SANCUS: *Staleness*-Aware *Communication*-Avoiding Full-Graph Decentralized Training in Large-Scale Graph Neural Networks (Extended Abstract)*

Jingshu Peng[1], **Zhao Chen**[1], **Yingxia Shao**[2], **Yanyan Shen**[3], **Lei Chen**[1] and **Jiannong Cao**[4]
[1]The Hong Kong University of Science and Technology
[2]Beijing University of Posts and Telecommunications
[3]Shanghai Jiao Tong University
[4]The Hong Kong Polytechnic University
{jpengab, zchenah,leichen}@cse.ust.hk, shaoyx@bupt.edu.cn, shenyy@sjtu.edu.cn,
csjcao@comp.polyu.edu.hk

Abstract

Graph neural networks (GNNs) have emerged due to their success at modeling graph data. Yet, it is challenging for GNNs to efficiently scale to large graphs. Thus, distributed GNNs come into play. To avoid communication caused by expensive data movement between workers, we propose SANCUS, a staleness-aware communication-avoiding decentralized GNN system. By introducing a set of novel bounded embedding staleness metrics and adaptively skipping broadcasts, SANCUS abstracts decentralized GNN processing as sequential matrix multiplication and uses historical embeddings via cache. Theoretically, we show bounded approximation errors of embeddings and gradients with convergence guarantee. Empirically, we evaluate SANCUS with common GNN models via different system setups on large-scale benchmark datasets. Compared to SOTA works, SANCUS can avoid up to 74% communication with at least 1.86× faster throughput on average without accuracy loss.

1 Introduction

The success of Graph Neural Networks (GNNs) [Kipf and Welling, 2017] has laid the foundation of recent advancement in the state of the art to model real-life graphs. In essence, GNNs are structure-aware models that construct the network architectures adapted to the original topology of the input graphs. By iteratively aggregating the neighbors of the targets, GNNs can exploit the structure and feature information at the same time. Despite the promising performance, the major challenge that limits the adoption of GNNs to large-scale graphs lies in the inability to utilize all data in finite time and the scalability of the algorithm itself. To mitigate the ***memory requirement*** with ever-increasing data and model size, distributed GNN processing is the inevitable remedy.

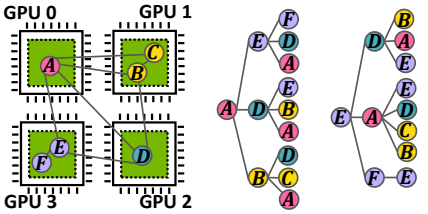


Figure 1: Distributed full-GNN example: nodes in same color are on same GPU. On the left, a 6-node graph is stored on 4 GPUs. On the right are the computational graphs of a 2-layer GNN for Node ***A*** and ***E***. During GNN neighborhood aggregation, intensive cross-device visits (10 times to update Node ***A*** and 9 for ***E***) to fetch neighbor data cause expensive communication overhead.

Compared to traditional graph processing or machine learning, new issues have emerged for distributed full-GNNs from the system perspective. Aside from the substantial memory footprint, distributed GNNs are memory-intensive as well as compute-intensive [Wang *et al.*, 2020; Thorpe *et al.*, 2021] due to coupled irregular neighbor fetching and iterative learning procedure. Consequently, the intensive communication, including not only gradients or parameters but also embeddings, makes efficient distributed GNN training more challenging. As exemplified by Figure 1, the training process needs to constantly query the target nodes, their neighbors, and their farther neighbors, to transfer both embeddings and gradients among workers. Thus, by virtue of such data movement, cross-device ***data communication*** can be one archenemy of efficient GNN processing. The incurred communication cost may account for 80% and even more of the total training time [Tripathy *et al.*, 2020; Gandhi and Iyer, 2021; Cai *et al.*, 2021].

In distributed training, the underlying system architecture of how workers communicate is crucial, especially for GNNs with substantial communication overhead. In general, two approaches exist: centralized and decentralized. Though most distributed GNN systems [Ma *et al.*, 2019; Zhu *et al.*, 2019; Jia *et al.*, 2020; Zheng *et al.*, 2020; Gandhi and Iyer, 2021; Min *et al.*, 2021] work in the popular centralized parameter server (PS) scheme, they often pay the price of heavy preprocessing and complex workflow, in pursuit of efficiency and scalability. By nature of GNNs, the in-

*This is an abridged version of a paper that won the Best Research Paper Award at VLDB 2022 [Peng *et al.*, 2022].

tensive communication between all the workers and the central PS plus the waiting time for stragglers may lead to high communication overhead [Cai *et al.*, 2021]. Decentralized architectures, however, can be more robust and easier to deploy, by avoiding the inconvenience in implementing and tuning a PS and centralized bottleneck bandwidth [Li *et al.*,]. Hence, Tripathy *et al.* [Tripathy *et al.*, 2020] offer CAGNET – so far the only SOTA decentralized parallel algorithms [Gholami *et al.*, 2018] adapted to GNNs. However, it calls for redundant and unnecessary broadcasts of all embeddings and gradients. Besides, all the workers must wait for stragglers to synchronize, leading to extra communication overhead.

In this paper, to fill this gap in efficient GNN processing, we propose SANCUS, a staleness-aware communication-avoiding decentralized GNN training system via adaptively skipping broadcast and caching historical embeddings with bounded staleness. To bypass the irregular communication between GPUs, we **firstly** revisit the parallel algorithms to distribute GNNs [Tripathy *et al.*, 2020] and decrease communication overhead in a fundamentally distinct way. As Figure 2 shows, each GPU loads the split submatrices without taking the semantic meaning into account, regarding the decentralized GNN processing purely as a sequence of matrix multiplication operations. The excessively large adjacency and embedding matrices are sliced into $\mathbf{A}_i$ and $\mathbf{H}_i$ ($i \in [1, 4]$) and distributed to GPU$_{i-1}$ with the full weight matrix $\mathbf{W}$. Then $\mathbf{H}_1$ to $\mathbf{H}_4$ are sequentially one-to-all broadcast to all GPUs in parallel. After 4 broadcasts, the whole $\mathbf{H}$ updates for a layer. By moving intact sub-matrices, SANCUS takes advantage of data parallelism to ***avoid communication*** of intensive neighbor fetching in Figure 1. **Secondly**, to further ***avoid communication*** under a decentralized scheme, we propose to cache and skip-broadcast the histrocial embeddings. We define historical embeddings as the embedding sub-matrices from earlier epochs in each distributed process, i.e., the sub-matrix $\mathbf{H}_i$ individually computed on GPU$_{i-1}$ in Figure 2. We utlize caching and design a novel skip-broadcast operator to support historical embeddings in SANCUS. **Thirdly**, to manage the ***system staleness*** caused by using mixed-version embeddings on each GPU, we propose the generalization of the widely-used *bounded gradient staleness* in centralized schemes [Cipar *et al.*,], to historical embeddings. We introduce a set of novel bounded embedding staleness metrics in decentralized GNNs. Particularly, to directly avoid communication, SANCUS adaptively skip-broadcasts embeddings within bounds and automatically reuses cached historical embeddings; otherwise, if embeddings become too stale, the results are broadcast and updated in cache among GPUs to keep the ***system staleness*** within bounds. Taking Figure 2 as a toy example, the embeddings $\mathbf{H}_3$ is not too stale, then SANCUS can skip broadcast once, now SANCUS only needs 7 broadcasts to update all embeddings in the 2-layer GNN. Again, it should be emphasized that there is no individual embedding fetching in GNN aggregation with SANCUS. Compared to the conventional distributed GNN in Figure 1, only to obtain the embeddings for node $\boldsymbol{A}$ and $\boldsymbol{E}$, 10 and 9 request-and-send operations are needed. To update all, 59 request-and-send operations are needed. However, SANCUS only needs as less as 7 broadcasts in total as Figure 2 shows. Also, it should be pointed out that SANCUS has few burden of preprocessing and can be easily applied to any distributed GNNs based on arbitrary matrix blocking and direct matrix operations.

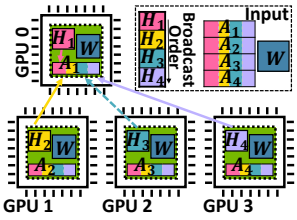


Figure 2: A toy 2-layer GNN example on SANCUS: GPU$_{i-1}$ keeps its shards of $\mathbf{H}_i$ and $\mathbf{A}_i$, with a full W; $\mathbf{H}_i$ are sent to all GPUs in order via one-to-all broadcast (arrows omitted without loss of generality). After 4 sequential broadcasts →,→,→, → and on-device computation, since the broadcasts are in parallel, all $\mathbf{H}_i$ updates for GPU$_{i-1}$. Next, SANCUS may tolerate $\mathbf{H}_3$ to skip 1 broadcast as shown by ⇢. In total, only $4 + 3 = 7$ broadcasts are needed.

In summary, our major contributions are: **(1) Problem Exploration.** We share a new perspective to accelerate GNNs by introducing historical embeddings with bounded staleness to decentralized GNNs, treating GNN processing purely as sequential matrix operations. **(2) Novel Metrics.** We introduce a set of novel bounded embedding staleness metrics in decentralized GNNs to effectively manage the system staleness caused by historical embeddings. **(3) New Criterion.** We provide the communication cost bound of SANCUS. We prove the approximation errors of the embeddings and gradients are bounded with convergence guarantee. **(4) Prominent Performance.** We evaluate SANCUS on large-scale benchmark datasets with prevalent GNN models via different system setups, to show its ability to generalize and superiority in efficiency while preserving effectiveness. SANCUS's performance with little or no accuracy loss demonstrates consistency with our theoretical findings.

2 Preliminaries

In this section, we introduce the related concepts of node-level graph representation learning. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph of order N with a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and nodes $\mathcal{V} = \{v_1, v_2, ..., v_N\}$. Consider the graph adjacency matrix $\mathbf{A}$, where the element A_{ij} in the matrix specifies the relation between the nodes v_i and v_j with $A_{ij} = 1$ if there exists an edge $(v_i, v_j) \in \mathcal{E}$ or otherwise $A_{ij} = 0$. $\mathbf{A}$ is symmetric since $\mathcal{G}$ is undirected. Denote $\hat{\mathbf{A}} = \bar{\mathbf{D}}^{-\frac{1}{2}}\bar{\mathbf{A}}\bar{\mathbf{D}}^{-\frac{1}{2}}$ as the adjacency matrix after symmetric normalization in GCN [Kipf and Welling, 2017], where $\bar{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ denotes the adjacency matrix with self-connections and $\bar{\mathbf{D}} \in \mathbb{R}^{N\times N} = \mathbf{D} + \mathbf{I}_N$ denotes the diagonal node degree matrix.

Without loss of generality, the ℓ-th layer propagation process of such GNNs [Abadal *et al.*, 2022; Wu *et al.*, 2021] can be formulated in matrix form as:

$$\mathbf{H}^{(\ell)} = \sigma\left(\mathbf{H}^{(\ell-1)}, \hat{\mathbf{A}}; \mathbf{W}^{(\ell-1)}\right) \tag{1}$$

where σ denotes the activation function such as ReLU and $\mathbf{W} \in \mathbb{R}^{F\times F}$ denotes the weight matrix. Initially, $\mathbf{H}^{(0)} = \mathbf{X}$

where $\mathbf{X} \in \mathbb{R}^{N \times F}$ is the node embedding matrix whose i-th row represents the length-F embedding vector of node v_i.

3 The SANCUS Framework

In this work, we propose SANCUS, an adaptive staleness-aware communication-avoiding decentralized GNN system. Fundamentally, SANCUS is simple yet effective which caches and reuses the stale historical embeddings and skips broadcast accordingly during the decentralized GNN training, based on a general communication-avoiding matrix blocking algorithm for parallel computing. The details can be referred in the original paper [Peng *et al.*, 2022].

We provide the overview of SANCUS in Figure 3. Primarily, there are five steps: (1) data loading, (2) staleness bound checking, (3) embedding broadcasting, (4) GNN model computing, and (5) results caching. Here, we briefly clarify these steps: (1) to begin with, the whole sparse adjacency matrix of the full graph and the dense embedding matrix are split into matrix blocks, then loaded to individual worker. Each worker keeps its own replica of the full model; (2) on each GPU, before broadcasting the last computing results, we check whether the staleness of historical embeddings are within proposed bounds. If the staleness is within bounds, the embedding broadcast is skipped and the cached historical embeddings are reused for this iteration's model computing; (3) otherwise, if the staleness exceeds the limit, the latest results are broadcast to all workers and updated in cache; (4) thus either latest embeddings or cached historical embeddings are loaded to the GNN model to compute; (5) finally, updated embeddings are dispatched to next iteration's staleness check before the broadcast.

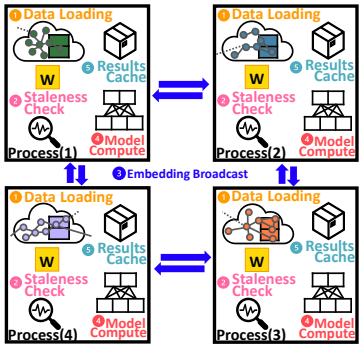


Figure 3: The overall architecture.

3.1 Historical Embeddings

Inspired by historical embeddings $\tilde{\mathbf{h}}^{(\ell)}$ [Chen *et al.*, 2018; Fey *et al.*, 2021], we generalize the idea to stale intermediate embedding results computed by other workers in distributed GNNs. Thus, the embedding matrix $\mathbf{H}^{(\ell)}$ in Equation (1) consists of two parts – the latest embedding submatrices from active workers which just broadcast the results and the historical embedding submatrices from stale workers whose embedding variation is small enough to be neglected. Historical embeddings are stored in cache on each GPU, preserving only the fresh ones. Let [] denotes the vertical concatenation of matrix blocks, then:

$$\begin{aligned}\mathbf{H}^{(\ell)} &= \sigma\left(\left[\mathbf{H}_i^{(\ell-1)}\right]_{i=1}^{P}, \hat{\mathbf{A}}; \mathbf{W}^{(\ell-1)}\right) \\ &\approx \sigma\left(\left[\mathbf{H}^{(\ell-1)}_{i:P(i)\Leftarrow\texttt{ACTIVE}} \mid \tilde{\mathbf{H}}^{(\ell-1)}_{i:P(i)\Leftarrow\texttt{STALE}}\right]_{i=1}^{P}, \hat{\mathbf{A}}; \mathbf{W}^{(\ell-1)}\right).\end{aligned} \tag{2}$$

3.2 Skip-Broadcast

We propose a communication primitive Skip-Broadcast that is efficient to implement and requires no centralized parameter servers. Skip-Broadcast allows seamless reshaping of the communication topology during training. To realize Skip-Broadcast, SANCUS keeps the state flag `Flag(i)` on each worker i to indicate the corresponding worker status for the embeddings $\mathbf{H_i}$ computed on that worker, where $i \in [1, p]$. Specifically, `Flag(i)==ACTIVE` means worker i needs to broadcast its latest version of embeddings $\mathbf{H_i}$. During the broadcast, the latest embeddings $\mathbf{H_i}$ should be sent to all other workers and cached there respectively. If `Flag(i)` turns to `STALE`, SANCUS can Skip-Broadcast $\mathbf{H}_i$ and let other workers utilize their cached stale embeddings.

3.3 Bounded Embedding Staleness

To manage system staleness, SANCUS supports bounded embedding staleness. Though bounded gradient staleness is deeply investigated [Cipar *et al.*, ; Jiang *et al.*, 2017; Xian *et al.*, 2021] in traditional distributed ML for stochastic gradient descent (SGD), its main purpose is to help SGD converge, mitigating negative effects from stale gradients. However, we actively utilize stale embeddings to avoid communication. By introducing a set of novel bounded embedding staleness metrics ϵ, we control the errors caused by staleness. We provide three staleness [Peng *et al.*, 2022] of historical embeddings, *Epoch-Fixed* ϵ_E, *Epoch-Adaptive* ϵ_A, and *Epoch-Adaptive Variation-Gap* ϵ_H. All metrics are defined and checked locally on each worker, we need no centralized or global monitor to break the decentralized scheme. Particularly, one can easily adapt the general definitions above to any specific distributed GNN systems as the metrics to study how the stale results affect the distributed training.

4 Theoretical Results

With SANCUS, we first obtain a tighter communication cost bound. Then, we bound the approximation errors of the embeddings and gradients and guarantee the convergence. The comprehensive theoretical results can be found in the original paper [Peng *et al.*, 2022].

5 Experiments

We evaluate SANCUS on five commonly-used [Abadal *et al.*, 2022] large-scale benchmark datasets Flickr, Reddit, Amazon, ogbn-products, and ogbn-papers100M [Zeng *et al.*, 2020; Hu *et al.*,]. We implement defined bounded staleness ϵ_E as SCS-E, ϵ_A as SCS-A, and ϵ_H as SCS-H correspondingly. Additionally, we implement SkipG [Miao *et al.*, 2021] on bounded gradient staleness from traditional distributed training. Our experiments are performed on four different GPU configurations: ① eight RTX 2080 Ti connected

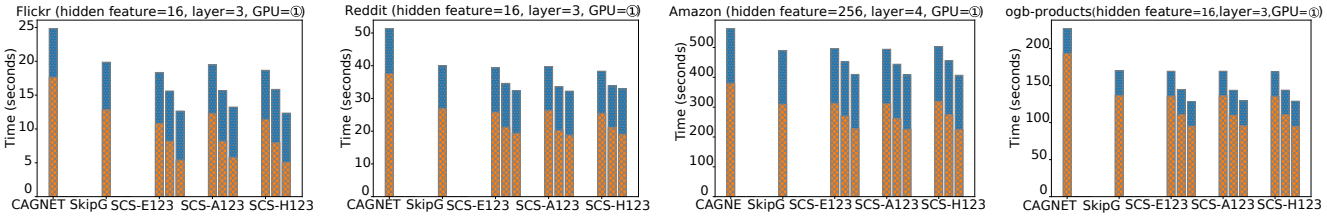
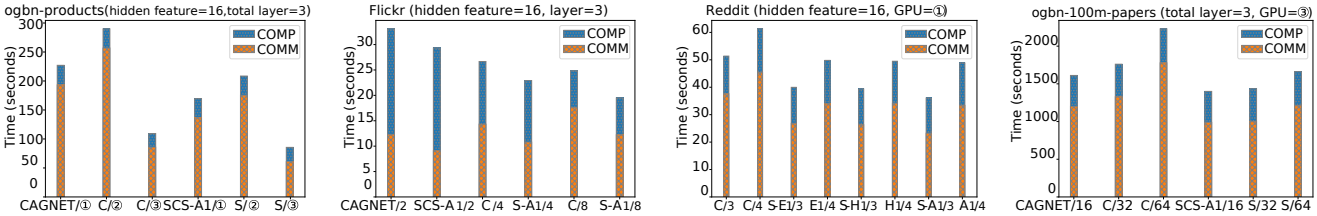


Figure 4: Communication-avoiding performance using all 8 GPUs. In each subplot, x-axis denotes the methods compared: CAGNET [2020], SkipG [2021], SCS-E1/E2/E3 with $\epsilon_E = \{1, 2, 3\}$, SCS-A1/A2/A3 with $\epsilon_A = \{1, 2, 3\}$, SCS-H1/H2/H3 with $\epsilon_H = \{0.01, 0.02, 0.03\}$; y-xis denotes time proportion during training, blue bar denotes computation cost, and orange bar denotes communication cost.



(a) GPU Configurations. (b) Number of GPUs. (c) Number of Total GNN Layers. (d) Size of Hidden Feature.

Figure 5: The performance with system variants compared to CAGNET. In each subplot, y-axis denotes the time proportion during training, where blue bar denotes model computation cost and orange bar denotes communication cost. For the x-axis, we denote the method/GPU configurations in Fig. 5a, method/number of GPUs in Fig. 5b, method/layer number in Fig. 5c, and method/hidden feature size in Fig. 5d.

by PCIe 3.0 $\times$ 16; ② two servers connected by 10Gbps Ethernet - each has four RTX 2080 Ti via PCIe 3.0; ③ four A100 40GB via NVLink; ④ four V100 32GB via NVLink.

First, we demonstrate the effectiveness on communication reduction of SANCUS on different benchmark datasets. In Figure 4, we show SANCUS results with accuracy loss within 0.01. Compared to the SOTA CAGNET and bounded-gradient method SkipG, all our system variants further avoid communication by at least 35% to 74% with bounded embedding staleness. SANCUS can preserve the GNN performance on all datasets. Considering the better performance preserving, the more adaptive SCS-A/H are in fact more robust.

Also, we demonstrate the generality of SANCUS with its worst-behaved baselines to study the influence of different system configurations. Figure 5 shows SANCUS achieves consistent communication avoiding in all configurations. As Figure 5b shows, with increasing GPU number, the total cost compared to CAGNET are reduced continually. Though communication cost increases with the GPU number, with SCS-H, we can reduce the communication cost using all 8 GPUs to get close to the communication cost of CAGNET using 2 GPUs, together with 67% reduction on the computation cost. Importantly, the communication proportion we avoid increases with the number of GPU used, which is hard for centralized architectures [Lian *et al.*, 2017] to achieve.

In Section 5, we give an overall throughput comparison of SANCUS (SCS-A1) to five SOTA distributed GNN systems over their commonly-used Reddit dataset. The worst-behaved baseline SCS-A1, in fact, still outperforms all related SOTA systems. SCS-A1 processes the fastest 10.3 epochs per second with an average $1.86\times$ throughput. As compared to Dorylus which aims at low-cost training, we are $68.7\times$ faster and 80% cheaper. We refer interested readers to the original paper [Peng *et al.*, 2022] for more experiments.

System	Config	TP	Reference
SCS-A	V100*4	10.3	—
CAGNET	V100*4	9	Fig 1 [2020]
RoC	P100*4	5	Fig 5 [2020]
Dorylus	Lambda on CPU*2	0.15	Sec 7.2 Table 4 [2021]
PaGraph	1080ti*4	5	Sec 5.2 5.5; Fig 9 [2020]
DGCL	V100*4	~ 7	Fig 8(a) [2021]

Table 1: TP is the throughput (epochs/s) of GCN on Reddit dataset over SOTA distributed GNN systems.

6 Conclusion

We present SANCUS, the first staleness-aware communication avoiding decentralized GNN system that adaptively avoids communication by caching historical embeddings and managing embedding staleness, while preserving model performance. We propose a set of novel bounded embedding staleness metrics. Then, we integrate the historical embedding cache and bounded embedding staleness check into decentralized GNNs to adaptively skip broadcast among GPUs. We present theoretical analysis to bound communication costs and approximation errors and conduct extensive experiments over large-scale benchmark datasets to demonstrate the efficiency and effectiveness of SANCUS, as well as the necessity of the adaptive strategy to manage system staleness.

Acknowledgments

Yingxia Shao's work is supported by the National Natural Science Foundation of China (Nos. U1936104, 62192784) and CCF-Baidu Open Fund. Yanyan Shen is partially supported by Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102). Lei Chen's work is partially supported by National Key Research and Development Program of China Grant No. 2018AAA0101100, the Hong Kong RGC GRF Project 16202218, CRF Project C6030-18G, C1031-18G, C5026-18G, CRF C2004-21GF, AOE Project AoE/E-603/18, RIF Project R6020-19, Theme-based project TRS T41-603/20R, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Microsoft Research Asia Collaborative Research Grant, HKUST-NAVER/LINE AI Lab, Didi-HKUST joint research lab, HKUST-Webank joint research lab grants and HKUST Global Strategic Partnership Fund (2021 SJTU-HKUST).

References

[Abadal *et al.*, 2022] Sergi Abadal, Akshay Jain, Robert Guirado, Jorge López-Alonso, and Eduard Alarcón. Computing graph neural networks: A survey from algorithms to accelerators. *ACM Comput. Surv.*, 54(9):191:1–191:38, 2022.

[Cai *et al.*, 2021] Zhenkun Cai, Xiao Yan, Yidi Wu, Kaihao Ma, James Cheng, and Fan Yu. DGCL: an efficient communication library for distributed GNN training. In *EuroSys '21: Sixteenth European Conference on Computer Systems, Online Event, United Kingdom, April 26-28, 2021*, pages 130–144. ACM, 2021.

[Chen *et al.*, 2018] Jianfei Chen, Jun Zhu, and Le Song. Stochastic training of graph convolutional networks with variance reduction. In *ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 941–949. PMLR, 2018.

[Cipar *et al.*,] James Cipar, Qirong Ho, Jin Kyu Kim, Seunghak Lee, Gregory R. Ganger, Garth Gibson, Kimberly Keeton, and Eric P. Xing. Solving the straggler problem with bounded staleness. In *HotOS XIV, Santa Ana Pueblo, New Mexico, USA, May 13-15, 2013*. USENIX Association.

[Fey *et al.*, 2021] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Jure Leskovec. Gnnautoscale: Scalable and expressive graph neural networks via historical embeddings. In *ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 3294–3304. PMLR, 2021.

[Gandhi and Iyer, 2021] Swapnil Gandhi and Anand Padmanabha Iyer. P3: distributed deep graph learning at scale. In *15th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2021, July 14-16, 2021*, pages 551–568. USENIX Association, 2021.

[Gholami *et al.*, 2018] Amir Gholami, Ariful Azad, Peter H. Jin, Kurt Keutzer, and Aydin Buluç. Integrated model, batch, and domain parallelism in training neural networks. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018*, pages 77–86. ACM, 2018.

[Hu *et al.*,] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[Jia *et al.*, 2020] Zhihao Jia, Sina Lin, Mingyu Gao, Matei Zaharia, and Alex Aiken. Improving the accuracy, scalability, and performance of graph neural networks with roc. In *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020.

[Jiang *et al.*, 2017] Jiawei Jiang, Bin Cui, Ce Zhang, and Lele Yu. Heterogeneity-aware distributed parameter servers. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, pages 463–478. ACM, 2017.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[Li *et al.*,] Hao Li, Asim Kadav, Erik Kruus, and Cristian Ungureanu. MALT: distributed data-parallelism for existing ML applications. In *EuroSys 2015, Bordeaux, France, April 21-24, 2015*. ACM.

[Lian *et al.*, 2017] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5330–5340, 2017.

[Lin *et al.*, 2020] Zhiqi Lin, Cheng Li, Youshan Miao, Yunxin Liu, and Yinlong Xu. Pagraph: Scaling GNN training on large graphs via computation-aware caching. In *SoCC '20: ACM Symposium on Cloud Computing, Virtual Event, USA, October 19-21, 2020*, pages 401–415. ACM, 2020.

[Ma *et al.*, 2019] Lingxiao Ma, Zhi Yang, Youshan Miao, Jilong Xue, Ming Wu, Lidong Zhou, and Yafei Dai. Neugraph: Parallel deep neural network computation on large graphs. In *2019 USENIX Annual Technical Conference, USENIX ATC 2019, Renton, WA, USA, July 10-12, 2019*. USENIX Association, 2019.

[Miao *et al.*, 2021] Xupeng Miao, Xiaonan Nie, Yingxia Shao, Zhi Yang, Jiawei Jiang, Lingxiao Ma, and Bin Cui. Heterogeneity-aware distributed machine learning training via partial reduce. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 2262–2270. ACM, 2021.

[Min *et al.*, 2021] Seungwon Min, Kun Wu, Sitao Huang, Mert Hidayetoglu, Jinjun Xiong, Eiman Ebrahimi, Deming Chen, and Wen-mei W. Hwu. Large graph convolutional network training with gpu-oriented data communication architecture. *Proc. VLDB Endow.*, 14(11):2087–2100, 2021.

[Peng *et al.*, 2022] Jingshu Peng, Zhao Chen, Yingxia Shao, Yanyan Shen, Lei Chen, and Jiannong Cao. SANCUS: staleness-aware communication-avoiding full-graph decentralized training in large-scale graph neural networks. *Proc. VLDB Endow.*, 15(9):1937–1950, 2022.

[Thorpe *et al.*, 2021] John Thorpe, Yifan Qiao, Jonathan Eyolfson, Shen Teng, Guanzhou Hu, Zhihao Jia, Jinliang Wei, Keval Vora, Ravi Netravali, Miryung Kim, and Guoqing Harry Xu. Dorylus: Affordable, scalable, and accurate GNN training with distributed CPU servers and serverless threads. In *OSDI 2021*, pages 495–514. USENIX Association, 2021.

[Tripathy *et al.*, 2020] Alok Tripathy, Katherine A. Yelick, and Aydin Buluç. Reducing communication in graph neural network training. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event, November 9-19, 2020*. IEEE/ACM, 2020.

[Wang *et al.*, 2020] Zhao Wang, Yijin Guan, Guangyu Sun, Dimin Niu, Yuhao Wang, Hongzhong Zheng, and Yinhe Han. Gnn-pim: A processing-in-memory architecture for graph neural networks. In *Conference on Advanced Computer Architecture*, pages 73–86. Springer, 2020.

[Wu *et al.*, 2021] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1):4–24, 2021.

[Xian *et al.*, 2021] Lintao Xian, Bingzhe Li, Jing Liu, Zhongwen Guo, and David H. C. Du. H-PS: A heterogeneous-aware parameter server with distributed neural network training. *IEEE Access*, 9:44049–44058, 2021.

[Zeng *et al.*, 2020] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. Graphsaint: Graph sampling based inductive learning method. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[Zheng *et al.*, 2020] Da Zheng, Chao Ma, Minjie Wang, Jinjing Zhou, Qidong Su, Xiang Song, Quan Gan, Zheng Zhang, and George Karypis. Distdgl: Distributed graph neural network training for billion-scale graphs. pages 36–44, 2020.

[Zhu *et al.*, 2019] Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, and Jingren Zhou. Aligraph: A comprehensive graph neural network platform. *Proc. VLDB Endow.*, 12(12):2094–2105, 2019.