Efficient Pre-Training of LLMs via Topology-Aware Communication Alignment on More Than 9600 GPUs

Anonymous Authors[1]

Abstract

The scaling law for large language models (LLMs) depicts that the path towards machine intelligence necessitates training at large scale. Thus, companies continuously build large-scale GPU clusters, and launch training jobs that span over thousands of computing nodes. However, LLM pre-training presents unique challenges due to its complex communication patterns, where GPUs exchange data in sparse yet high-volume bursts within specific groups. Inefficient resource scheduling exacerbates bandwidth contention, leading to suboptimal training performance. This paper presents Arnold, a scheduling system summarizing our experience to effectively align LLM communication patterns with data center topology at scale. An in-depth characteristic study is performed to identify the impact of physical network topology to LLM pre-training jobs, and a scheduling algorithm is developed to effectively align communication patterns with the physical network topology in modern data centers. In production training, our scheduling system improves the end-to-end performance by 10.6% when training with more than 9600 GPUs, a significant improvement for our training pipeline.

1. Introduction

Pre-training large language models (LLMs) at scale is a highly resource-intensive process that requires vast computational infrastructure. The performance of LLM training is fundamentally dependent on three factors: dataset size, computational power, and model parameters (Kaplan et al., 2020). To meet these demands, companies continually enhance their computing infrastructure by incorporating cutting-edge GPUs and redesigning network architectures (imbue team; Qian et al., 2024; Wang et al., 2024). However, LLM pre-training presents unique challenges that distinguish it from conventional deep learning tasks — in this paper, we explore *how to develop an efficient resource scheduling mechanism to support the LLM training workflow to accommodate the resource-intensive and complex communication patterns in modern data centers*.

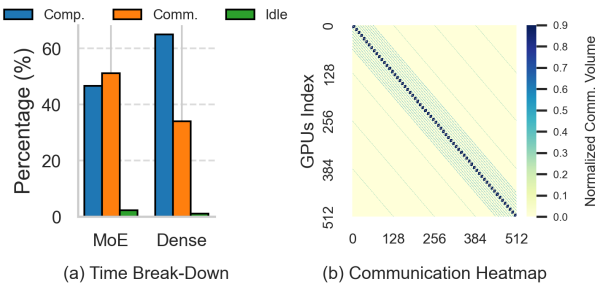

Figure 1: Communication characteristics of LLMs training.

Existing cluster schedulers (e.g., (Weng et al., 2022; 2023; Xiao et al., 2018; 2020; Choudhury et al., 2024; Cao et al., 2024)) fail to integrate network topology-aware scheduling specific to LLM workloads. The primary limitation is their lack of awareness of the high-volume, yet sparsely active distributed communication patterns inherent in LLM training. For example, Figure 1 (a) indicates that 30% - 50% of the time is spent on communication during production LLM training, but studies (Wang et al., 2024) show that more than 99% of the GPU pairs do not exhibit direct traffic, with data exchange occurring exclusively within specific communication groups, as shown in Figure 1 (b). Meanwhile, modern GPU clusters use multi-tier, fat-tree network topologies (Al-Fares et al., 2008) (Figure 3), and inefficient job placement leads to significant bandwidth loss and communication overhead. Current schedulers are not designed to optimize network-aware placement at the scale required by LLM pre-training jobs (LPJs).

To enable effective scheduling of LPJs in data centers, we identify two key challenges that limit the effectiveness of existing cluster schedulers.

- **Misalignment of communication patterns with data center topology.** Schedulers optimize GPU locality through bin-packing but lack awareness of LPJ communication structures. As shown in Figure 2, even if the scheduler packs an 4-node (32 GPUs) LPJ inside 2 leaf switches, the communication groups may still not be aligned, because both DP and PP groups engage cross-spine-switch

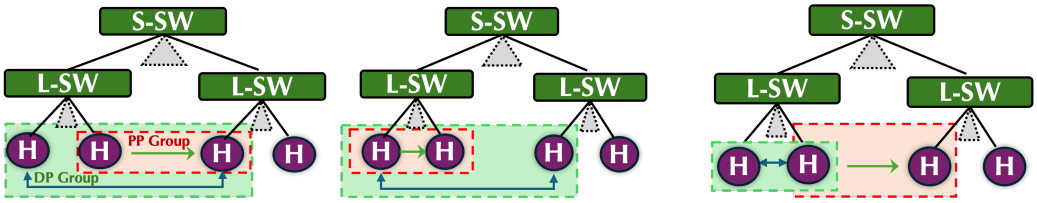

Figure 2: Alignment of communication groups and data center topology.

communication that has a longer distance. This misalignment stems from the scheduler's lack of awareness of the LPJ's communication structure at scheduling time, limiting its ability to allocate GPU resources according to the job's communication patterns.

- **Unaddressed trade-offs across communication dimensions.** Figure 2 shows two potential alignments of the LPJ, with one prioritizing DP communication and the other prioritizing PP. This presents a fundamental trade-off between the two, because DP and PP are orthogonal parallelism strategies widely used in LLM training. Each GPU participates in both a DP and an PP group, making it impossible to perfectly align both communication patterns simultaneously.

To address the challenges, we present Arnold, a system that co-designs training frameworks and cluster scheduling, effectively aligning LPJs with modern data center network topology. To optimize training performance, we performed an in-depth characterization study to investigate the impact of physical network topology on LLM training. Based on the observation, we devise a scheduling algorithm to reduce the maximum weighted spread of communication groups for LPJs. We also develop a resource management policy that manages job queues to reserve nodes for imminent LPJs.

Through trace-based experiments, we show the effectiveness of our scheduling algorithm by benchmarking against other SOTA algorithms. We also perform a production run with 9600+ GPUs and show our proposed system improves the end-to-end training performance by 10.6%.

2. Design

Data center topology. Figure 3 gives an overview of our HPC cluster, which is similar to other modern data centers. More than 2000 nodes are interconnected by three layers of switches, forming a CLOS-like topology (Clos, 1953). The leaf switch is denoted as s0, which interconnects nodes within the same rack. Then, several s0 switches link to a spine switch (s1), forming a minipod of nodes. Finally, s1 switches link to core switches, enabling communication between minipods. The switches in each layer have 32 ports both for uplinks and downlinks. The greatest number of

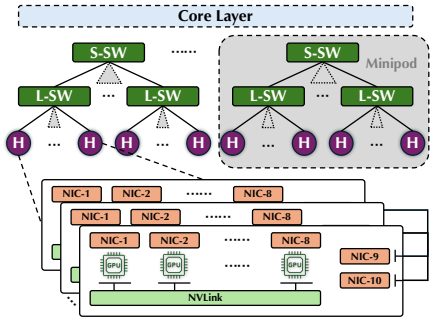

Figure 3: Data center topology.

hops occurs when the nodes of two different minipods communicate with each other. The compute nodes are equipped with 8 H800 GPUs, each of which is connected to an InfiniBand (NVIDIA, a) NIC. GPUs within a node are connected by high-bandwidth links such as NVLink (NVIDIA, b) with a bandwidth of 400Gbps, while inter-node communication is achieved via the InfiniBand network.

Characterization. We proportionally down-scaled a production model and ran the workload with 96 GPUs, spanning 2 minipods, to further understand the impact of network topology on LLM training.

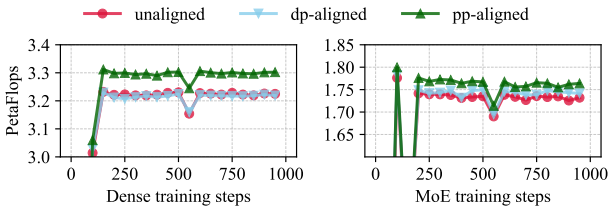

Figure 4: Comparison of three different placement strategies for two types of LLMs.

Figure 4 shows the throughput of the LPJ under the three different placement strategies. The throughput becomes stable after 200 steps except for a slight fluctuation around the 550th step due to garbage collection. PP-aligned placement consistently outperforms the other two, demonstrating that prioritizing PP group communication leads to improved performance. The average improvement for the dense model

and the MoE model is 2.3% and 1.8% respectively. For the dense model, we also observe that the PP communication dominates the communication overhead, since prioritizing the placement of DP groups leads to no speedup. For the MoE model, reducing the spread of both the DP and PP groups contributes to performance gains, with the optimizations of the PP group providing more improvements.

Figure 5 shows that if we scale the model size by adding more layers, the performance improvement continues to increase. We attribute this to communication being the primary performance bottleneck, with larger models further amplifying communication volume. Thus, more pronounced benefits are obtained as the model size increases.

We further examine the sensitivity of training performance to intra-minipod network topology by varying node placement within a single minipod. For a dense 24B parameter model, the maximum observed performance variation is 0.3%, and the impact is negligible for other models. Since the communication overhead of a group is typically dominated by the slowest link, and LPJ communication groups frequently span multiple minipods, we conclude that training performance is insensitive to intra-minipod topology.

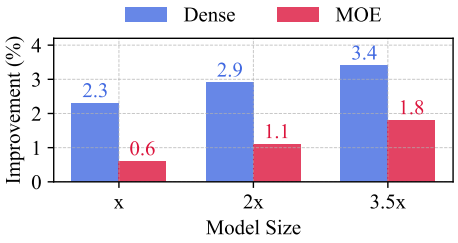

Figure 5: Performance improvement by scaling model sizes under the optimized alignment.

We repeated the characterization experiment in another GPU cluster, and found that the best placement can be subjected to model sizes and GPU types. Since LPJs are typically scheduled in advance and deployed for a long duration, it is essential to perform a characterization beforehand to identify communication bottlenecks within the group. This enables the optimization of placement strategies accordingly and balances the trade-off.

Workload representation. Given a user-specified number of GPUs and degree of hybrid parallelism of an LPJ, job scheduling systems enqueue the job and perform resource scheduling to find the best placement in our GPU cluster. Arnold represents an LPJ by a communication matrix, where a row represents a PP group and a column represents a DP group. Formally, given a job specification including the total number of GPUs, the degree of PP, TP, and Arnold computes the size of the communication matrix using Equation 1.

$$\begin{aligned} DP &= \#GPUs/TP/PP \\ \#row &= DP/(8/TP) \\ \#col &= PP \end{aligned} \tag{1}$$

Scheduling algorithm. We identify that communication groups are homogeneous and synchronous for LPJs because nodes are gang-scheduled and must synchronize their gradients at the end of a training step. As a result, each PP group always starts approximately at the same time and performs the same amount of computation and communication. Similarly, DP groups perform gradient synchronization at the same time. The characteristics allow us to simplify the scheduling objective function by coarsening a scheduling unit as a communication group. We derive the following scheduling algorithm:

$$MIN \quad [\alpha \sum_j (y_j) + \beta T] \tag{2}$$

$$\text{s.t.} \quad \forall i : \sum_j s_{ij} \le T \quad \textbf{(Max Spread)} \tag{3}$$

$$\forall j : \sum_i p_{ij} \le c_j y_j \quad \textbf{(Capacity Const.)} \tag{4}$$

$$\forall i : \sum_j p_{ij} = 1 \quad \textbf{(Allocation Const.)} \tag{5}$$

$$\forall i, j : p_{ij} \le s_{ij} \quad \textbf{(Minipod Selection)} \tag{6}$$

$$\forall j : y_j \in \{0, 1\} \tag{7}$$

$$\forall i, j : s_{ij} \in \{0, 1\}, p_{ij} \in [0, 1] \tag{8}$$

Where y_j indicates whether the j-th minipod is used and c_j is the normalized capacity of the minipod, updated dynamically based on the number of available nodes. s_{ij} denotes whether the i-th communication group is allocated to minipod j, p_{ij} denotes the percentage of the i-th communication group allocated to minipod j. T is an introduced auxiliary variable that allows us to minimize the maximum spread of communication groups. The weight parameters α and β represent the affinity that controls the trade-off between DP and PP groups, and $\alpha + \beta = 1$. Overall, minimizing T effectively consolidates communication groups into the smallest possible number of minipods, while $\sum_j (y_j)$ controls the spread of the other communication group.

The objective function can be solved using off-the-shelf mixed-integer programming (MIP) solvers efficiently for online scheduling (Bolusani et al., 2024). After solving the MIP, we assign continuous rank indices to nodes belonging to the same minipod to reduce cross-switch communication within each communication group.

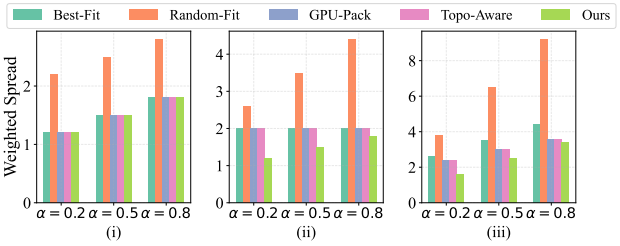

Figure 6: Simulation expriments.

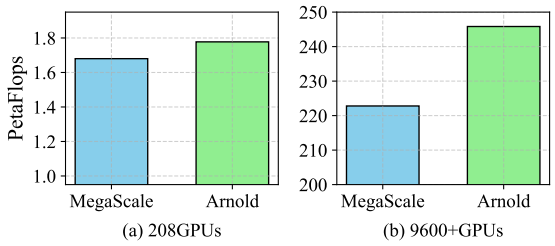

Figure 7: Cluster experiments

3. Experiments

We evaluate Arnold using both simulation and real-cluster experiments. To benchmark scheduling algorithms cost-effectively, we develop a simulator, as direct evaluation on production clusters is prohibitively expensive. After identifying the highest-performing scheduling algorithm through simulation, we deploy it on our production cluster to validate its effectiveness under real workloads.

Simulation experiment. We use the sum of the maximum spread of DP and PP group as the metric. We compare the scheduling algorithm with the following baselines.

1. Best-fit (Panigrahy et al., 2011) assigns the nodes to the minipod with the least remaining resources.
2. Random-fit (Weng et al., 2023) assigns nodes to minipods randomly such that the assignment is balanced.
3. GPU-packing (Weng et al., 2022; Xiao et al., 2018) packs multiple jobs to the same GPU. We modify the algorithm to pack multi-GPU jobs to a minipod to satisfy the network topology semantics.
4. Topo-aware (Amaral et al., 2017) recursively bi-partitions the physical graph and maps the job graph to the sub-graphs (Hierarchical Static Mapping Dual Recursive Bi-partitioning (Ercal et al., 1988)). The graph bi-partitioning is implemented by the Fiduccia Mattheyses algorithm (Fiduccia & Mattheyses, 1982).

Table 1: Benchmark setting. Network topology $\{x\}, \{y\}$ represent {x} minipod and {y} nodes in total, and the numbers in job configurations are the degree of DP, TP, PP. The scheduling unit is the PP group.

Settings	Network Topology	Job Configs
(i)	3, 18	12, 4, 2
(ii)	5, 438	24, 4, 8
(iii)	11, 1019	46, 8, 8

We use 3 settings in the benchmark as listed in Table 1, where the network topology is taken from a subset of our GPU cluster, and the job configurations are representative for small, medium, large jobs respectively. We also vary the value of α to investigate different degree of affinity.

Figure 6 compares the performance of different algorithms. Our algorithm consistently outperforms other baselines and up to 1.67x compared to the best baseline. On average, it leads to 1.2x reduction of the weighted sum of the maximum spread for communication groups. In the simple topology (setting *(i)*), our algorithm achieves the same score as best-fit, gpu-pack and topo-aware, because the network topology and job configurations are relatively simple, so there is no room to improve the placement. For medium and large jobs, our algorithm is better than the other baselines due to the large search space of possible placement.

Cluster experiment. To evaluate Arnold in real-world environments, we run experiments in our GPU cluster. *The specific information such as the number of GPUs and the model size, is hidden due to business concerns.* One of our LLMs is a MoE variant and was trained previously with more than 9600 GPUs (1200+ nodes). We first run the experiment by scheduling the job with 208 GPUs, and validate the speedup achieved by Arnold. We then run the pre-training at full scale. We compare Arnold with an SOTA production system for LLMs, MegaScale (Jiang et al., 2024), which takes a full-stack solution to optimize LLMs training and scale to $O(10,000)$ GPUs.

Figure 7 illustrates the average throughput of the two systems. Arnold achieves an average speedup of 5.7% and 10.6% respectively. We observe that Arnold reduces the maximum spread for the DP group and the PP group by 3x and 2x in the medium-scale experiment, while for the full-scale experiment, the reduction is 1.5x and 1.3x. This is because it is more likely to spread nodes across minipods in the cluster for medium-scale experiment if not planned carefully. However, for the full-scale experiment, the requested GPUs take up more than 50% of the total number of GPUs in the cluster, so the space of scheduling is shrunk.

4. Conclusion

In this work, we present Arnold, a scheduling system that summarizes our experience in effectively scheduling LPJs at scale. We show the effectiveness both in simulation-based and real-world GPU clusters.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Al-Fares, M., Loukissas, A., and Vahdat, A. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, pp. 63–74, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581750. doi: 10.1145/1402958.1402967. URL https://doi.org/10.1145/1402958.1402967.

Amaral, M., Polo, J., Carrera, D., Seelam, S., and Steinder, M. Topology-aware gpu scheduling for learning workloads in cloud environments. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '17, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450351140. doi: 10.1145/3126908.3126933. URL https://doi.org/10.1145/3126908.3126933.

Bolusani, S., Besançon, M., Bestuzheva, K., Chmiela, A., Dionísio, J., Donkiewicz, T., van Doornmalen, J., Eifler, L., Ghannam, M., Gleixner, A., Graczyk, C., Halbig, K., Hedtke, I., Hoen, A., Hojny, C., van der Hulst, R., Kamp, D., Koch, T., Kofler, K., Lentz, J., Manns, J., Mexi, G., Mühmer, E., Pfetsch, M. E., Schlösser, F., Serrano, F., Shinano, Y., Turner, M., Vigerske, S., Weninger, D., and Xu, L. The scip optimization suite 9.0, 2024. URL https://arxiv.org/abs/2402.17702.

Cao, J., Guan, Y., Qian, K., Gao, J., Xiao, W., Dong, J., Fu, B., Cai, D., and Zhai, E. Crux: Gpu-efficient communication scheduling for deep learning training. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, pp. 1–15, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706141. doi: 10.1145/3651890.3672239. URL https://doi.org/10.1145/3651890.3672239.

Choudhury, A., Wang, Y., Pelkonen, T., Srinivasan, K., Jain, A., Lin, S., David, D., Soleimanifard, S., Chen, M., Yadav, A., Tijoriwala, R., Samoylov, D., and Tang, C. MAST: Global scheduling of ML training across Geo-Distributed datacenters at hyperscale. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pp. 563–580, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/choudhury.

Clos, C. A study of non-blocking switching networks. *The Bell System Technical Journal*, 32(2):406–424, 1953. doi: 10.1002/j.1538-7305.1953.tb01433.x.

Ercal, F., Ramanujam, J., and Sadayappan, P. Task allocation onto a hypercube by recursive mincut bipartitioning. In *Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications: Architecture, Software, Computer Systems, and General Issues - Volume 1*, C3P, pp. 210–221, New York, NY, USA, 1988. Association for Computing Machinery. ISBN 0897912780. doi: 10.1145/62297.62323. URL https://doi.org/10.1145/62297.62323.

Fiduccia, C. and Mattheyses, R. A linear-time heuristic for improving network partitions. In *19th Design Automation Conference*, pp. 175–181, 1982. doi: 10.1109/DAC.1982.1585498.

imbue team. From bare metal to a 70b model: infrastructure set-up and scripts. https://imbue.com/research/70b-infrastructure/. Accessed: 2024-12-01.

Jiang, Z., Lin, H., Zhong, Y., Huang, Q., Chen, Y., Zhang, Z., Peng, Y., Li, X., Xie, C., Nong, S., Jia, Y., He, S., Chen, H., Bai, Z., Hou, Q., Yan, S., Zhou, D., Sheng, Y., Jiang, Z., Xu, H., Wei, H., Zhang, Z., Nie, P., Zou, L., Zhao, S., Xiang, L., Liu, Z., Li, Z., Jia, X., Ye, J., Jin, X., and Liu, X. MegaScale: Scaling large language model training to more than 10,000 GPUs. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pp. 745–760, Santa Clara, CA, April 2024. USENIX Association. ISBN 978-1-939133-39-7. URL https://www.usenix.org/conference/nsdi24/presentation/jiang-ziheng.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

NVIDIA. Infiniband networking solutions. https://www.nvidia.com/en-us/networking/products/infiniband/, a. Accessed: 2024-12-01.

NVIDIA. Nvlink. https://www.nvidia.com/en-us/data-center/nvlink/, b. Accessed: 2024-12-01.

Panigrahy, R., Prabhakaran, V., Talwar, K., Wieder, U., and Ramasubramanian, R. Validating heuristics for virtual machines consolidation. Technical Report MSR-TR-2011-9, January 2011. URL https://www.microsoft.com/en-us/research/publication/validating-heuristics-for-virtual-machines-consolidation/.

Qian, K., Xi, Y., Cao, J., Gao, J., Xu, Y., Guan, Y., Fu, B., Shi, X., Zhu, F., Miao, R., Wang, C., Wang, P., Zhang, P., Zeng, X., Ruan, E., Yao, Z., Zhai, E., and Cai, D. Alibaba hpn: A data center network for large language model training. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, pp. 691–706, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706141. doi: 10.1145/3651890.3672265. URL https://doi.org/10.1145/3651890.3672265.

Wang, W., Ghobadi, M., Shakeri, K., Zhang, Y., and Hasani, N. Rail-only: A low-cost high-performance network for training llms with trillion parameters, 2024. URL https://arxiv.org/abs/2307.12169.

Weng, Q., Xiao, W., Yu, Y., Wang, W., Wang, C., He, J., Li, Y., Zhang, L., Lin, W., and Ding, Y. MLaaS in the wild: Workload analysis and scheduling in Large-Scale heterogeneous GPU clusters. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pp. 945–960, Renton, WA, April 2022. USENIX Association. ISBN 978-1-939133-27-4. URL https://www.usenix.org/conference/nsdi22/presentation/weng.

Weng, Q., Yang, L., Yu, Y., Wang, W., Tang, X., Yang, G., and Zhang, L. Beware of fragmentation: Scheduling GPU-Sharing workloads with fragmentation gradient descent. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pp. 995–1008, Boston, MA, July 2023. USENIX Association. ISBN 978-1-939133-35-9. URL https://www.usenix.org/conference/atc23/presentation/weng.

Xiao, W., Bhardwaj, R., Ramjee, R., Sivathanu, M., Kwatra, N., Han, Z., Patel, P., Peng, X., Zhao, H., Zhang, Q., Yang, F., and Zhou, L. Gandiva: Introspective cluster scheduling for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pp. 595–610, Carlsbad, CA, October 2018. USENIX Association. ISBN 978-1-939133-08-3. URL https://www.usenix.org/conference/osdi18/presentation/xiao.

Xiao, W., Ren, S., Li, Y., Zhang, Y., Hou, P., Li, Z., Feng, Y., Lin, W., and Jia, Y. AntMan: Dynamic scaling on GPU clusters for deep learning. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pp. 533–548. USENIX Association, November 2020. ISBN 978-1-939133-19-9. URL https://www.usenix.org/conference/osdi20/presentation/xiao.