Capsule: an Out-of-Core Training Mechanism for Colossal GNNs [Supplementary Materials]

Analysis on Different Hardware Configurations

We additionally conduct experiments on servers equipped with GPUs that have commonly available memory capacities on the market, as shown in Tables 1 and 2. This indicates that, in space-constrained environments, most systems encounter out-of-memory errors, highlighting Capsule's exceptional viability.

Analysis on GeForce RTX 4080 Server ($\mathcal{M}_{\mathbf{GPU}} = 16GB$, $\mathcal{M}_{\mathbf{main}} = 128GB$). For preprocessing, baselines like MariusGNN and Ginex require a long runtime or run into OOM errors, while Capsule completes the process with smallest time and memory overhead. For example, Capsule can achieve up to a 5.25-fold increase in runtime efficiency (PA) and a 5.46-fold reduction in memory usage (PA), as shown in Table 1.

For training, Capsule achieves minimal runtime cost and memory usage, as shown in Table 1. For large graphs like PA, FR, UK, and WB, other baseline systems either fail to train or require significantly high memory usage (OOM), while Capsule consistently completes training with the lowest memory consumption and shortest runtime. For medium graphs, like RD and PD, Capsule can achieve up to a 28.91-fold increase in runtime efficiency (GCN in PD) and a 4.27-fold reduction in memory usage (GraphSage in PD).

Analysis on GeForce RTX 3060 Server ($\mathcal{M}_{\mathbf{GPU}} = 12GB$, $\mathcal{M}_{\mathbf{main}} = 64GB$). For preprocessing, baselines like MariusGNN and Ginex require a long runtime or run into OOM errors, while Capsule completes the process with smallest time and memory overhead. For example, Capsule can achieve up to a 5.72-fold increase in runtime efficiency (PD) and a 2.35-fold reduction in memory usage (PA), as shown in Table 2.

For training, Capsule achieves minimal runtime cost and memory usage, as shown in Table 2. For large graphs like PA, FR, UK, and WB, other baseline systems either fail to train or require significantly high memory usage (OOM), while Capsule consistently completes training with the lowest memory consumption and shortest runtime. For medium graphs, like RD and PD, Capsule can achieve up to a 8.63-fold increase in runtime efficiency (GCN in PD) and a 3.58-fold reduction in memory usage (GraphSage in PD).

Differences of the Experimental Setting and Hardware Environments

(1) **Experimental Settings Differ**. The performance of MariusGNN and Ginex is sensitive to parameter settings. For example, the number of buckets partitioned in MarriusGNN and the neighbor cache size in Ginex affect training time and storage efficiency. However, these parameter settings are not mentioned in [2] and [3]. In our setup, we use 32 buckets, which is the best parameter observed from our testings, and a neighbor cache size as 6B, as recommended by Ginex [1] authors.

(2) **Hardware Environments Differ**. The disk type in [3] is AWS p3.2xlarge (1.5GB/s), whereas our disk type is SAMSUNG MZ7LM1T9HMJP (520MB/s). The GPU in [2] is A100 (80GB), while we use V100 (16GB).

Table 1: Runtime and Space Cost Performance on GeForce RTX 4080 Server ($\mathcal{M}_{\mathbf{GPU}} = 16GB$, $\mathcal{M}_{\mathbf{main}} = 128GB$, 20 epochs, Prep.: Preprocessing, time(s), $\mathcal{M}_{\mathbf{main}}$(GB))

Stage	Dataset / Method	RD		PD		PA		FR		UK		WB	
		time/sec	memory/GB	time/sec	memory/GB	time/sec	memory/GB	time/sec	memory/GB	time/sec	memory/GB	time/sec	memory/GB
Prep.	MariusGNN	18	7.7	25	8.4	1,046	107	723	118	OOM	OOM	792	71.5
	Ginex	24	7.0	38	8.7	OOM	OOM	547.2	126	OOM	OOM	OOM	OOM
	Capsule	**15.7**	**3.5**	**18.7**	**4.3**	**199.1**	**19.6**	**370**	**26.7**	**315**	**28.4**	**237.6**	**13.9**
GraphSage	DGL	634	9.7	572	12.8	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	PyG	504	8.6	732	9.2	OOM	OOM	1179	107.3	OOM	OOM	1,685	92.1
	MariusGNN	281	10.9	244	11.5	OOM	OOM	OOM	OOM	OOM	OOM	Fail	Fail
	Ginex	694	7.0	730	9.4	OOM	OOM	1,520	126	OOM	OOM	OOM	OOM
	Capsule (DGL)	**68**	**3.1**	**62**	**3.2**	**370**	**9**	**624**	**23.3**	**212**	**7**	**430**	**13**
	Capsule (PyG)	**232**	**2.8**	**392**	**3**	**630**	**9**	**1,308**	**22.4**	**414**	**6**	**392**	**13**
GCN	DGL	714	9.8	688	12.8	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	PyG	486	8.6	754	9.2	OOM	OOM	1,244	107.3	OOM	OOM	1,776	92.1
	MariusGNN	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	Ginex	1,188	7.0	1908	9.4	OOM	OOM	N/A	126	OOM	OOM	OOM	OOM
	Capsule (DGL)	**66**	**3**	**66**	**3.3**	**334**	**9.4**	**1,144**	**23.3**	**232**	**6.8**	**418**	**13.0**
	Capsule (PyG)	**228**	**2.5**	**450**	**3.0**	**758**	**9**	**1,056**	**19.8**	**480**	**6.8**	**410**	**12.2**
GAT	DGL	342	9.8	398	11.6	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	PyG	576	8.4	750	8.9	OOM	OOM	1,658	107.3	OOM	OOM	1,675	92.1
	MariusGNN	666	10.5	354	7.4	OOM	OOM	OOM	OOM	OOM	OOM	6,457	111.2
	Ginex	Fail	Fail	Fail	Fail	OOM	OOM	Fail	126	OOM	OOM	OOM	OOM
	Capsule (DGL)	**122**	**3**	**174**	**3.3**	**584**	**9.0**	**1,186**	**23.3**	**372**	**6.8**	**614**	**12.2**
	Capsule (PyG)	**346**	**2.5**	**598**	**3.0**	**1,742**	**9.0**	**1,602**	**25.2**	**542**	**7.5**	**466**	**12.5**

Table 2: Runtime and Space Cost Performance on GeForce RTX 3060 Server in ($\mathcal{M}_{\mathbf{GPU}} = 12GB$, $\mathcal{M}_{\mathbf{main}} = 64GB$, 20 epochs, Prep.: Preprocessing, time(s), $\mathcal{M}_{\mathbf{main}}$(GB))

Stage	Dataset / Method	RD		PD		PA		FR		UK		WB	
		time/sec	memory/GB	time/sec	memory/GB	time/sec	memory/GB	time/sec	memory/GB	time/sec	memory/GB	time/sec	memory/GB
Prep.	MariusGNN	26	7.7	25.9	8.4	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	Ginex	57	8.7	135	8.7	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	Capsule	**21.4**	**3.7**	**23.6**	**4.3**	**456.6**	**19.8**	**1,027**	**26.6**	**362.9**	**28.6**	**574.4**	**14.5**
GraphSage	DGL	833	9.7	862	8.5	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	PyG	747	7	1,148	8.1	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	MariusGNN	657	10.5	377	11.1	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	Ginex	1,052	8.7	1,015	9	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	Capsule (DGL)	**178**	**4.8**	**150**	**3.5**	**538**	**9.3**	**928**	**24.0**	**344**	**6.4**	**1,060**	**23.1**
	Capsule (PyG)	**558**	**3.0**	**944**	**3.1**	**1,514**	**9.6**	**2,096**	**28.2**	**1,000**	**7.1**	**1,120**	**23.1**
GCN	DGL	841	7.6	866	8.5	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	PyG	730	7	1,232	8.1	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	MariusGNN	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	Ginex	1,170	8.7	1,364	9.1	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	Capsule (DGL)	**174**	**3.3**	**158**	**3.4**	**574**	**9.5**	**938**	**23.6**	**370**	**7.3**	**1,078**	**20.3**
	Capsule (PyG)	**508**	**3.0**	**1,000**	**3.1**	**1,648**	**9.7**	**2,294**	**27.3**	**1,064**	**7.3**	**1,178**	**23.1**
GAT	DGL	435	7.6	438	8.5	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	PyG	612	7	784	8.1	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	MariusGNN	OOM	OOM	1,263	7.4	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	Ginex	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
	Capsule (DGL)	**178**	**3.3**	**214**	**3.4**	**818**	**9.3**	**1,076**	**24.2**	**526**	**7.3**	**1,294**	**23.2**
	Capsule (PyG)	**458**	**3.0**	**782**	**3.2**	**2,798**	**10.5**	**2,598**	**29.1**	**1,178**	**7.2**	**1,428**	**24.6**

REFERENCES

[1] Yeonhong Park, Sunhong Min, and Jae W. Lee. 2022. Ginex: SSD-enabled billion-scale graph neural network training on a single machine via provably optimal in-memory caching. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2626–2639. https://doi.org/10.14778/3551793.3551819

[2] Jie Sun, Mo Sun, Zheng Zhang, Jun Xie, Zuocheng Shi, Zihan Yang, Jie Zhang, Fei Wu, and Zeke Wang. 2023. Helios: An Efficient Out-of-core GNN Training System on Terabyte-scale Graphs with In-memory Performance. *CoRR* abs/2310.00837 (2023). https://doi.org/10.48550/ARXIV.2310.00837 arXiv:2310.00837

[3] Roger Waleffe, Jason Mohoney, Theodoros Rekatsinas, and Shivaram Venkataraman. 2023. MariusGNN: Resource-Efficient Out-of-Core Training of Graph Neural Networks. In *Proceedings of the Eighteenth European Conference on Computer Systems* (Rome, Italy) *(EuroSys '23)*. Association for Computing Machinery, New York, NY, USA, 144–161. https://doi.org/10.1145/3552326.3567501