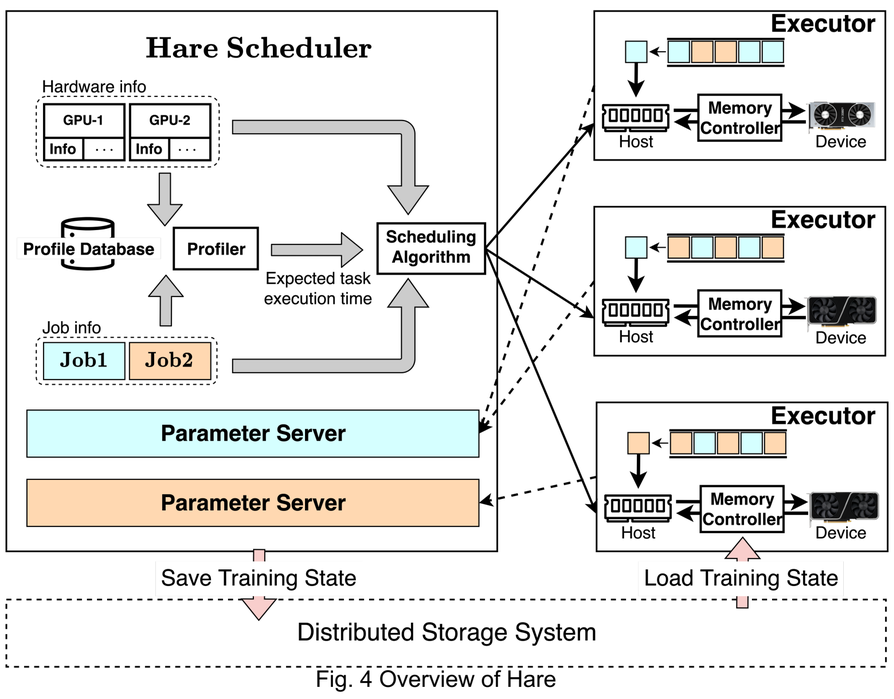
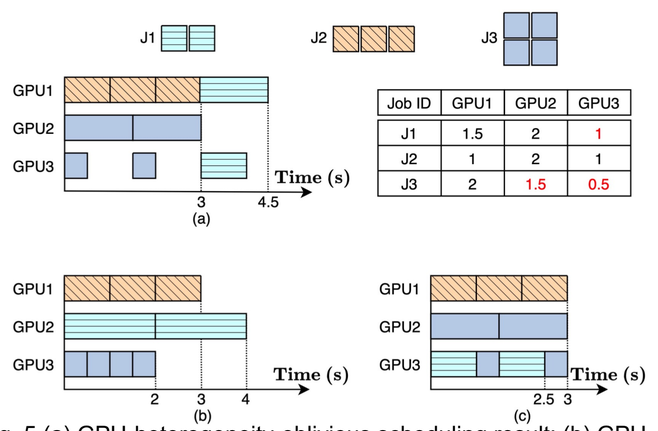


Fig. 4 Overview of Hare

- **High training efficiency:** Given a number of DML jobs, Hare needs to schedule them on a cluster to finish the training as fast as possible.
- **High GPU utilization:** Hare aims to improve the GPU utilization by reducing system cost and minimizing GPU idle time.
- **Starvation-free:** The scheduling algorithm design should be starvation-free so that every task has a chance to run.



Job ID	GPU1	GPU2	GPU3
J1	1.5	2	1
J2	1	2	1
J3	2	1.5	0.5

Fig. 5 (a) GPU-heterogeneity-oblivious scheduling result; (b) GPU-heterogeneity-aware scheduling result, but without exploiting intra-job parallelism; (c) A better scheduling result.

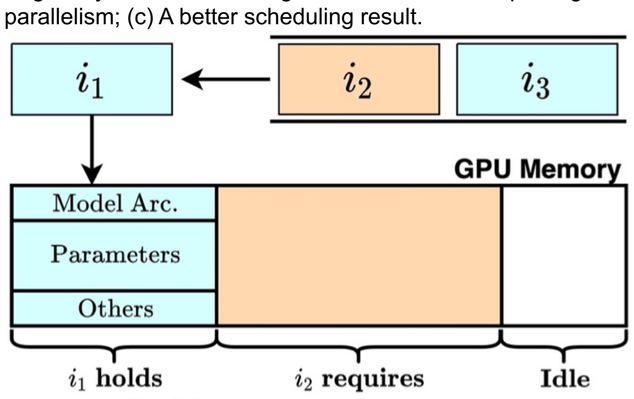


Fig. 6 Speculative memory management

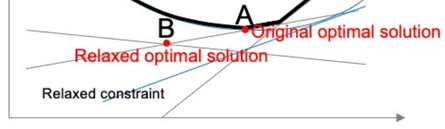
Design 1: Task Scheduling Algorithm

Objective: minimize the total weighted job completion time (JCT)

Theorem-1: the above problem is NP-hard

Algorithm:

- Relax the problem to give a lower bound.
- Decide the scheduling ordering of tasks according to the solution of relaxed problem. Then, we Greedily assign tasks to GPUs with the earliest available time



Design 2: Fast Task Switching

Early task cleaning. Hare deletes intermediate data of each layer once its backward training completes. Released GPU memory can be used for pre-loading data of the next task, so that it can start earlier.

Speculative memory management. Hare uses a simple heuristic that always gives higher GPU memory priority to the next tasks, and greedily keeps models of latest completed tasks until they cannot be accommodated.

Evaluation

	VGG19	ResNet50	Inception V3	Bert_base	Transformer	DeepSpeech	FastGCN	GraphSAGE
Default	3288.94 ms (98.21%)	5961.16ms (97.37%)	7807.43 ms (96.99%)	9016.99 ms (93.95%)	5257.17 ms (95.41%)	5125.64 ms (94.15%)	5327.24 ms (98.47%)	5213.54 ms (98.29%)
PipeSwitch	4.01 ms (2.40%)	4.75 ms (5.46%)	5.03 ms (2.39%)	12.57 ms (1.99%)	10.34 ms (2.03%)	8.91 ms (1.59%)	2.86 ms (7.56%)	2.42 ms (8.64%)
Hare	2.77 ms (1.82%)	2.04 ms (3.71%)	2.46 ms (1.43%)	5.03 ms (1.13%)	5.79 ms (1.36%)	4.27 ms (1.25%)	1.83 ms (4.53%)	0.96 ms (3.36%)

Tab. 1 Average Task Switching Time of Different Jobs

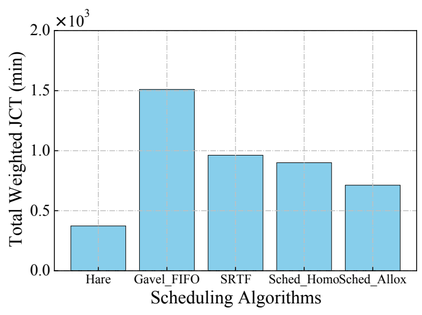


Fig. 7 Total weighted JCT

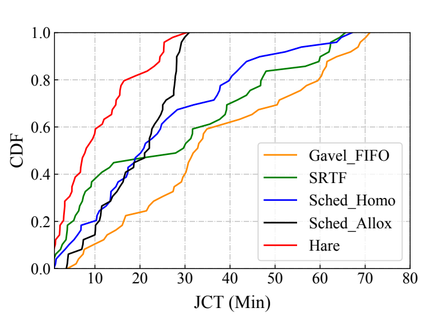


Fig. 8 CDF of job completion time

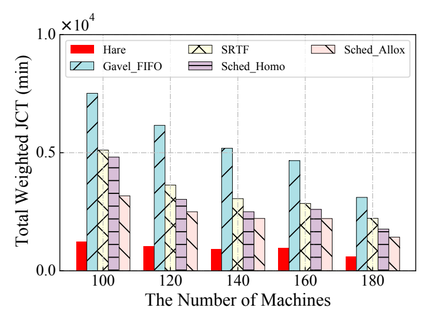


Fig. 9 Performance under different number of GPUs

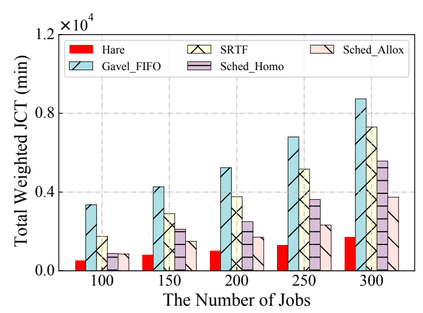


Fig. 10 Performance under different number of jobs

We first study the benefits of fast task switching by showing the average switching time of different jobs in Tab 1. A default task switching scheme, without any optimization, needs more than 3000ms for all jobs. PipeSwitch can reduce the average switching time to 12.57ms for Bert_base and less for others. The maximum switching time of Hare is no more than 6ms. The proportion of task switching time to the total task time is also shown in the table. We can see that Hare constrains the task switching overhead within 2% for most of models, and the largest overhead under FastGCN is no more than 5%.

The total weighted job completion time (JCT) of several schemes running on the testbed and the simulator is shown in Fig. 7. Compared with other schemes, Hare can reduce total weighted JCT by 47.6% to 75.3%, significantly outperforming other schemes. Fig. 8 shows the cumulative distributed function (CDF) of JCT of all jobs. We can see that about 90.5% of jobs can complete within 25 minutes by Hare, while Sched_Allox and Sched_Homo can complete only 66.7% and 56.5%, respectively.

We study the influence of number of GPUs in Fig. 9. The number of ML jobs is set to 200. The weighted JCT of all schemes decreases as more GPUs are used. Hare always outperforms other schemes under all cases. Sched_Allox is slower than Hare by about 2x, but it is still significantly faster than others, thanks to its heterogeneity-aware design.

We then consider 160 GPUs and change the number of jobs from 100 to 300 to see how it affects the performance. As shown in Fig. 10, as the number of jobs increases, the total weighted JCT grows under all schemes. Meanwhile, the performance gaps between Hare and other schemes become bigger. For example, Hare outperforms others by 54.6%-80.5% when processing 300 jobs.

Conclusion

- **Hare:** An DML jobs training system with the objectives of ***high training efficiency***, ***high GPU utilization***, and ***starvation-free***
- **Task Scheduling Algorithm:**
  - Minimize total weighted JCT
  - Reduce total weighted JCT by **47.6% to 75.3%** over other schemes
- **Fast Task Switching:**
  - Hare can reduce the task switching overhead to less than **6ms**.

Reference

1) Xiao, Wencong, et al. "Gandiva: Introspective cluster scheduling for deep learning." 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18).
2) Gu, Juncheng, et al. "Tiresias: A GPU cluster manager for distributed deep learning." 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19).
3) Peng, Yanghua, et al. "Optimus: an efficient dynamic resource scheduler for deep learning clusters." Proceedings of the Thirteenth EuroSys Conference. Mahajan, Kshiteej, et al. "Themis: Fair and efficient {GPU} cluster scheduling." 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20).
4) Xiao, Wencong, et al. "AntMan: Dynamic Scaling on GPU Clusters for Deep Learning." 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20).
5) Le, Tan N., et al. "Allox: compute allocation in hybrid clusters." Proceedings of the Thirteenth EuroSys Conference. 2020.
6) Qiao, Aurick, et al. "Pollux: Co-adaptive cluster scheduling for goodput-optimized deep learning." 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)