Biweekly Update: Traffic Pattern Analysis and Comparison of Distributed Deep Learning Models

Li He V00972954

1 Literature Review:

Read and analyzed several key papers related to distributed deep learning and network-aware optimization:
According to **CASSINI**[1], it introduce a network-aware job scheduling system specifically designed to optimize the performance of machine learning (ML) workloads in large-scale distributed clusters. Their approach consider interleaving the communication patterns of different training jobs when placing them on servers. CASSINI is an inter-job communication scheduler that aims to reuse different network links over time by assigning a time-dimension offset to each job. However, dynamic networks and jobs can affect the actual traffic pattern of each job. Thus, merely setting a time offset for each job cannot eliminate communication contention among jobs. **Crux** [2] proposes the concept of GPU intensity, and reduces the GPU utilization problem to a flow optimization problem. To approximate the optimal flow scheduling in real-world multi-tenant GPU clusters, Crux proposes GPU intensity-based path selection, priority assignment, and priority compression for DLT jobs.

In the paper **PipeDream**[3], they propose pipeline parallelism, a combination of data and model parallelism with pipelining, which showed pipelining can be used to accelerate the DNN training. [4] designed a new aggregation hardware accelerator to support multiple active training jobs and demonstrated the benefits of in-network aggregation extend to non-aggregation traffic. In order to fully overlap gradient synchronization communication with computation with minimal staleness, **SAPipe** [5] introduces partial staleness, which restricts the number of layers learned with stale gradients. To further mitigate convergence issues caused by staleness, SAPipe adopts weight prediction and delay compensation.

In this paper **MLTCP**[6], they introduces a novel approach to leverage congestion control algorithms to approximate interleaved flow schedules for DNN flows in a distributed manner. The key idea is to dynamically adjust the flow congestion window (or sending rate) to induce a sliding effect so that the DNN jobs automatically converge to approximately optimal interleaving. [7] demonstrates that for a specific combination of jobs, introducing unfairness creates a desirable side effect that improves the training time of all jobs competing for bandwidth. A recent advancement in this domain is **TopoopT** [8] , which introduces a co-optimization framework for network topology and parallelization strategies in distributed training jobs.

These papers provided insights into communication bottlenecks, scheduling strategies, and optimization techniques for distributed deep learning.

2 Tool Exploration:

Discovered ASTRA-SIM (`http://github.com/astra-sim/astra-sim`), which is an open-source simulation methodology for modeling distributed training systems. ASTRA-SIM can model various parallelization strategies and the overlapping of compute with comm kernels. Additionally, its network back-end can simulate the comm operations in detail.

ASTRA-SIM supports:

- Modeling of various parallelization strategies (e.g., data parallelism, model parallelism).
- Simulation of computation-communication overlap.
- Evaluation of network topologies and communication kernels.

This tool is highly relevant for simulating and analyzing traffic patterns in distributed deep learning systems.

Exploring Compute Canada:

Identified Compute Canada as a potential platform for capturing real-world network traffic during distributed training. From my initial understanding, Compute Canada allows for network traffic capture with code, which could complement the simulations done in ASTRA-SIM. Currently researching how to set up and use Compute Canada for this purpose.

3 Next Steps Planning

Setting Up ASTRA-SIM:

Installed and configured ASTRA-SIM on my local machine. Explored its documentation and example use cases to understand its capabilities. Started simulating basic distributed training scenarios to familiarize myself with the tool.

Compute Canada Set up:

Continue exploring Compute Canada's capabilities for network traffic capture. Set up a distributed training job on Compute Canada and capture network traffic using available tools.

References

[1] S. Rajasekaran, M. Ghobadi, A. Akella, and U. Austin, "Cassini: Network-Aware Job Scheduling in Machine Learning Clusters."

[2] J. Cao, Y. Guan, K. Qian, J. Gao, W. Xiao, J. Dong, B. Fu, D. Cai, and E. Zhai, "Crux: GPU-Efficient Communication Scheduling for Deep Learning Training," in *Proceedings of the ACM SIGCOMM 2024 Conference*. Sydney NSW Australia: ACM, Aug. 2024, pp. 1–15. [Online]. Available: https://dl.acm.org/doi/10.1145/3651890.3672239

[3] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons, and M. Zaharia, "PipeDream: generalized pipeline parallelism for DNN training," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. Huntsville Ontario Canada: ACM, Oct. 2019, pp. 1–15. [Online]. Available: https://dl.acm.org/doi/10.1145/3341301.3359646

[4] "In-network Aggregation for Shared Machine Learning Clusters."

[5] Y. Chen, C. Xie, M. Ma, J. Gu, Y. Peng, H. Lin, C. Wu, and Y. Zhu, "SAPipe: Staleness-Aware Pipeline for Data-Parallel DNN Training."

[6] S. Rajasekaran, S. Narang, A. A. Zabreyko, and M. Ghobadi, "MLTCP: A Distributed Technique to Approximate Centralized Flow Scheduling For Machine Learning," in *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*. Irvine CA USA: ACM, Nov. 2024, pp. 167–176. [Online]. Available: https://dl.acm.org/doi/10.1145/3696348.3696878

[7] S. Rajasekaran, M. Ghobadi, G. Kumar, and A. Akella, "Congestion control in machine learning clusters," in *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*. Austin Texas: ACM, Nov. 2022, pp. 235–242. [Online]. Available: https://dl.acm.org/doi/10.1145/3563766.3564115

[8] W. Wang, M. Khazraee, Z. Zhong, and M. Ghobadi, "TopoopT: Co-optimizing Network Topology and Parallelization Strategy for Distributed Training Jobs."