Evaluation-*centric* Research

"Experiences from a decade of systems research"

RONG CHEN

IPADS, Shanghai Jiao Tong University

March 2022

Joint work with the *faculties and students of IPADS, SJTU*

IPADS
INSTITUTE OF PARALLEL AND DISTRIBUTED SYSTEMS

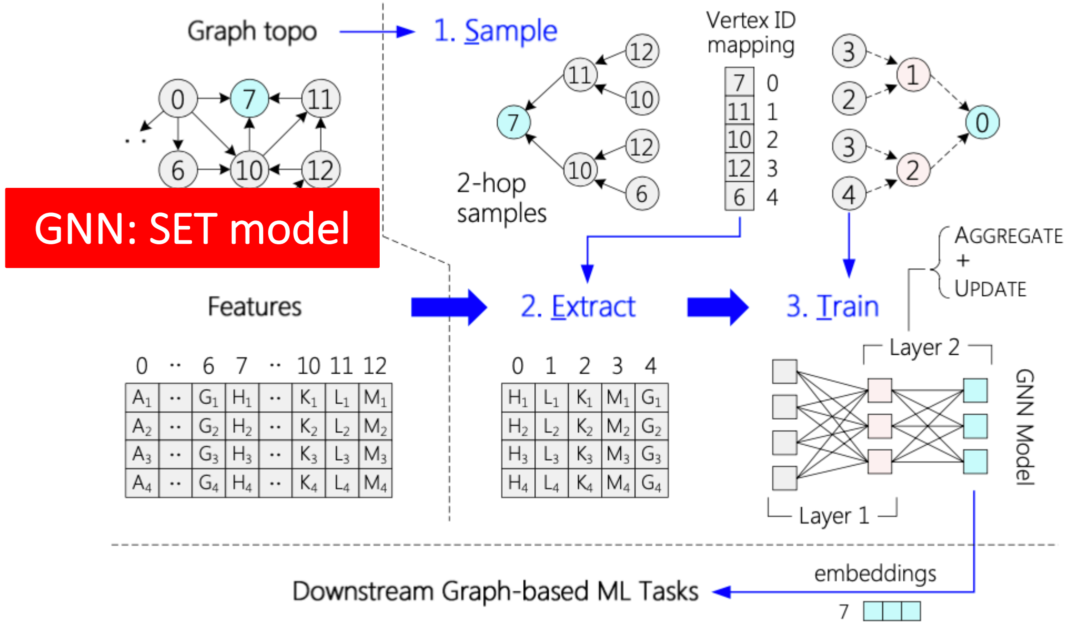
Evaluation-centric Research

1. Motivate your work

Evaluation-centric Research

1. Motivate your work

Case#1: space-sharing for GNN



Evaluation-centric Research

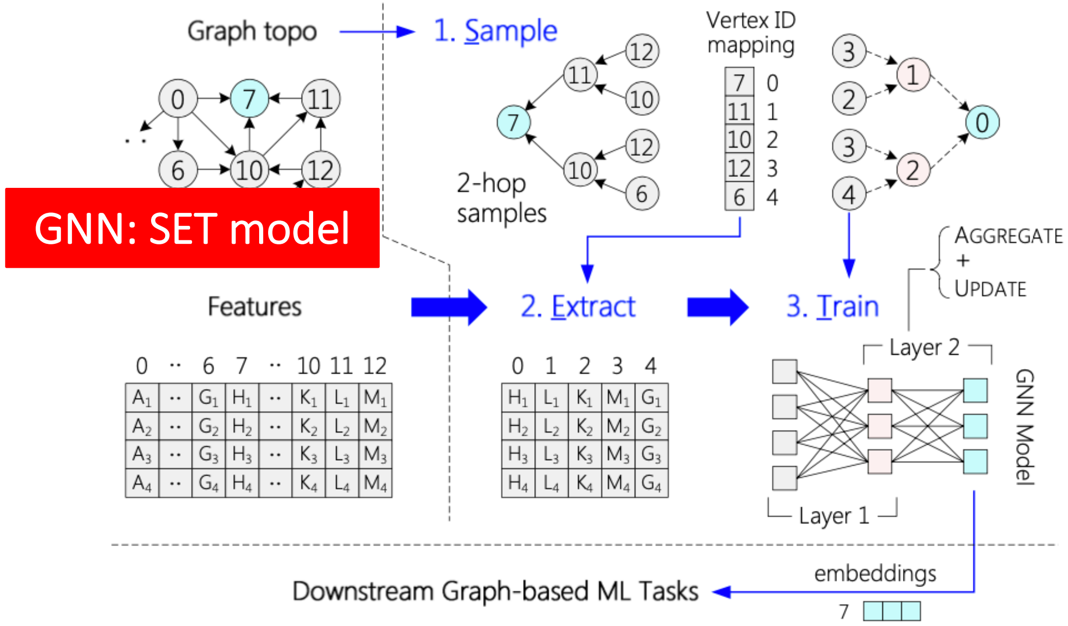
1. Motivate your work

Case#1: space-sharing for GNN

2020.10 survey GPU-accelerated GNN training

2020.11 CPU-based sampling is bottleneck

2021.03 GPU-based sampling, and evaluation



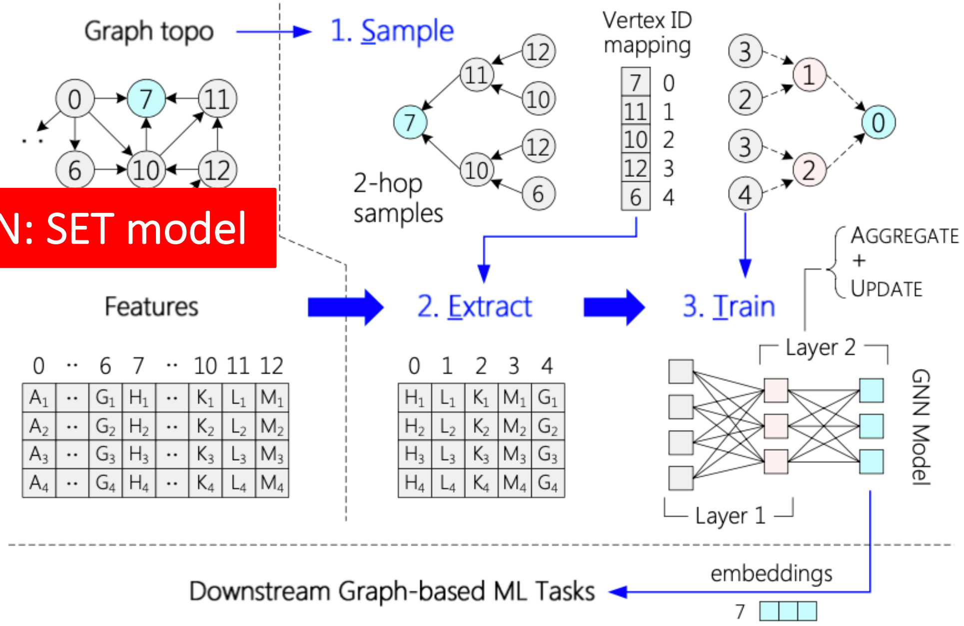
Batch Size	CPU Sampler(未优化) First batch latency	GPU SamplerFirst batch latency
8192	4.67 secs	0.68 secs
16384	8.23 secs	0.88 secs
32768	14.74 secs	1.18 secs
65536	23.20 secs	1.68 secs
131072	39.07 secs	OOM

Evaluation-centric Research

1. Motivate your work

Case#1: space-sharing for GNN

GNN: SET model



2020.10 survey GPU-accelerated GNN training

2020.11 CPU-based sampling is bottleneck

2021.03 GPU-based sampling, and evaluation

2021.05 OPT: pipelining, caching, dynamic workload partition

2021.05 V100 GPU and Friendster dataset

2021.06 GPU memory contention

motivation思考

Status	Complete 🙌
Time	June 4, 2021
Assignment	杨健邦
Property	Empty

Issues

- Memory contention between sampling and training?
  - Topology data - sampling memory
  - Feature data - feature extraction memory

Evaluation-centric Research

1. Motivate your work

Case#1: space-sharing for GNN

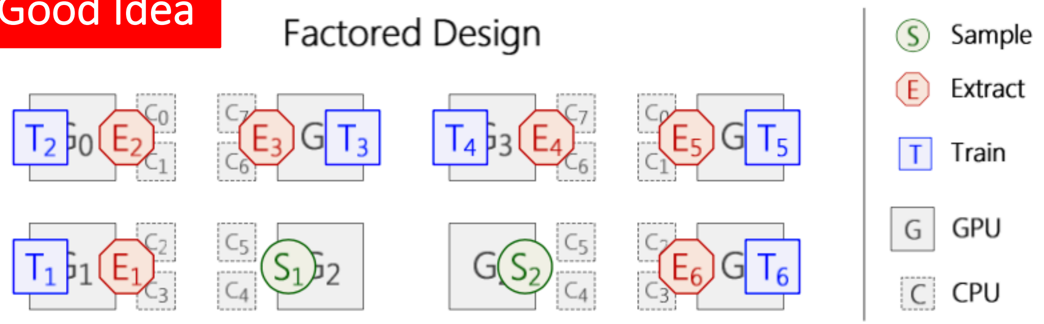
motivation思考

Status	Complete 🙌
Time	June 4, 2021
Assignment	杨健邦
Property	Empty

Issues

- Memory contention between sampling and training?
  - Topology data - sampling memory
  - Feature data - feature extraction memory

Good Idea



Evaluation-centric Research

1. Motivate your work

Case#1: space-sharing for GNN

motivation思考

Status	Complete 🙌
Time	June 4, 2021
Assignment	杨健邦
Property	Empty

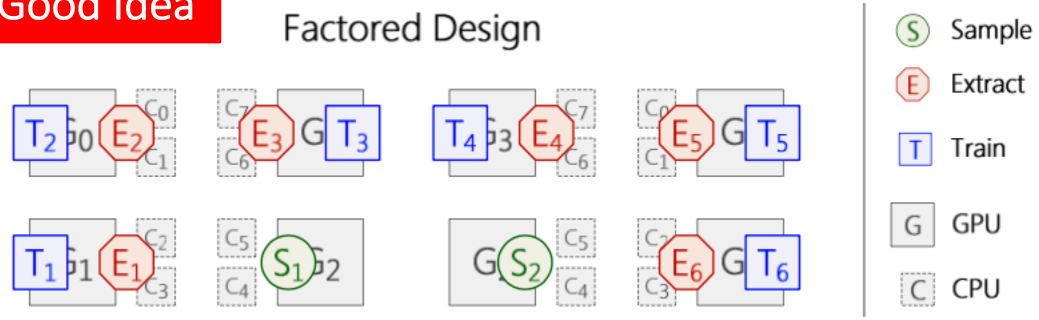
Issues

- Memory contention between sampling and training?
  - Topology data - sampling memory
  - Feature data - feature extraction memory

Rooms and Limits

GNN Systems	Sample	Extract	Train	Total
DGL [1]	4.91	11.32	4.00	20.78
w/ GPU-base Sampling	1.21	10.87	3.97	16.18
T_{SOTA}	2.93	5.55	4.00	12.50
w/ GPU-base Caching [35]	2.88	1.73	4.00	8.62
w/ GPU-base Sampling	0.70	5.46	4.01	10.21
w/ Both	0.70	3.62	4.00	8.37

Good Idea



Evaluation-centric Research

1. Motivate your work

Case#1: space-sharing for GNN

motivation思考

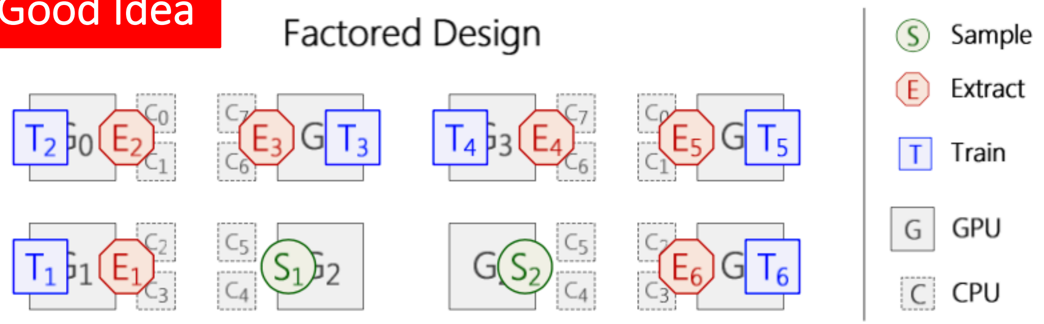
Status	Complete 🙌
Time	June 4, 2021
Assignment	杨健邦
Property	Empty

Issues

- Memory contention between sampling and training?
  - Topology data - sampling memory
  - Feature data - feature extraction memory

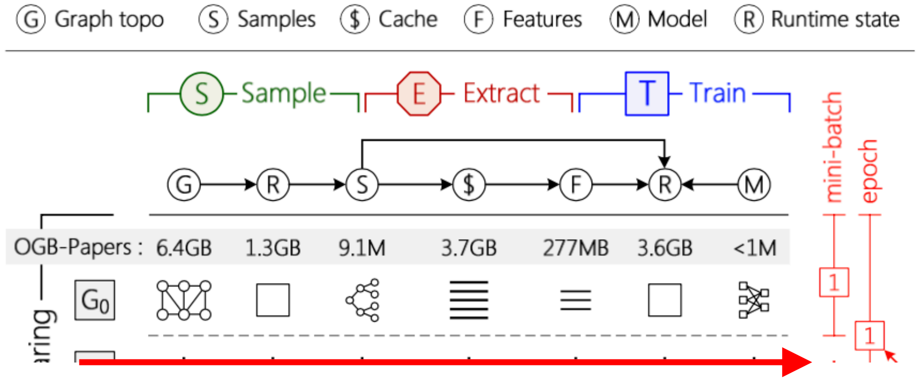
Good Idea

Factored Design



Rooms and Limits

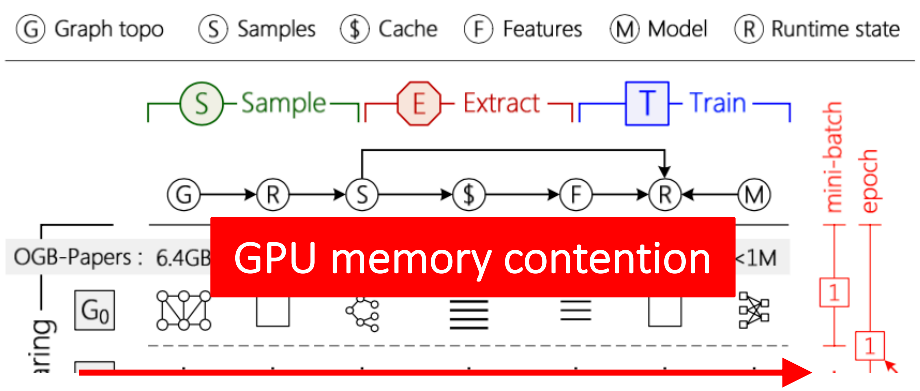
GNN Systems	Sample	Extract	Train	Total
DGL [1]	4.91	11.32	4.00	20.78
w/ GPU-base Sampling	1.21	10.87	3.97	16.18
T_{SOTA}	2.93	5.55	4.00	12.50
w/ GPU-base Caching [35]	2.88	1.73	4.00	8.62
w/ GPU-base Sampling	0.70	5.46	4.01	10.21
w/ Both	0.70	3.62	4.00	8.37

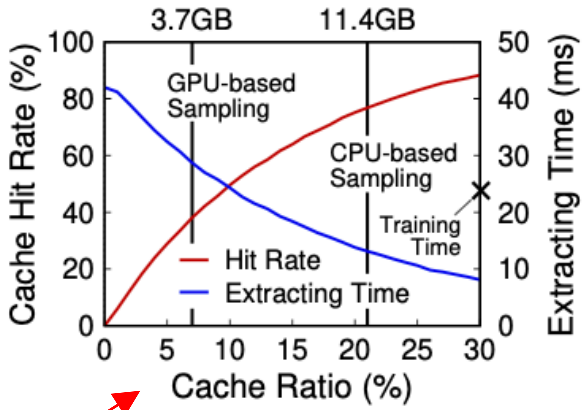


Evaluation-centric Research

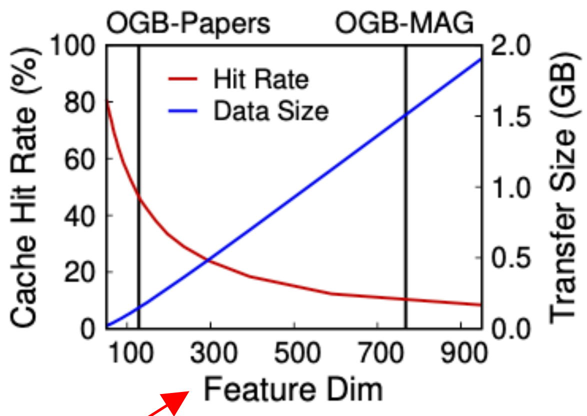
1. Motivate your work

Case#1: space-sharing for GNN





OPT goal



trends

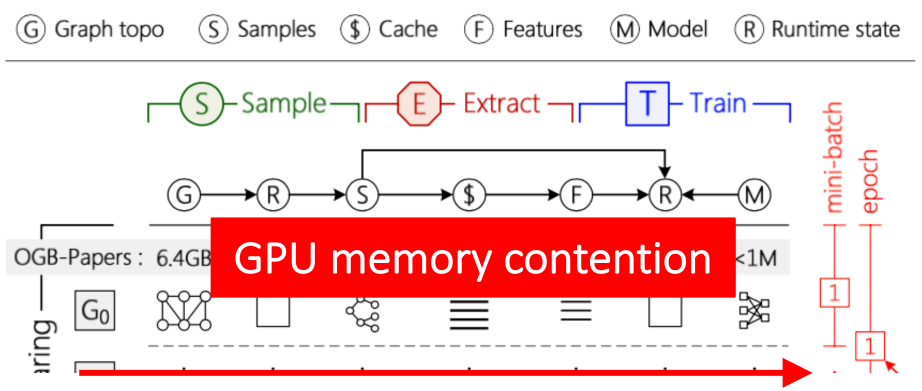
Motivation experiments

- (INSIGHT) Impact factors: cache-ratio, data size

Evaluation-centric Research

1. Motivate your work

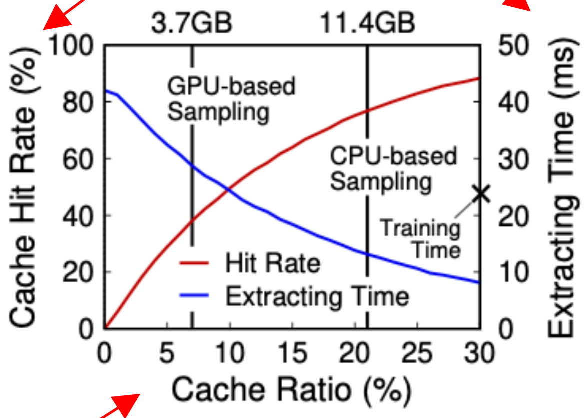
Case#1: space-sharing for GNN



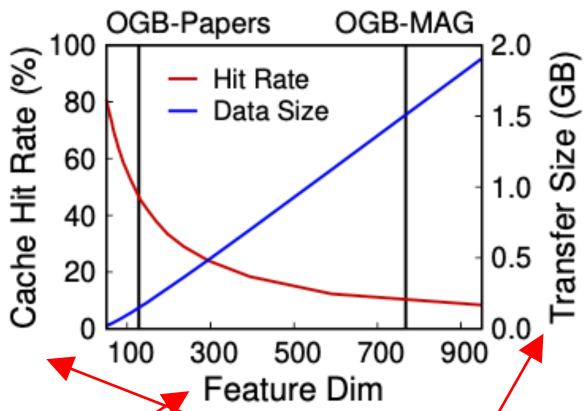
Motivation experiments

- (INSIGHT) Impact factors: cache-ratio, data size
- Key perf. metrics: hit-rate, extracting time

Key metric



OPT goal



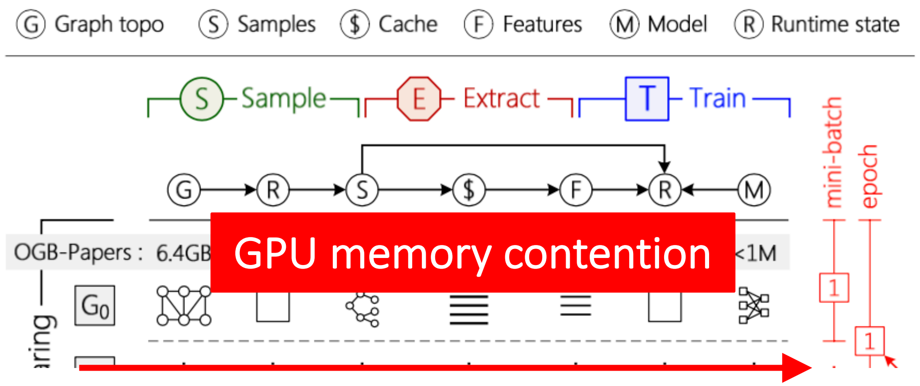
trends

Key metric

Evaluation-centric Research

1. Motivate your work

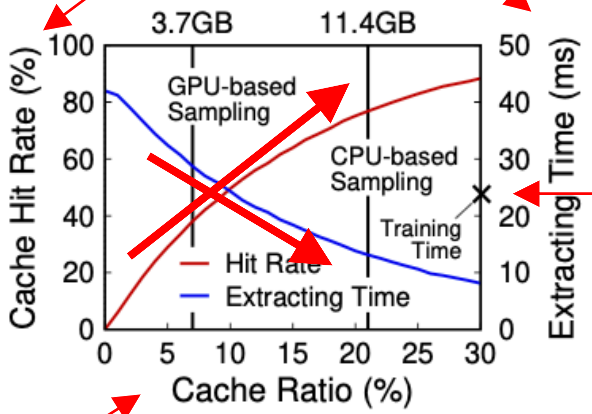
Case#1: space-sharing for GNN



Motivation experiments

- (INSIGHT) Impact factors: cache-ratio, data size
- Key perf. metrics: hit-rate, extracting time
- Focus: line shape, typical cases, reference value

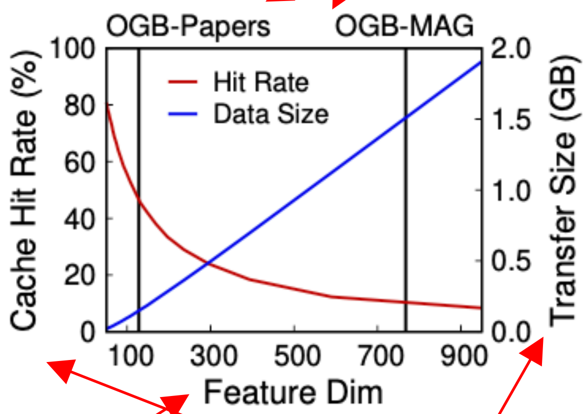
Key metric



Reference value

OPT goal

Typical cases

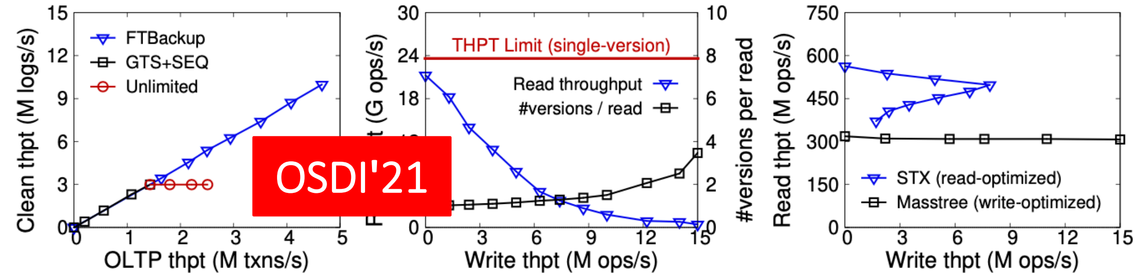


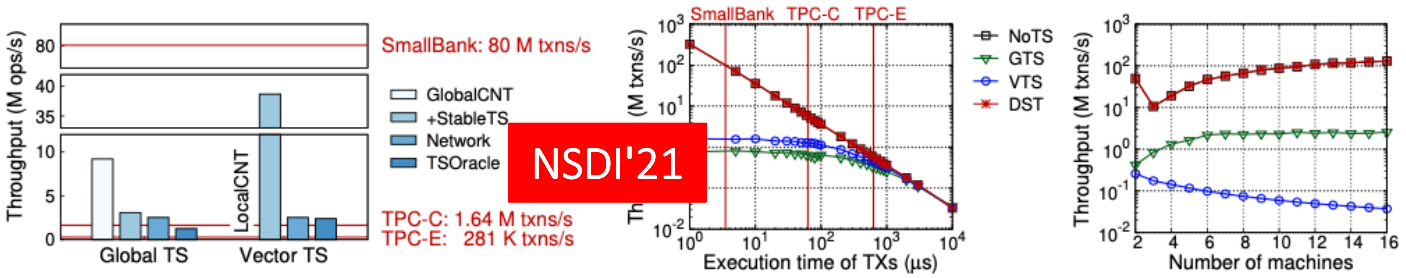
trends

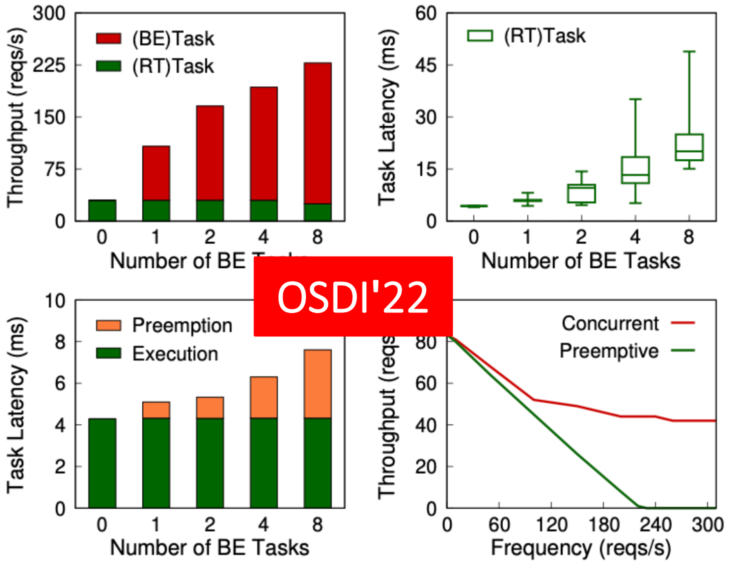
Key metric

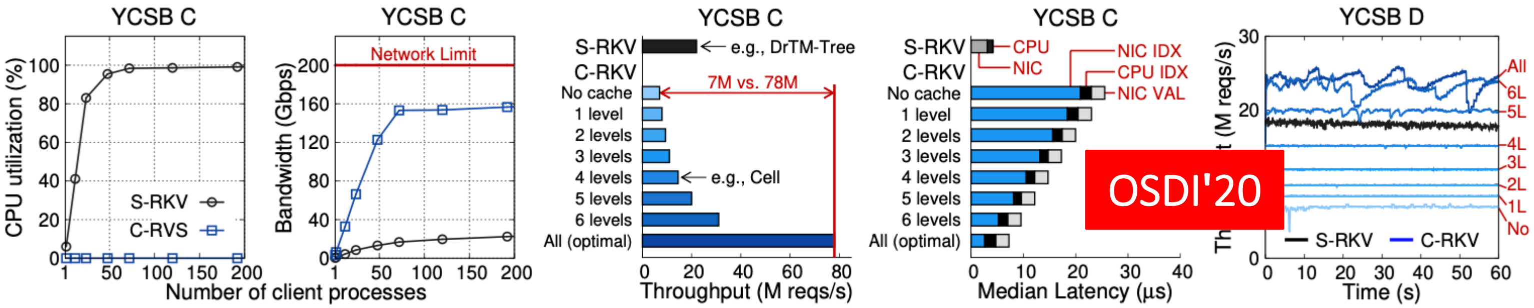
Evaluation-centric Research

1. Motivate your work





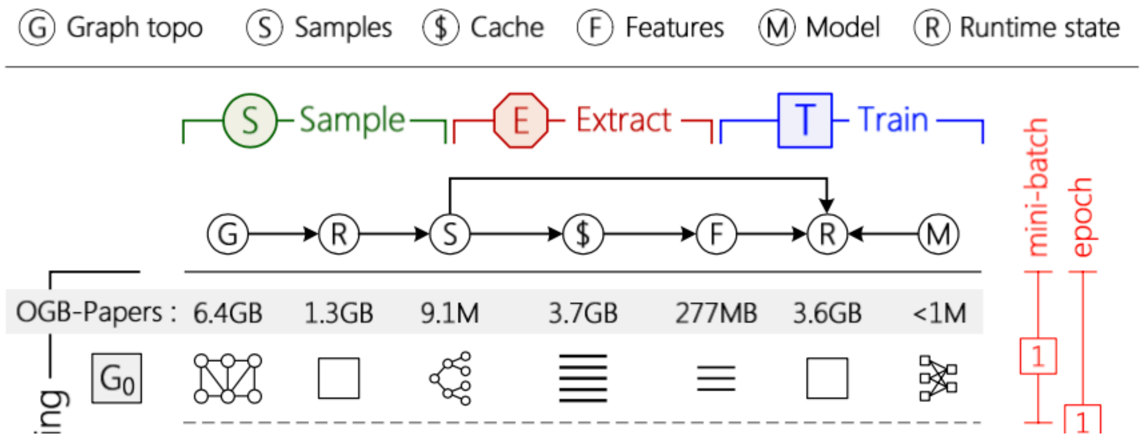




Evaluation-centric Research

1. Motivate your work
2. Support your observation

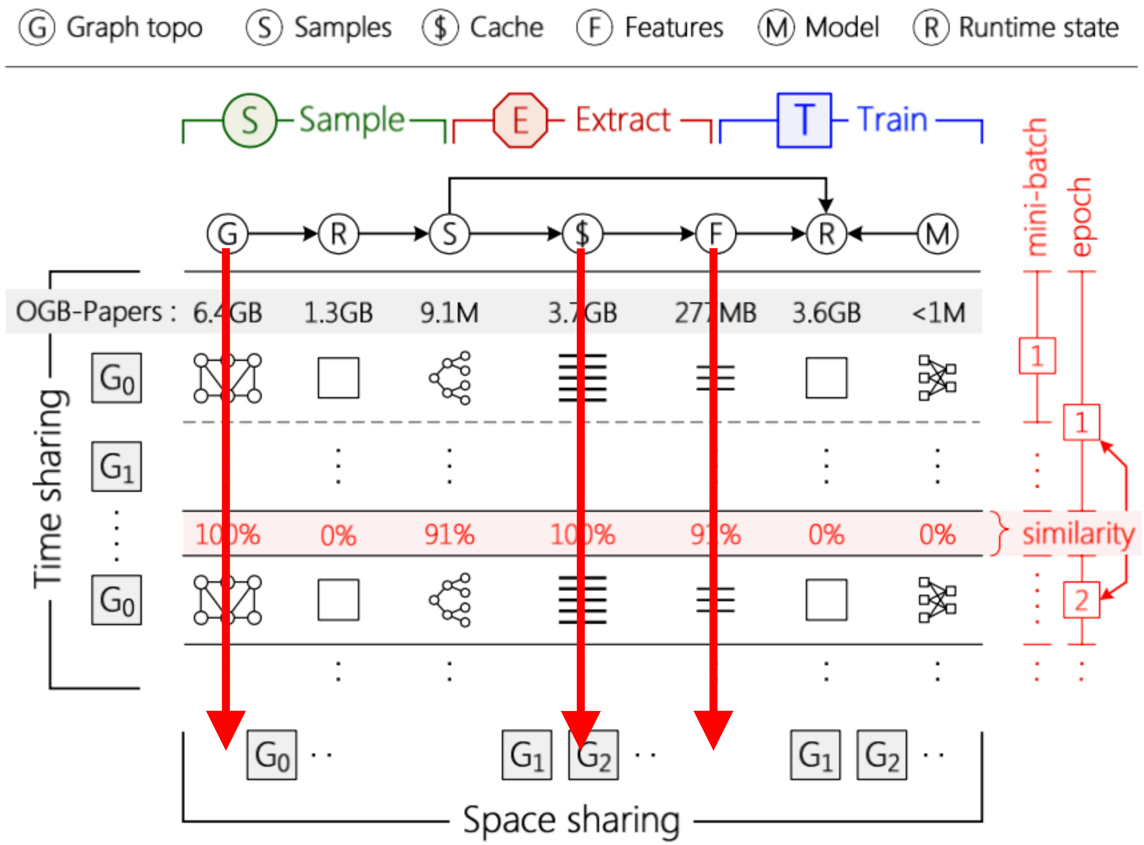
Opportunity: inter-task locality. Our work is motivated by an attractive observation that different training epochs in the same stage share a large amount or even all of the data, which means that sample-based GNN training has extremely good *inter-task* data locality. As shown in Figure 3,



Evaluation-centric Research

1. Motivate your work
2. Support your observation

Opportunity: inter-task locality. Our work is motivated by an attractive observation that different training epochs in the same stage share a large amount or even all of the data, which means that sample-based GNN training has extremely good *inter-task* data locality. As shown in Figure 3,



Evaluation-centric Research

1. Motivate your work
2. Support your observation

Opportunity: inter-task locality. Our work is motivated by an attractive observation that different training epochs in the same stage share a large amount or even all of the data, which means that sample-based GNN training has extremely good *inter-task* data locality. As shown in Figure 3,

A Pre-sampling Based Caching Policy

i and *j*. As shown in Table 2, for the top-ranked vertices, on average over 75% of the access footprint overlaps between two iterations. This indicates that it is feasible to pre-sample a few rounds to estimate vertex hotness.

Table 2. The similarity (in percentage) of access footprint between two epochs for various datasets and sampling algorithms.

Sampling algorithms	**PR** [5]	**TW** [34]	**PA** [4]	**UK** [9]
3-hop random	73.97	78.89	91.29	77.46
Random walks	78.16	72.68	87.14	64.40
3-hop weighted	77.69	66.64	89.57	72.96

Evaluation-centric Research

1. Motivate your work
2. Support your observation

Opportunity: inter-task locality. Our work is motivated by an attractive observation that different training epochs in the same stage share a large amount or even all of the data, which means that sample-based GNN training has extremely good *inter-task* data locality. As shown in Figure 3,

A Pre-sampling Based Caching Policy

i and *j*. As shown in Table 2, for the top-ranked vertices, on average over 75% of the access footprint overlaps between two iterations. This indicates that it is feasible to pre-sample a few rounds to estimate vertex hotness.

<u>Observation experiments</u>

- Metric: definition, setup
- Scope of application: algos, datasets, workloads
- Effectiveness: rare case? bound?

Diff. Datasets

Diff. Algorithms

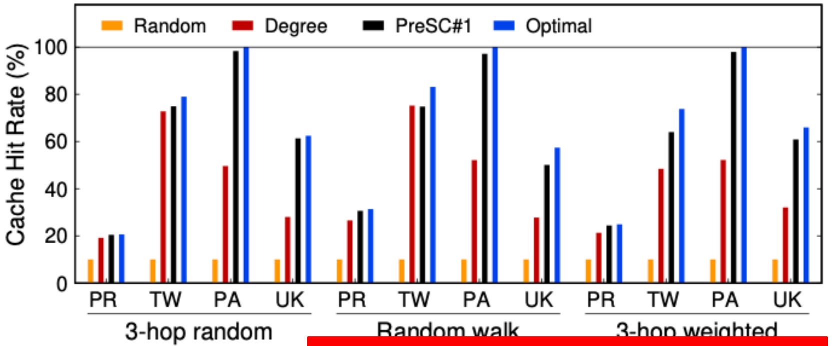
Table 2. The similarity (in percentage) of access footprint between two epochs for various datasets and sampling algorithms.

Sampling algorithms	PR [5]	TW [34]	PA [4]	UK [9]
3-hop random	73.97	78.89	91.29	77.46
Random walks	78.16	72.68	87.14	64.40
3-hop weighted	77.69	66.64	89.57	72.96

Evaluation-centric Research

1. Motivate your work

2. Support your observation



confirmed by evaluation

<u>Observation experiments</u>

- Metric: definition, setup
- Scope of application: algos, datasets, workloads
- Effectiveness: rare case? bound?

Opportunity: inter-task locality. Our work is motivated by an attractive observation that different training epochs in the same stage share a large amount or even all of the data, which means that sample-based GNN training has extremely good *inter-task* data locality. As shown in Figure 3,

A Pre-sampling Based Caching Policy

i and *j*. As shown in Table 2, for the top-ranked vertices, on average over 75% of the access footprint overlaps between two iterations. This indicates that it is feasible to pre-sample a few rounds to estimate vertex hotness.

Diff. Algorithms

Diff. Datasets

Table 2. The similarity (in percentage) of access footprint between two epochs for various datasets and sampling algorithms.

Sampling algorithms	**PR** [5]	**TW** [34]	**PA** [4]	**UK** [9]
3-hop random	73.97	78.89	91.29	77.46
Random walks	78.16	72.68	87.14	64.40
3-hop weighted	77.69	66.64	89.57	72.96

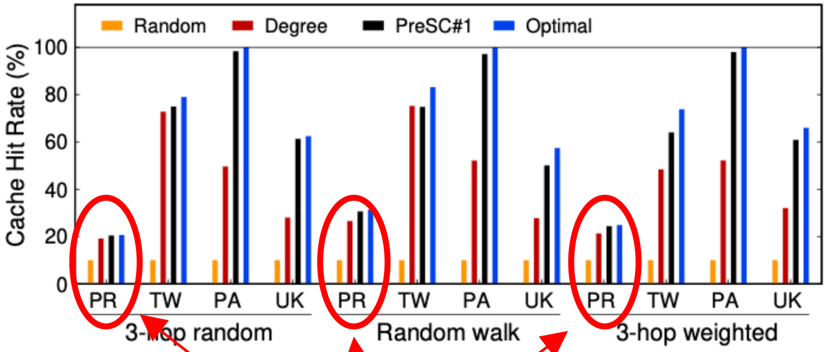
Evaluation-centric Research

1. Motivate your work
2. Support your observation

Opportunity: inter-task locality. Our work is motivated by an attractive observation that different training epochs in the same stage share a large amount or even all of the data, which means that sample-based GNN training has extremely good *inter-task* data locality. As shown in Figure 3,

A Pre-sampling Based Caching Policy

i and *j*. As shown in Table 2, for the top-ranked vertices, on average over 75% of the access footprint overlaps between two iterations. This indicates that it is feasible to pre-sample a few rounds to estimate vertex hotness.



NEW OBSERVATION

Table 2. The similarity (in percentage) of access footprint between two epochs for various datasets and sampling algorithms.

Sampling algorithms	**PR** [5]	**TW** [34]	**PA** [4]	**UK** [9]
3-hop random	73.97	78.89	91.29	77.46
Random walks	78.16	72.68	87.14	64.40
3-hop weighted	77.69	66.64	89.57	72.96

Evaluation-centric Research

1. Motivate your work
2. Support your observation

Numerous kernels. Unlike traditional GPU applications that only contain a few kernels (e.g., at most 14 kernels in Rodinia [10]), it is common to see hundreds of kernels in modern DNN models (see Table 1). In response, large amounts of

Table 1: *The amount of GPU kernels in DNNs evaluated in §7 and the execution time (in millisecond). The codes are generated by TVM [14] and run on AMD Radeon Instinct MI50 G*

OSDI'22

Model	ResNet	DenseNet	VGG	Inception	Bert
#Kernels	307	207	55	146	205
Exec. Time	13.6	3.5	4.4	8.3	5.4

Table 2. *The similarity of vertices with types in different datasets.* #P, #T, *and* $\#V_T$ *denote the number of predicates, types, and vertices with at least one type.* Similarity *denotes the percentage of vertices with a similar combination of predicates as other vertices of its type. Note that we consider a different combination of types as a new type.*

Dataset	#P	#T	$\#V_T$	Similarity
LUBM-2560	17	14	52,272,182	96.29%
WSDTS	86	39	10,234,195	72.28%
DBPSB	14,128	54,736	707,641	74.95%

SOCC'21

tion of vertices of an RDF graph into different groups. We observe that *vertices with the same type commonly have a similar combination of predicates*. For example, in Figure 2, all institutes (INS) has two predicates: so and ty_{INS}. Table 2 shows the percentage of vertices with a similar combination of predicates as other vertices of its type for three synthetic and real-life datasets [3, 5, 7]. Therefore, we argue that the

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation

Performance breakdown

The Earlier The Better

- Confirm-results vs. Find-issues
- Expectation → "Spot The Differences"

CASE: DGL vs. FGNN vs. ?

- Stage-by-stage breakdown
- Speedup? and Overhead?

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation

Performance breakdown

The Earlier The Better

- Confirm-results vs. Find-issues
- Expectation → "Spot The Differences"

CASE: DGL vs. FGNN vs. ?

- Stage-by-stage breakdown
- Speedup? and Overhead?

Model	Dataset	Sampling	Extracting	Training	Sample	Extract	Extract+C	Convert	Train
GCN	Reddit	1.39	2.16	0.78	0.59	1.43	0.08	0.10	0.74
	Products	1.96	2.59	1.19	0.51	1.59	0.17	0.15	1.22
	Papers	10.10	10.33	5.17	1.73	5.99	0.80	0.74	4.29
	Friendster	X.XX	X.XX	X.XX	3.39	37.60	9.02	0.64	6.91
GraphSAGE	Reddit	1.39	2.14	1.07	0.60	1.42	0.08	0.09	0.94
	Products	1.96	2.57	1.29	0.51	1.60	0.17	0.15	1.23
	Papers	10.20	10.15	5.11	1.73	6.53	0.80	0.68	4.17
	Friendster	X.XX	X.XX	X.XX	3.40	37.89	9.06	0.58	6.64
PinSAGE	Reddit	X.XX	X.XX	X.XX	0.10	1.22	0.07	0.10	1.19
	Products	X.XX	X.XX	X.XX	0.21	0.89	0.10	0.13	1.94
	Papers	X.XX	X.XX	X.XX	0.70	2.88	0.47	0.67	6.78
	Friendster	X.XX	X.XX	X.XX	1.35	13.83	3.37	0.53	12.25

GNN	Data set	DGL			PyG			FGNN		
		Sample	**Extract**	**Train**	**Sample**	**Extract**	**Train**	**Sample** = S + M + C	**Extract** (Ratio, Hit%)	**Train**
GCN	**PR**	0.35	2.81	1.22	7.15	3.19	2.14	0.39 = 0.29 + 0.01 + 0.09	0.19 (100%,100%)	1.18
	TW	0.74	9.44	1.48	6.25	9.52	2.51	0.37 = 0.26 + 0.03 + 0.08	0.80 (25%, 89%)	1.50
	PA	1.20	10.70	4.00	9.08	10.27	5.91	0.96 = 0.68 + 0.10 + 0.18	0.61 (21%, 99%)	3.81
	UK	OOM	OOM	OOM	7.19	16.69	4.83	0.56 = 0.38 + 0.03 + 0.14	3.08 (14%, 70%)	3.12
GSG	**PR**	0.13	1.92	0.23	3.89	2.06	0.23	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	0.24
	TW	0.38	4.65	0.44	3.38	4.70	0.34	0.16 = 0.11 + 0.01 + 0.04	0.44 (32%, 89%)	0.42
	PA	0.56	6.06	1.25	4.69	6.36	0.88	0.46 = 0.32 + 0.06 + 0.08	0.34 (25%, 99%)	1.12
	UK	OOM	OOM	OOM	4.01	8.45	0.84	0.27 = 0.18 + 0.02 + 0.06	1.44 (18%, 72%)	1.02
PSG	**PR**	0.16	1.56	1.75	×	×	×	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	1.72
	TW	0.23	4.97	2.57	×	×	×	0.28 = 0.22 + 0.02 + 0.05	0.55 (26%, 86%)	2.54
	PA	0.53	5.00	6.14	×	×	×	0.61 = 0.47 + 0.04 + 0.09	0.41 (22%, 97%)	5.97
	UK	OOM	OOM	OOM	×	×	×	0.65 = 0.48 + 0.03 + 0.13	3.39 (13%, 57%)	6.99

GNN	Dataset	DGL			T_{SOTA}			GNNLab		
		S	E	T	S = G + M	E (R%, H%)	T	S = G + M + C	E (R%, H%)	T
GCN	PR	0.35	2.81	1.22	0.30 = 0.29 + 0.01	0.04 (100, 100)	1.18	0.39 = 0.29 + 0.01 + 0.09	0.15 (100, 100)	1.18
	TW	0.74	9.44	1.48	0.29 = 0.26 + 0.03	3.68 (1, 29)	1.53	0.37 = 0.26 + 0.03 + 0.08	0.76 (25, 89)	1.51
	PA	1.20	10.70	4.00	0.79 = 0.70 + 0.10	3.64 (7, 38)	4.00	0.96 = 0.68 + 0.10 + 0.18	0.49 (21, 99)	3.82
	UK	OOM	OOM	OOM	OOM	OOM	OOM	0.56 = 0.39 + 0.03 + 0.14	3.06 (14, 70)	3.09
GSG	PR	0.13	1.92	0.23	0.16 = 0.15 + 0.01	0.03 (100, 100)	0.25	0.20 = 0.15 + 0.01 + 0.04	0.08 (100, 100)	0.24
	TW	0.38	4.65	0.44	0.12 = 0.11 + 0.01	0.62 (15, 77)	0.44	0.16 = 0.11 + 0.01 + 0.03	0.41 (32, 89)	0.43
	PA	0.56	6.06	1.25	0.38 = 0.33 + 0.06	1.42 (11, 56)	1.18	0.46 = 0.31 + 0.06 + 0.08	0.28 (25, 99)	1.15
	UK	OOM	OOM	OOM	0.19 = 0.19 + 0.00	4.49 (0, 0)	1.08	0.26 = 0.18 + 0.02 + 0.06	1.39 (18, 72)	1.01
PSG	PR	0.40	1.64	1.75	0.16 = 0.16 + 0.01	0.03 (100, 100)	1.74	0.20 = 0.15 + 0.01 + 0.04	0.08 (100, 100)	1.72
	TW	0.72	5.22	2.59	0.23 = 0.22 + 0.02	1.12 (4, 60)	2.60	0.28 = 0.21 + 0.02 + 0.05	0.51 (26, 86)	2.52
	PA	1.86	4.85	5.78	0.54 = 0.49 + 0.05	1.68 (6, 37)	6.09	0.61 = 0.47 + 0.04 + 0.09	0.33 (22, 97)	6.01
	UK	OOM	OOM	OOM	OOM	OOM	OOM	0.65 = 0.49 + 0.03 + 0.13	3.37 (13, 57)	7.00

2021 7.13

Submission

2021 10.9

Camera-ready

2022 2.21

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation

Performance breakdown

The Earlier The Better

- Confirm-results vs. Find-issues
- Expectation → "Spot The Differences"

CASE: DGL vs. FGNN vs. ?

- Stage-by-stage breakdown
- Speedup? and Overhead?

2021 7.13

Submission

Model	Dataset	Sampling	Extracting	Training	Sample	Extract	Extract+C	Convert	Train
GCN	Reddit	1.39	2.16	0.78	0.59	1.43	0.08	0.10	0.74
	Products	1.96	2.59	1.19	0.51	1.59	0.17	0.15	1.22
	Papers	10.10	10.33	5.17	1.73	5.99	0.80	0.74	4.29
	Friendster	X.XX	X.XX	X.XX	3.39	37.60	9.02	0.64	6.91
GraphSAGE	Reddit	1.39	2.14	1.07	0.60	1.42	0.08	0.09	0.94
	Products	1.96	2.57	1.29	0.51	1.60	0.17	0.15	1.23
	Papers	10.20	10.15	5.11	1.73	6.53	0.80	0.68	4.17
	Friendster	X.XX	X.XX	X.XX	3.40	37.89	9.06	0.58	6.64
PinSAGE	Reddit	X.XX	X.XX	X.XX	0.10	1.22	0.07	0.10	1.19
	Products	X.XX	X.XX	X.XX	0.21	0.89	0.10	0.13	1.94
	Papers	X.XX	X.XX	X.XX	0.70	2.88	0.47	0.67	6.78
	Friendster	X.XX	X.XX	X.XX	1.35	13.83	3.37	0.53	12.25

2021 10.9

Camera-ready

GNN	Data set	DGL			PyG			FGNN		
		Sample	Extract	Train	Sample	Extract	Train	Sample = S + M + C	Extract (Ratio, Hit%)	Train
GCN	PR	0.35	2.81	1.22	7.15	3.19	2.14	0.39 = 0.29 + 0.01 + 0.09	0.19 (100%,100%)	1.18
	TW	0.74	9.44	1.48	6.25	9.52	2.51	0.37 = 0.26 + 0.03 + 0.08	0.80 (25%, 89%)	1.50
	PA	1.20	10.70	4.00	9.08	10.27	5.91	0.96 = 0.68 + 0.10 + 0.18	0.61 (21%, 99%)	3.81
	UK	OOM	OOM	OOM	7.19	16.69	4.83	0.56 = 0.38 + 0.03 + 0.14	3.08 (14%, 70%)	3.12
GSG	PR	0.13	1.92	0.23	3.89	2.06	0.23	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	0.24
	TW	0.38	4.65	0.44	3.38	4.70	0.34	0.16 = 0.11 + 0.01 + 0.04	0.44 (32%, 89%)	0.42
	PA	0.56	6.06	1.25	4.69	6.36	0.88	0.46 = 0.32 + 0.06 + 0.08	0.34 (25%, 99%)	1.12
	UK	OOM	OOM	OOM	4.01	8.45	0.84	0.27 = 0.18 + 0.02 + 0.06	1.44 (18%, 72%)	1.02
PSG	PR	0.16	1.56	1.75	×	×	×	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	1.72
	TW	0.23	4.97	2.57	×	×	×	0.28 = 0.22 + 0.02 + 0.05	0.55 (26%, 86%)	2.54
	PA	0.53	5.00	6.14	×	×	×	0.61 = 0.47 + 0.04 + 0.09	0.41 (22%, 97%)	5.97
	UK	OOM	OOM	OOM	×	×	×	0.65 = 0.48 + 0.03 + 0.13	3.39 (13%, 57%)	6.99

2022 2.21

GNN	Dataset	DGL			T_{SOTA}			GNNLab		
		S	E	T	S = G + M	E (R%, H%)	T	S = G + M + C	E (R%, H%)	T
GCN	PR	0.35	2.81	1.22	0.30 = 0.29 + 0.01	0.04 (100, 100)	1.18	0.39 = 0.29 + 0.01 + 0.09	0.15 (100, 100)	1.18
	TW	0.74	9.44	1.48	0.29 = 0.26 + 0.03	3.68 (1, 29)	1.53	0.37 = 0.26 + 0.03 + 0.08	0.76 (25, 89)	1.51
	PA	1.20	10.70	4.00	0.79 = 0.70 + 0.10	3.64 (7, 38)	4.00	0.96 = 0.68 + 0.10 + 0.18	0.49 (21, 99)	3.82
	UK	OOM	OOM	OOM	OOM	OOM	OOM	0.56 = 0.39 + 0.03 + 0.14	3.06 (14, 70)	3.09
GSG	PR	0.13	1.92	0.23	0.16 = 0.15 + 0.01	0.03 (100, 100)	0.25	0.20 = 0.15 + 0.01 + 0.04	0.08 (100, 100)	0.24
	TW	0.38	4.65	0.44	0.12 = 0.11 + 0.01	0.62 (15, 77)	0.44	0.16 = 0.11 + 0.01 + 0.03	0.41 (32, 89)	0.43
	PA	0.56	6.06	1.25	0.38 = 0.33 + 0.06	1.42 (11, 56)	1.18	0.46 = 0.31 + 0.06 + 0.08	0.28 (25, 99)	1.15
	UK	OOM	OOM	OOM	0.19 = 0.19 + 0.00	4.49 (0, 0)	1.08	0.26 = 0.18 + 0.02 + 0.06	1.39 (18, 72)	1.01
PSG	PR	0.40	1.64	1.75	0.16 = 0.16 + 0.01	0.03 (100, 100)	1.74	0.20 = 0.15 + 0.01 + 0.04	0.08 (100, 100)	1.72
	TW	0.72	5.22	2.59	0.23 = 0.22 + 0.02	1.12 (4, 60)	2.60	0.28 = 0.21 + 0.02 + 0.05	0.51 (26, 86)	2.52
	PA	1.86	4.85	5.78	0.54 = 0.49 + 0.05	1.68 (6, 37)	6.09	0.61 = 0.47 + 0.04 + 0.09	0.33 (22, 97)	6.01
	UK	OOM	OOM	OOM	OOM	OOM	OOM	0.65 = 0.49 + 0.03 + 0.13	3.37 (13, 57)	7.00

2021
7.13

		DGL			FGNN				
Model	Dataset	Sampling	Extracting	Training	Sample	Extract	Extract+C	Convert	Train
GCN	Reddit	1.39	2.16	0.78	0.59	1.43	0.08	0.10	0.74
	Products	1.96	2.59	1.19	0.51	1.59	0.17	0.15	1.22
	Papers	10.10	10.33	5.17	1.73	5.99	0.80	0.74	4.29
	Friendster	X.XX	X.XX	X.XX	3.39	37.60	9.02	0.64	6.91

2021
7.13

Model	Dataset	DGL			FGNN				
		Sampling	Extracting	Training	Sample	Extract	Extract+C	Convert	Train
GCN	Reddit	1.39	2.16	0.78	0.59	1.43	0.08	0.10	0.74
	Products	1.96	2.59	1.19	0.51	1.59	0.17	0.15	1.22
	Papers	10.10	10.33	5.17	1.73	5.99	0.80	0.74	4.29
	Friendster	X.XX	X.XX	X.XX	3.39	37.60	9.02	0.64	6.91

2021
9.18

Model	Dataset	Sample	Extract	Train
GCNgpu	Reddit	0.79	2.17	0.50
	Products	1.12	1.84	0.56
	Papers	6.63	6.22	2.43
	Friendster	FAIL	FAIL	FAIL

- GPU-based sampling
- New version DGL

2021 7.13

		DGL			FGNN				
Model	Dataset	Sampling	Extracting	Training	Sample	Extract	Extract+C	Convert	Train
GCN	Reddit	1.39	2.16	0.78	0.59	1.43	0.08	0.10	0.74
	Products	1.96	2.59	1.19	0.51	1.59	0.17	0.15	1.22
	Papers	10.10	10.33	5.17	1.73	5.99	0.80	0.74	4.29
	Friendster	X.XX	X.XX	X.XX	3.39	37.60	9.02	0.64	6.91

2021 9.18

Model	Dataset	Sample	Extract	Train
GCNgpu	Reddit	0.79	2.17	0.50
	Products	1.12	1.84	0.56
	Papers	6.63	6.22	2.43
	Friendster	FAIL	FAIL	FAIL

- GPU-based sampling
- New version DGL

2021 9.21

Model	Dataset	Sample	Extract	Train
GCNgpu	Products	1.14	1.85	0.57
	Papers	6.45	6.10	2.20
	Twitter	2.49	4.27	0.98
	UK-2006-05	FAIL	FAIL	FAIL

- Use proper datasets
- Refine evaluation

2021 7.13

		DGL			FGNN				
Model	Dataset	Sampling	Extracting	Training	Sample	Extract	Extract+C	Convert	Train
GCN	Reddit	1.39	2.16	0.78	0.59	1.43	0.08	0.10	0.74
	Products	1.96	2.59	1.19	0.51	1.59	0.17	0.15	1.22
	Papers	10.10	10.33	5.17	1.73	5.99	0.80	0.74	4.29
	Friendster	X.XX	X.XX	X.XX	3.39	37.60	9.02	0.64	6.91

2021 9.18

Model	Dataset	Sample	Extract	Train
GCNgpu	Reddit	0.79	2.17	0.50
	Products	1.12	1.84	0.56
	Papers	6.63	6.22	2.43
	Friendster	FAIL	FAIL	FAIL

- GPU-based sampling
- New version DGL

2021 9.21

Model	Dataset	Sample	Extract	Train
GCNgpu	Products	1.14	1.85	0.57
	Papers	6.45	6.10	2.20
	Twitter	2.49	4.27	0.98
	UK-2006-05	FAIL	FAIL	FAIL

- Use proper datasets
- Refine evaluation

2021 9.22

Model	Dataset	Sample	Extract	Train
GCN+gpu	Products	0.68	2.85	1.36
	Papers	3.18	10.78	6.64
	Twitter	1.35	8.56	2.04
	UK-2006-05	FAIL	FAIL	FAIL

- Change sampling algo.
- Correct evaluation

2021 7.13

Model	Dataset	Sampling	Extracting	Training	Sample	Extract	Extract+C	Convert	Train
		DGL					FGNN		
GCN	Reddit	1.39	2.16	0.78	0.59	1.43	0.08	0.10	0.74
	Products	1.96	2.59	1.19	0.51	1.59	0.17	0.15	1.22
	Papers	10.10	10.33	5.17	1.73	5.99	0.80	0.74	4.29
	Friendster	X.XX	X.XX	X.XX	3.39	37.60	9.02	0.64	6.91

2021 9.18

Model	Dataset	Sample	Extract	Train
GCNgpu	Reddit	0.79	2.17	0.50
	Products	1.12	1.84	0.56
	Papers	6.63	6.22	2.43
	Friendster	FAIL	FAIL	FAIL

- GPU-based sampling
- New version DGL

2021 9.21

Model	Dataset	Sample	Extract	Train
GCNgpu	Products	1.14	1.85	0.57
	Papers	6.45	6.10	2.20
	Twitter	2.49	4.27	0.98
	UK-2006-05	FAIL	FAIL	FAIL

- Use proper datasets
- Refine evaluation

2021 9.22

Model	Dataset	Sample	Extract	Train
GCN+gpu	Products	0.68	2.85	1.36
	Papers	3.18	10.78	6.64
	Twitter	1.35	8.56	2.04
	UK-2006-05	FAIL	FAIL	FAIL

- Change sampling algo.
- Correct evaluation

2021 9.26

Model	Dataset	Sample	Extract	Train
GCN+gpu	Products	0.35	2.82	0.74
	Papers	1.19	10.70	2.64
	Twitter	0.74	8.64	1.06
	UK-2006-05	FAIL	FAIL	FAIL

- Remove overhead

2021 7.13

Model	Dataset	Sampling (DGL)	Extracting (DGL)	Training (DGL)	Sample (FGNN)	Extract (FGNN)	Extract+C (FGNN)	Convert (FGNN)	Train (FGNN)
GCN	Reddit	1.39	2.16	0.78	0.59	1.43	0.08	0.10	0.74
	Products	1.96	2.59	1.19	0.51	1.59	0.17	0.15	1.22
	Papers	10.10	10.33	5.17	1.73	5.99	0.80	0.74	4.29
	Friendster	X.XX	X.XX	X.XX	3.39	37.60	9.02	0.64	6.91

2021 9.27

Model	Dataset	Sample	Extract	Train	Cache Pct.	Sample	Extract	Train
GCN+gpu	Products	0.35	2.82	0.74	1.00	1.07	0.25	1.07
	Papers	1.19	10.70	2.64	0.16	1.25	0.78	3.75
	Twitter	0.74	8.64	1.06	0.22	0.75	0.93	1.06(1.00+0.06)
	UK-2006-05	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL

- Change sampling algo.
- Add low-level metrics

2021 7.13

Model	Dataset	Sampling (DGL)	Extracting (DGL)	Training (DGL)	Sample (FGNN)	Extract (FGNN)	Extract+C (FGNN)	Convert (FGNN)	Train (FGNN)
GCN	Reddit	1.39	2.16	0.78	0.59	1.43	0.08	0.10	0.74
	Products	1.96	2.59	1.19	0.51	1.59	0.17	0.15	1.22
	Papers	10.10	10.33	5.17	1.73	5.99	0.80	0.74	4.29
	Friendster	X.XX	X.XX	X.XX	3.39	37.60	9.02	0.64	6.91

2021 9.27

Model	Dataset	Sample	Extract	Train	Cache Pct.	Sample	Extract	Train
GCN+gpu	Products	0.35	2.82	0.74	1.00	1.07	0.25	1.07
	Papers	1.19	10.70	2.64	0.16	1.25	0.78	3.75
	Twitter	0.74	8.64	1.06	0.22	0.75	0.93	1.06(1.00+0.06)
	UK-2006-05	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL

- Change sampling algo.
- Add low-level metrics

2021 9.29

Model	Dataset	Sample	Extract	Train	Cache Pct.	Hit Rate	Sample(S+I+Q)	Extract	Train(T+C)
GCN	Products	0.35	2.82	0.74	1.00	1.00	0.40(0.29+0.03+0.08)	0.24	0.94(0.91+0.03)
	Papers	1.19	10.70	2.64	0.20	0.99	1.00(0.69+0.17+0.14)	0.90	2.85(2.67+0.18)
	Twitter	0.74	8.64	1.06	0.24	0.89	0.39(0.26+0.07+0.06)	1.09	1.09(1.02+0.07)
	UK-2006	FAIL	FAIL	FAIL	0.13	0.67	0.59(0.39+0.08+0.11)	3.96	2.23(2.06+0.17)

- Add low-level metrics
- In-depth breakdown
- Refine design & implementation

2021 7.13

Model	Dataset	DGL Sampling	DGL Extracting	DGL Training	FGNN Sample	FGNN Extract	FGNN Extract+C	FGNN Convert	FGNN Train
GCN	Reddit	1.39	2.16	0.78	0.59	1.43	0.08	0.10	0.74
	Products	1.96	2.59	1.19	0.51	1.59	0.17	0.15	1.22
	Papers	10.10	10.33	5.17	1.73	5.99	0.80	0.74	4.29
	Friendster	X.XX	X.XX	X.XX	3.39	37.60	9.02	0.64	6.91

2021 9.27

Model	Dataset	Sample	Extract	Train	Cache Pct.	Sample	Extract	Train
GCN+gpu	Products	0.35	2.82	0.74	1.00	1.07	0.25	1.07
	Papers	1.19	10.70	2.64	0.16	1.25	0.78	3.75
	Twitter	0.74	8.64	1.06	0.22	0.75	0.93	1.06(1.00+0.06)
	UK-2006-05	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL

- Change sampling algo.
- Add low-level metrics

2021 9.29

Model	Dataset	Sample	Extract	Train	Cache Pct.	Hit Rate	Sample(S+I+Q)	Extract	Train(T+C)
GCN	Products	0.35	2.82	0.74	1.00	1.00	0.40(0.29+0.03+0.08)	0.24	0.94(0.91+0.03)
	Papers	1.19	10.70	2.64	0.20	0.99	1.00(0.69+0.17+0.14)	0.90	2.85(2.67+0.18)
	Twitter	0.74	8.64	1.06	0.24	0.89	0.39(0.26+0.07+0.06)	1.09	1.09(1.02+0.07)
	UK-2006	FAIL	FAIL	FAIL	0.13	0.67	0.59(0.39+0.08+0.11)	3.96	2.23(2.06+0.17)

- Add low-level metrics
- In-depth breakdown
- Refine design & implementation

2021 10.1

Model	Dataset	Sample	Extract	Train	Cache Pct.	Hit Rate	Sample(S+I+Q)	Extract	Train(T+C)
GCN	Products	0.35	2.84	1.22	1.00	1.00	0.40(0.29+0.03+0.08)	0.23	1.18(1.15+0.03)
	Papers	1.20	10.77	3.97	0.20	0.99	1.00(0.69+0.17+0.14)	0.66	3.85(3.67+0.18)
	Twitter	0.74	8.52	1.52	0.24	0.89	0.39(0.26+0.07+0.06)	0.86	1.52(1.46+0.06)
	UK-2006	FAIL	FAIL	FAIL	0.13	0.67	0.59(0.39+0.08+0.11)	3.40	3.04(2.89+0.15)

- Correct eval.

2021 7.13

Model	Dataset	Sampling	Extracting	Training	Sample	Extract	Extract+C	Convert	Train
			DGL				FGNN		
GCN	Reddit	1.39	2.16	0.78	0.59	1.43	0.08	0.10	0.74
	Products	1.96	2.59	1.19	0.51	1.59	0.17	0.15	1.22
	Papers	10.10	10.33	5.17	1.73	5.99	0.80	0.74	4.29
	Friendster	X.XX	X.XX	X.XX	3.39	37.60	9.02	0.64	6.91

2021 10.3

Model	**Datasets**	**DGL**			**FGNN**				
		Sample	**Extract**	**Train**	**Ratio**	**Hit**	**Sample** = S + I + Q	**Extract**	**Train** = T + C
GCN	**PR**	0.35	2.84	1.22	100%	100%	0.40 = 0.29 + 0.03 + 0.08	0.23	1.18 = 1.15 + 0.03
	PA	1.20	10.77	3.97	20%	99%	1.00 = 0.69 + 0.17 + 0.14	0.66	3.85 = 3.67 + 0.18
	TW	0.74	8.52	1.52	24%	89%	0.39 = 0.26 + 0.07 + 0.06	0.86	1.52 = 1.46 + 0.06
	UK	×	×	×	13%	67%	0.59 = 0.39 + 0.08 + 0.11	3.40	3.04 = 2.89 + 0.15

- Refine format

2021 7.13

Model	Dataset	Sampling	DGL Extracting	Training	Sample	Extract	FGNN Extract+C	Convert	Train
GCN	Reddit	1.39	2.16	0.78	0.59	1.43	0.08	0.10	0.74
	Products	1.96	2.59	1.19	0.51	1.59	0.17	0.15	1.22
	Papers	10.10	10.33	5.17	1.73	5.99	0.80	0.74	4.29
	Friendster	X.XX	X.XX	X.XX	3.39	37.60	9.02	0.64	6.91

2021 10.3

Model	Datasets	DGL			FGNN				
		Sample	**Extract**	**Train**	**Ratio**	**Hit**	**Sample** = S + I + Q	**Extract**	**Train** = T + C
GCN	**PR**	0.35	2.84	1.22	100%	100%	0.40 = 0.29 + 0.03 + 0.08	0.23	1.18 = 1.15 + 0.03
	PA	1.20	10.77	3.97	20%	99%	1.00 = 0.69 + 0.17 + 0.14	0.66	3.85 = 3.67 + 0.18
	TW	0.74	8.52	1.52	24%	89%	0.39 = 0.26 + 0.07 + 0.06	0.86	1.52 = 1.46 + 0.06
	UK	×	×	×	13%	67%	0.59 = 0.39 + 0.08 + 0.11	3.40	3.04 = 2.89 + 0.15

- Refine format

2021 10.5

GNN Model	Dataset	DGL			FGNN		
		Sample	**Extract**	**Train**	**Sample** = S + I + Q	**Extract** (Ratio, Hit%)	**Train** = T + C
GCN	**PR**	0.35	2.81	1.22	0.39 = 0.29 + 0.01 + 0.09	0.18 (100%,100%)	1.18 = 1.15 + 0.03
	PA	1.20	10.70	4.00	0.96 = 0.68 + 0.10 + 0.18	0.60 (20%, 99%)	3.85 = 3.68 + 0.17
	TW	0.74	9.44	1.48	0.37 = 0.26 + 0.03 + 0.08	0.84 (24%, 89%)	1.49 = 1.42 + 0.07
	UK	×	×	×	0.55 = 0.38 + 0.03 + 0.14	3.24 (13%, 67%)	3.11 = 2.95 + 0.16

- Refine format
- Refine evaluation

2021 7.13

Model	Dataset	DGL Sampling	DGL Extracting	DGL Training	FGNN Sample	FGNN Extract	FGNN Extract+C	FGNN Convert	FGNN Train
GCN	Reddit	1.39	2.16	0.78	0.59	1.43	0.08	0.10	0.74
	Products	1.96	2.59	1.19	0.51	1.59	0.17	0.15	1.22
	Papers	10.10	10.33	5.17	1.73	5.99	0.80	0.74	4.29
	Friendster	X.XX	X.XX	X.XX	3.39	37.60	9.02	0.64	6.91

2021 10.3

- Refine format

Model	Datasets	DGL			FGNN				
		Sample	Extract	Train	Ratio	Hit	Sample = S + I + Q	Extract	Train = T + C
GCN	PR	0.35	2.84	1.22	100%	100%	0.40 = 0.29 + 0.03 + 0.08	0.23	1.18 = 1.15 + 0.03
	PA	1.20	10.77	3.97	20%	99%	1.00 = 0.69 + 0.17 + 0.14	0.66	3.85 = 3.67 + 0.18
	TW	0.74	8.52	1.52	24%	89%	0.39 = 0.26 + 0.07 + 0.06	0.86	1.52 = 1.46 + 0.06
	UK	×	×	×	13%	67%	0.59 = 0.39 + 0.08 + 0.11	3.40	3.04 = 2.89 + 0.15

2021 10.5

- Refine format
- Refine evaluation

GNN Model	Dataset	DGL			FGNN		
		Sample	Extract	Train	Sample = S + I + Q	Extract (Ratio, Hit%)	Train = T + C
GCN	PR	0.35	2.81	1.22	0.39 = 0.29 + 0.01 + 0.09	0.18 (100%,100%)	1.18 = 1.15 + 0.03
	PA	1.20	10.70	4.00	0.96 = 0.68 + 0.10 + 0.18	0.60 (20%, 99%)	3.85 = 3.68 + 0.17
	TW	0.74	9.44	1.48	0.37 = 0.26 + 0.03 + 0.08	0.84 (24%, 89%)	1.49 = 1.42 + 0.07
	UK	×	×	×	0.55 = 0.38 + 0.03 + 0.14	3.24 (13% 67%)	3.11 = 2.95 + 0.16

2021 10.9

- Add new baseline
- Refine format
- Refine evaluation

GNN	Data set	DGL			PyG			FGNN		
		Sample	Extract	Train	Sample	Extract	Train	Sample = S + M + C	Extract (Ratio, Hit%)	Train
GCN	PR	0.35	2.81	1.22	7.15	3.19	2.14	0.39 = 0.29 + 0.01 + 0.09	0.19 (100%,100%)	1.18
	TW	0.74	9.44	1.48	6.25	9.52	2.51	0.37 = 0.26 + 0.03 + 0.08	0.80 (25%, 89%)	1.50
	PA	1.20	10.70	4.00	9.08	10.27	5.91	0.96 = 0.68 + 0.10 + 0.18	0.61 (21%, 99%)	3.81
	UK	OOM	OOM	OOM	7.19	16.69	4.83	0.56 = 0.38 + 0.03 + 0.14	3.08 (14% 70%)	3.12

2021
10.9

GNN	Data set	DGL			PyG			FGNN		
		Sample	Extract	Train	Sample	Extract	Train	Sample = S + M + C	Extract (Ratio, Hit%)	Train
GCN	PR	0.35	2.81	1.22	7.15	3.19	2.14	0.39 = 0.29 + 0.01 + 0.09	0.19 (100%,100%)	1.18
	TW	0.74	9.44	1.48	6.25	9.52	2.51	0.37 = 0.26 + 0.03 + 0.08	0.80 (25%, 89%)	1.50
	PA	1.20	10.70	4.00	9.08	10.27	5.91	0.96 = 0.68 + 0.10 + 0.18	0.61 (21%, 99%)	3.81
	UK	OOM	OOM	OOM	7.19	16.69	4.83	0.56 = 0.38 + 0.03 + 0.14	3.08 (14%, 70%)	3.12
GSG	PR	0.13	1.92	0.23	3.89	2.06	0.23	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	0.24
	TW	0.38	4.65	0.44	3.38	4.70	0.34	0.16 = 0.11 + 0.01 + 0.04	0.44 (32%, 89%)	0.42
	PA	0.56	6.06	1.25	4.69	6.36	0.88	0.46 = 0.32 + 0.06 + 0.08	0.34 (25%, 99%)	1.12
	UK	OOM	OOM	OOM	4.01	8.45	0.84	0.27 = 0.18 + 0.02 + 0.06	1.44 (18%, 72%)	1.02
PSG	PR	0.16	1.56	1.75	×	×	×	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	1.72
	TW	0.23	4.97	2.57	×	×	×	0.28 = 0.22 + 0.02 + 0.05	0.55 (26%, 86%)	2.54
	PA	0.53	5.00	6.14	×	×	×	0.61 = 0.47 + 0.04 + 0.09	0.41 (22%, 97%)	5.97
	UK	OOM	OOM	OOM	×	×	×	0.65 = 0.48 + 0.03 + 0.13	3.39 (13%, 57%)	6.99

2021 10.9

GNN	Data set	DGL Sample	DGL Extract	DGL Train	PyG Sample	PyG Extract	PyG Train	FGNN Sample = S + M + C	FGNN Extract (Ratio, Hit%)	FGNN Train
GCN	PR	0.35	2.81	1.22	7.15	3.19	2.14	0.39 = 0.29 + 0.01 + 0.09	0.19 (100%,100%)	1.18
	TW	0.74	9.44	1.48	6.25	9.52	2.51	0.37 = 0.26 + 0.03 + 0.08	0.80 (25%, 89%)	1.50
	PA	1.20	10.70	4.00	9.08	10.27	5.91	0.96 = 0.68 + 0.10 + 0.18	0.61 (21%, 99%)	3.81
	UK	OOM	OOM	OOM	7.19	16.69	4.83	0.56 = 0.38 + 0.03 + 0.14	3.08 (14%, 70%)	3.12
GSG	PR	0.13	1.92	0.23	3.89	2.06	0.23	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	0.24
	TW	0.38	4.65	0.44	3.38	4.70	0.34	0.16 = 0.11 + 0.01 + 0.04	0.44 (32%, 89%)	0.42
	PA	0.56	6.06	1.25	4.69	6.36	0.88	0.46 = 0.32 + 0.06 + 0.08	0.34 (25%, 99%)	1.12
	UK	OOM	OOM	OOM	4.01	8.45	0.84	0.27 = 0.18 + 0.02 + 0.06	1.44 (18%, 72%)	1.02
PSG	PR	~~0.16~~	1.56	1.75	×	×	×	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	1.72
	TW	~~0.23~~	4.97	2.57	×	×	×	0.28 = 0.22 + 0.02 + 0.05	0.55 (26%, 86%)	2.54
	PA	~~0.53~~	5.00	6.14	×	×	×	0.61 = 0.47 + 0.04 + 0.09	0.41 (22%, 97%)	5.97
	UK	~~OOM~~	OOM	OOM	×	×	×	0.65 = 0.48 + 0.03 + 0.13	3.39 (13%, 57%)	6.99

Camera-ready

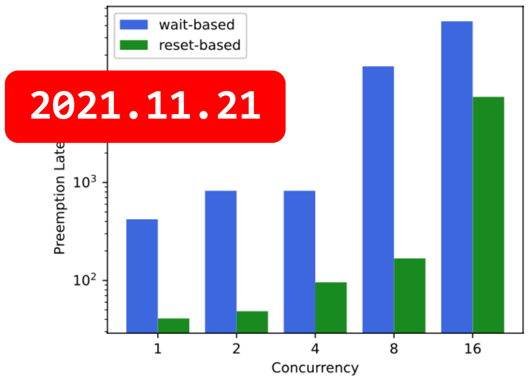
2022 2.21

GNN	Dataset	DGL S	DGL E	DGL T	T_{SOTA} S = G + M	T_{SOTA} E (R%, H%)	T_{SOTA} T	GNNLab S = G + M + C	GNNLab E (R%, H%)	GNNLab T
GCN	PR	0.35	2.81	1.22	0.30 = 0.29 + 0.01	0.04 (100, 100)	1.18	0.39 = 0.29 + 0.01 + 0.09	0.15 (100, 100)	1.18
	TW	0.74	9.44	1.48	0.29 = 0.26 + 0.03	3.68 (1, 29)	1.53	0.37 = 0.26 + 0.03 + 0.08	0.76 (25, 89)	1.51
	PA	1.20	10.70	4.00	0.79 = 0.70 + 0.10	3.64 (7, 38)	4.00	0.96 = 0.68 + 0.10 + 0.18	0.49 (21, 99)	3.82
	UK	OOM	OOM	OOM	OOM	OOM	OOM	0.56 = 0.39 + 0.03 + 0.14	3.06 (14, 70)	3.09
GSG	PR	0.13	1.92	0.23	0.16 = 0.15 + 0.01	0.03 (100, 100)	0.25	0.20 = 0.15 + 0.01 + 0.04	0.08 (100, 100)	0.24
	TW	0.38	4.65	0.44	0.12 = 0.11 + 0.01	0.62 (15, 77)	0.44	0.16 = 0.11 + 0.01 + 0.03	0.41 (32, 89)	0.43
	PA	0.56	6.06	1.25	0.38 = 0.33 + 0.06	1.42 (11, 56)	1.18	0.46 = 0.31 + 0.06 + 0.08	0.28 (25, 99)	1.15
	UK	OOM	OOM	OOM	0.19 = 0.19 + 0.00	4.49 (0, 0)	1.08	0.26 = 0.18 + 0.02 + 0.06	1.39 (18, 72)	1.01
PSG	PR	0.40	1.64	1.75	0.16 = 0.16 + 0.01	0.03 (100, 100)	1.74	0.20 = 0.15 + 0.01 + 0.04	0.08 (100, 100)	1.72
	TW	0.72	5.22	2.59	0.23 = 0.22 + 0.02	1.12 (4, 60)	2.60	0.28 = 0.21 + 0.02 + 0.05	0.51 (26, 86)	2.52
	PA	1.86	4.85	5.78	0.54 = 0.49 + 0.05	1.68 (6, 37)	6.09	0.61 = 0.47 + 0.04 + 0.09	0.33 (22, 97)	6.01
	UK	OOM	OOM	OOM	OOM	OOM	OOM	0.65 = 0.49 + 0.03 + 0.13	3.37 (13, 57)	7.00

- Refine baseline
- Correct impl.
- Confirm merits
- Confirm overhead

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation

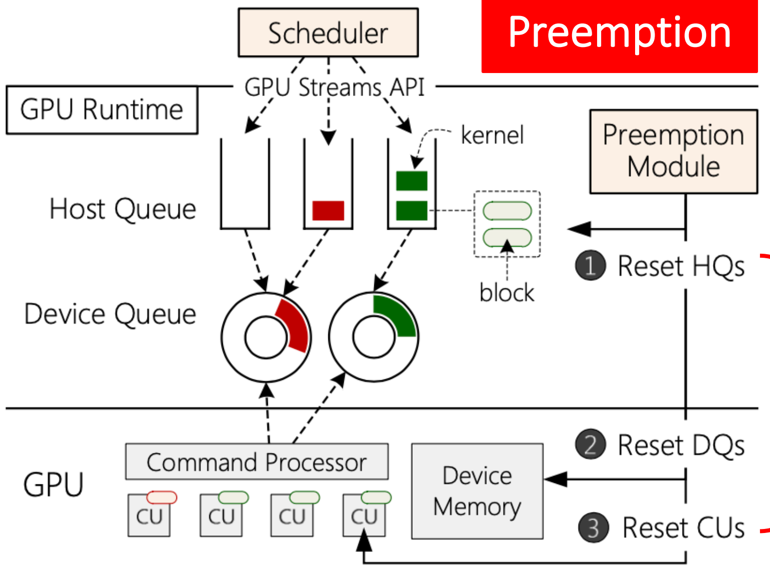


- GPU preemption
- Wait-based vs. Reset-based
- wrt. *#tasks* preempted
- 1-2 order-of-magnitude faster

Evaluation-centric Research

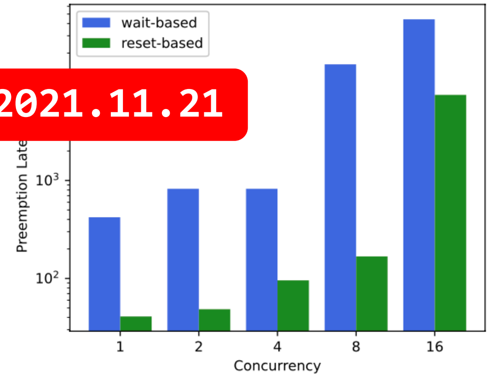
1. Motivate your work
2. Support your observation
3. Revise your implementation

Instant GPU Preemption

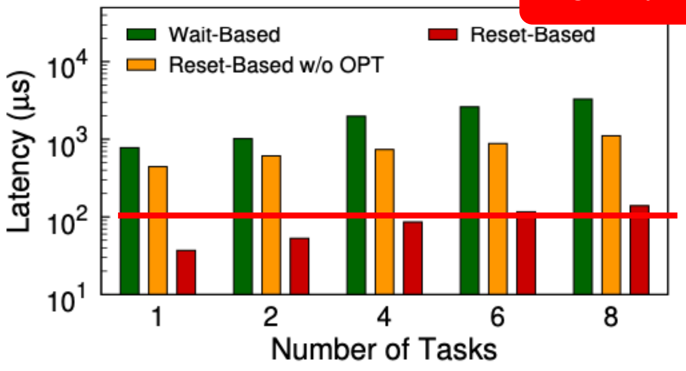


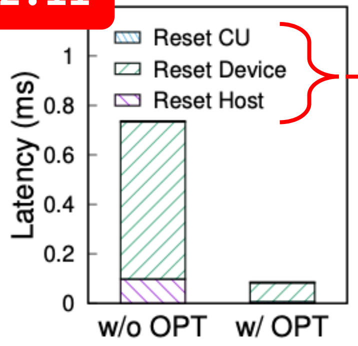
OSDI'22

2021.11.21



2021.12.11

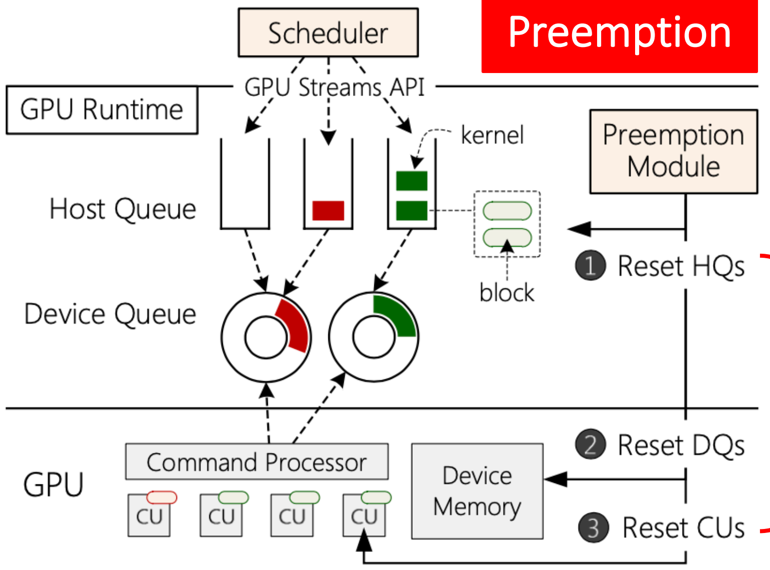




Evaluation-centric Research

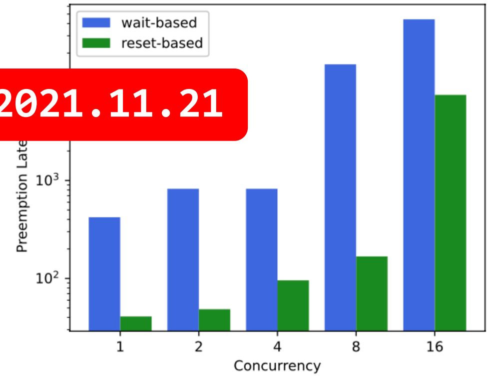
1. Motivate your work
2. Support your observation
3. Revise your implementation

Instant GPU Preemption

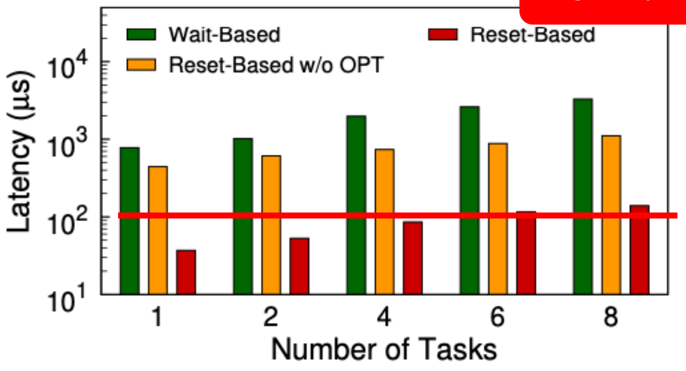


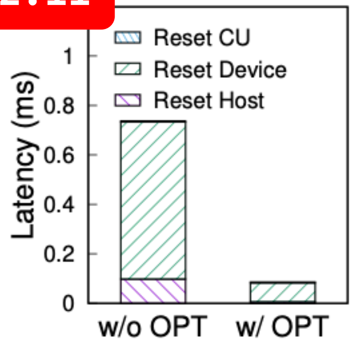
OSDI'22

2021.11.21

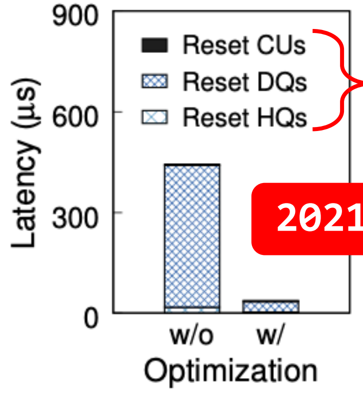


2021.12.11





2021.12.12



Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation

- “First things first”: know your LIMITS
- Enough earnings, close to optimal
- Finding optimal is a clear plus
- Realize advantage/disadvantage

OPTIMAL caching policy

[4]Given a cache ratio, to obtain the optimal cache performance (transferred data size/cache hit rate), all sample footprints are recorded. After training, we calculate the corresponding metric if we cache the most visited vertices.

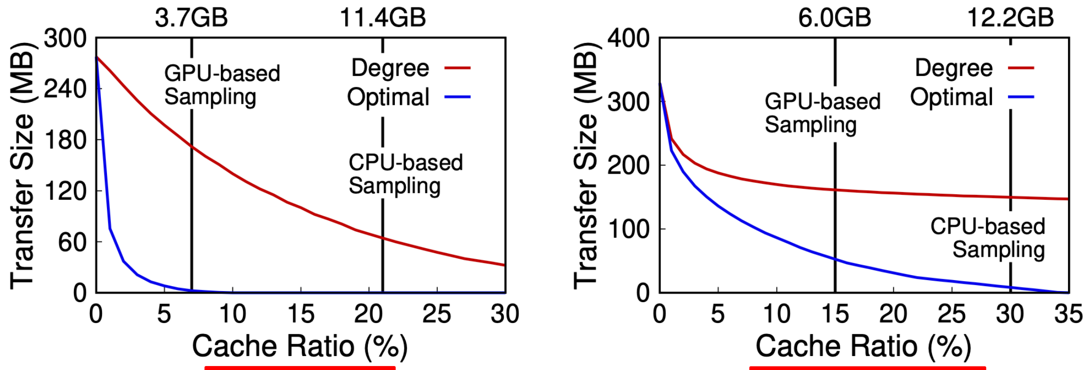


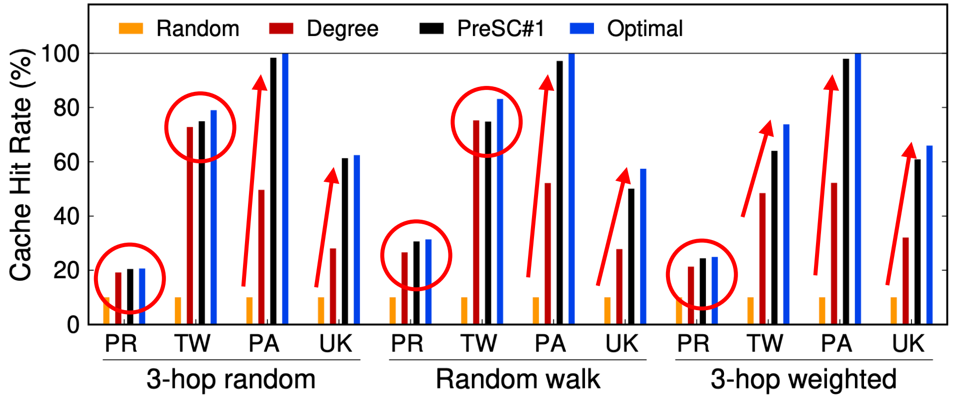
Figure 5. Tl Dataset ansferred data of degr Algorithm optimal caching policies with the increase of cache ratio for (a) OGB-Papers with uniform sampling and (b) Twitter with weighted sampling.

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation

- "First things first": know your LIMITS
- Enough earnings, close to optimal
- Finding optimal is a clear plus
- Realize advantage/disadvantage

addition, our pre-sampling based caching policy achieves 90% – 99% of the optimal cache hit rate in all experiments.



Efficiency

Robustness

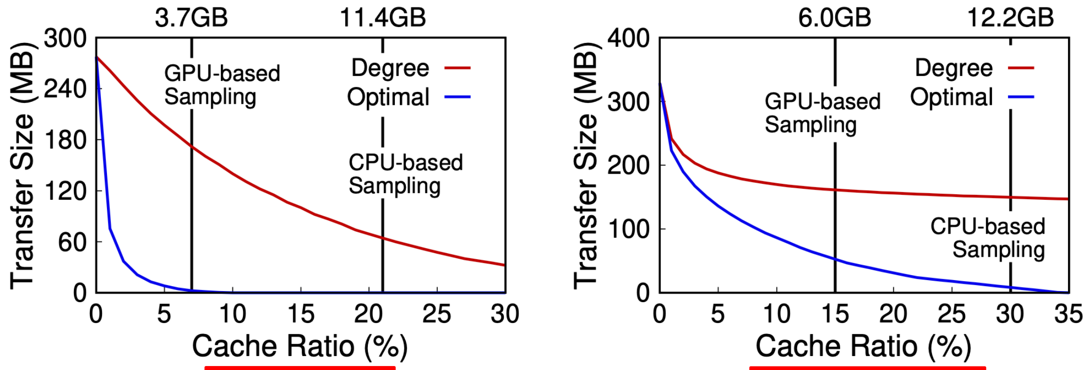


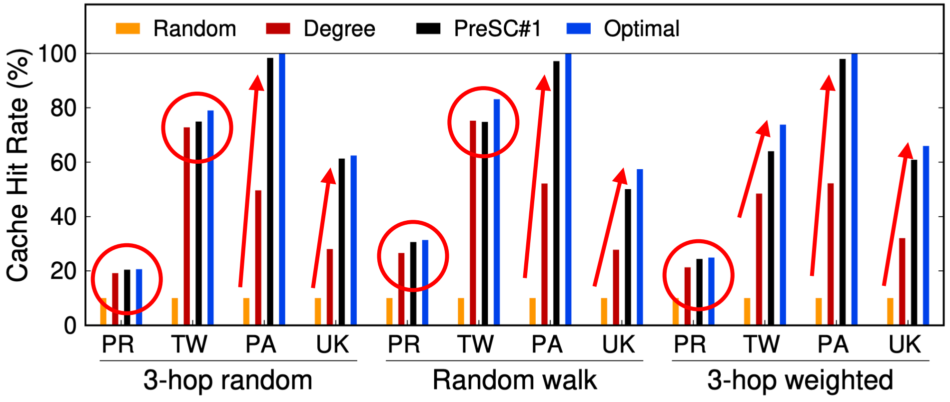
Figure 5. Th Dataset ansferred data of degr Algorithm ptimal caching policies with the increase of cache ratio for (a) OGB-Papers with uniform sampling and (b) Twitter with weighted sampling.

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation

- “First things first”: know your LIMITS
- Enough earnings, close to optimal
- Finding optimal is a clear plus
- Realize advantage/disadvantage

addition, our pre-sampling based caching policy achieves 90% – 99% of the optimal cache hit rate in all experiments.



Efficiency

Robustness

Design in-depth experiments

Amplify advantages

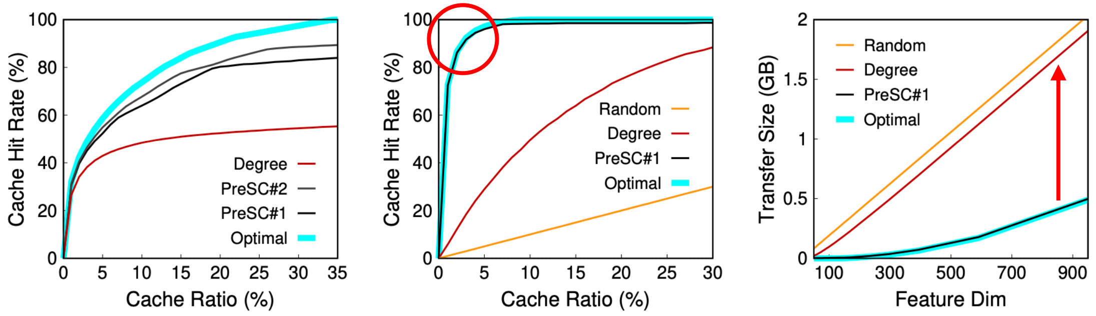


Figure 11. The comparison among different caching policies for (a) Twitter with weighted sampling, (b) OGB-Papers with 3-hop neighborhood, and (c) OGB-Papers with the increase of feature dimensions. PreSC#K conducts K sampling stages.

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
   - “First things first”: know your LIMITS
   - Enough earnings, close to optimal
   - Finding optimal is a clear plus
   - Realize advantage/disadvantage Limitation

PR can be loaded into a single GPU worst case

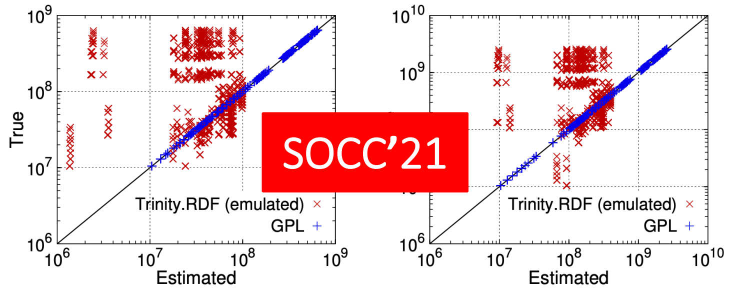
Dataset	#Vertex	#Edge	Dim.	#TS	Vol_G	Vol_F
PR [5]	2.4M	124M	100	197K	481MB	934MB
TW [34]	41.7M	1.5B	256	417K	5.6GB	40GB
PA [4]	111M	1.6B	128	1.2M	6.4GB	53GB
UK [9]	77.7M	3.0B	256	1.0M	11.3GB	74GB

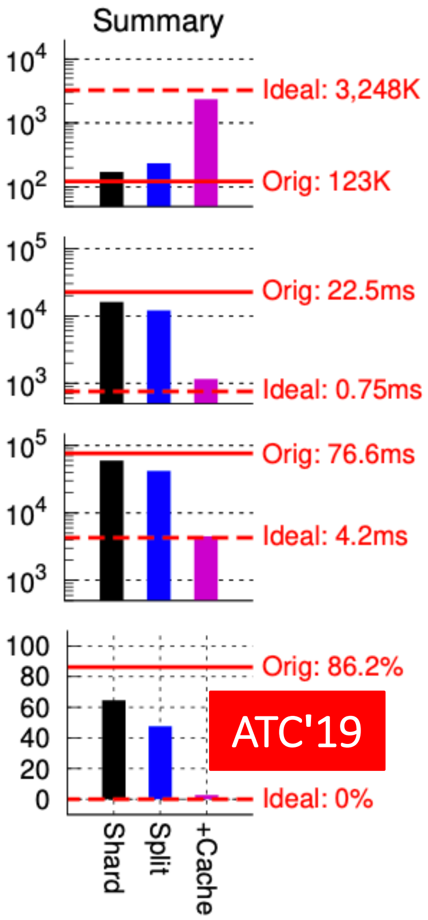
overhead

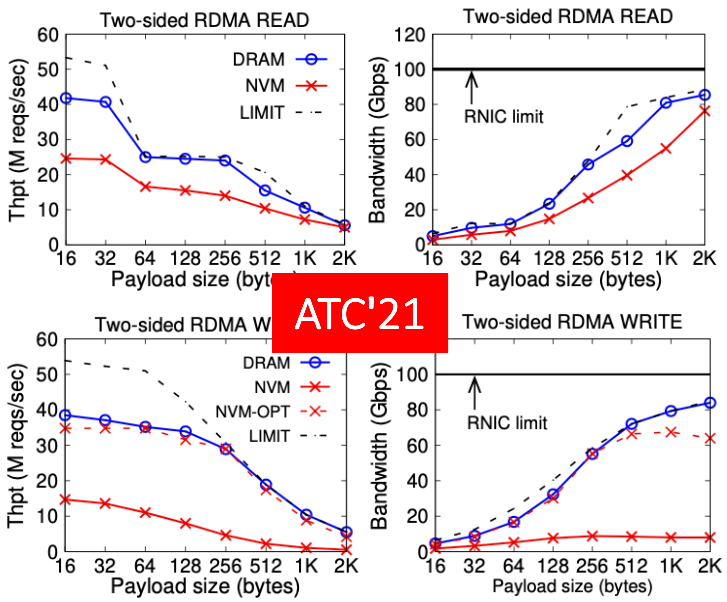
GNN	Dataset	T_{SOTA}			GNNLab		
		S = G + M	E (R%, H%)	T	S = G + M + C	E (R%, H%)	T
GCN	PR	0.30 = 0.29 + 0.01	0.04 (100, 100)	1.18	0.39 = 0.29 + 0.01 + 0.09	0.15 (100, 100)	1.18
	TW	0.29 = 0.26 + 0.03	3.68 (1, 29)	1.53	0.37 = 0.26 + 0.03 + 0.08	0.76 (25, 89)	1.51
	PA	0.79 = 0.70 + 0.10	3.64 (7, 38)	4.00	0.96 = 0.68 + 0.10 + 0.18	0.49 (21, 99)	3.82
	UK	OOM	OOM	OOM	0.56 = 0.39 + 0.03 + 0.14	3.06 (14, 70)	3.09
GSG	PR	0.16 = 0.15 + 0.01	0.03 (100, 100)	0.25	0.20 = 0.15 + 0.01 + 0.04	0.08 (100, 100)	0.24
	TW	0.12 = 0.11 + 0.01	0.62 (15, 77)	0.44	0.16 = 0.11 + 0.01 + 0.03	0.41 (32, 89)	0.43
	PA	0.38 = 0.33 + 0.06	1.42 (11, 56)	1.18	0.46 = 0.31 + 0.06 + 0.08	0.28 (25, 99)	1.15
	UK	0.19 = 0.19 + 0.00	4.49 (0, 0)	1.08	0.26 = 0.18 + 0.02 + 0.06	1.39 (18, 72)	1.01
PSG	PR	0.16 = 0.16 + 0.01	0.03 (100, 100)	1.74	0.20 = 0.15 + 0.01 + 0.04	0.08 (100, 100)	1.72
	TW	0.23 = 0.22 + 0.02	1.12 (4, 60)	2.60	0.28 = 0.21 + 0.02 + 0.05	0.51 (26, 86)	2.52
	PA	0.54 = 0.49 + 0.05	1.68 (6, 37)	6.09	0.61 = 0.47 + 0.04 + 0.09	0.33 (22, 97)	6.01
	UK	OOM	OOM	OOM	0.65 = 0.49 + 0.03 + 0.13	3.37 (13, 57)	7.00

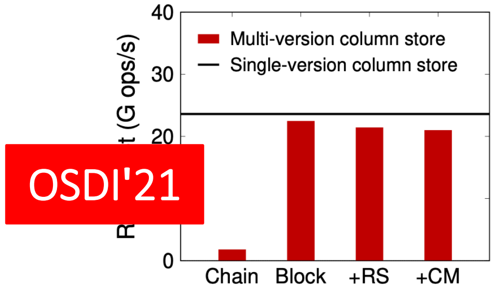
Evaluation-centric Research

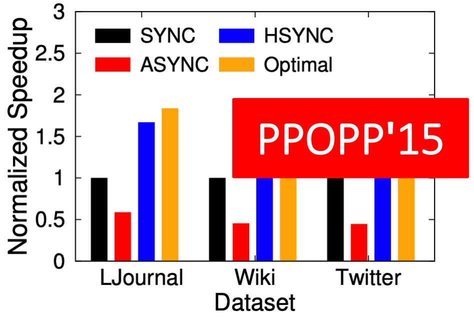
1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation







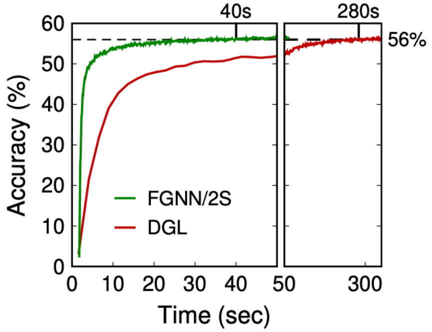




Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution

EuroSys'22 submission



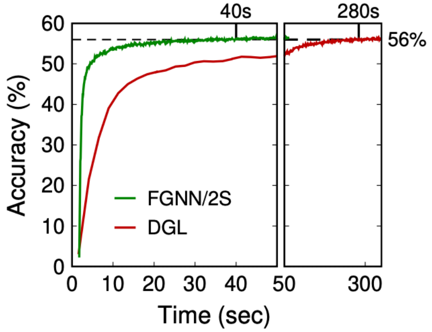
- Converge to the same target
- More faster (7x speedup)
- Fewer epochs (100 vs. 120)

Review: why fewer epochs?

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution

EuroSys'22 submission

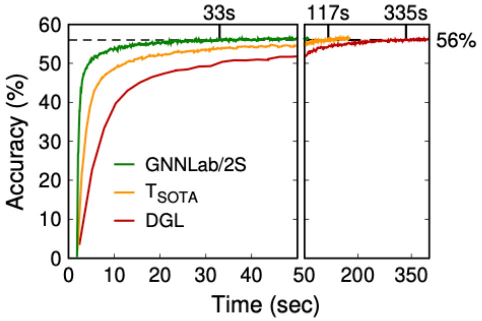


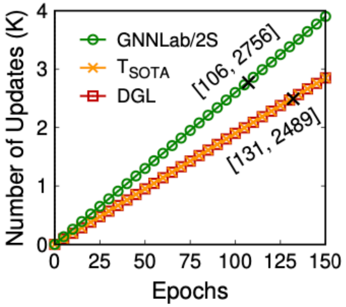
- <u>Converge to the same target</u>
- More faster (7x speedup)
- Fewer epochs (100 vs. 120)

Review: why fewer epochs?

Final

1. Faster training (*per epoch*)
2. Fewer epochs (**NEW**)
   "the fewer trainers, the more gradient updates"



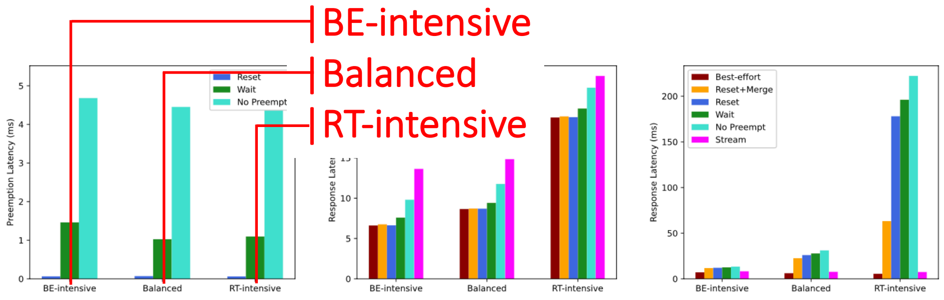


- Correct evaluation
  8.1x · 1.23x = 10x
- New low-level metric
- A new merit of GNNLab ("space sharing")

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution

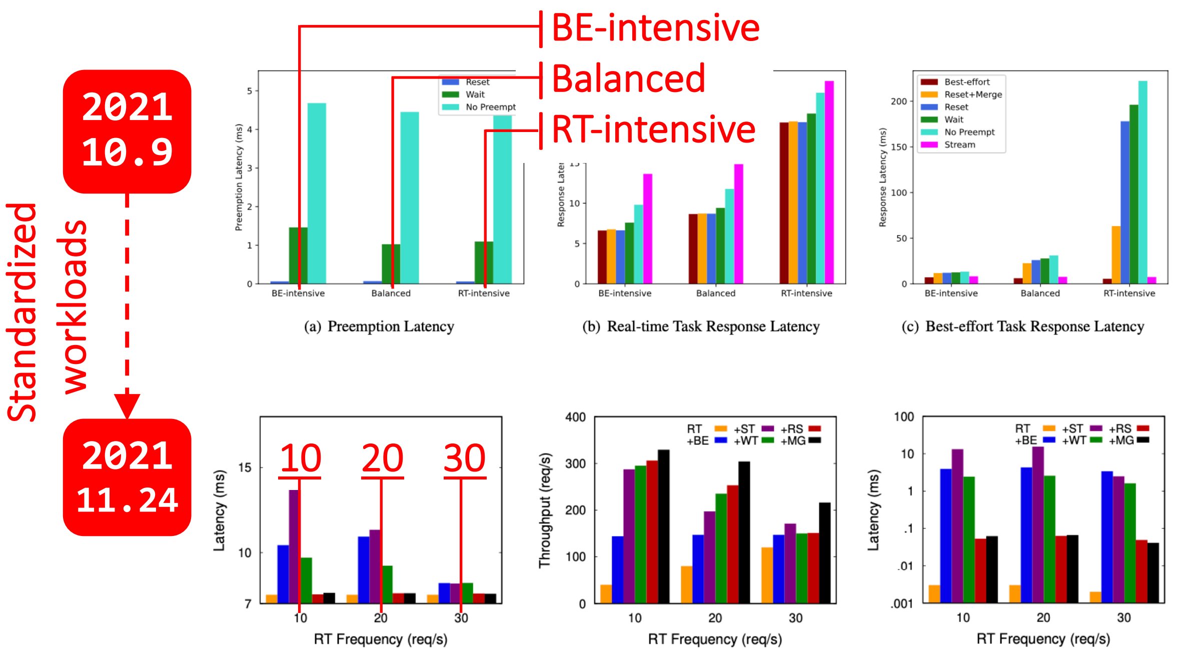
2021
10.9



(a) Preemption Latency (b) Real-time Task Response Latency (c) Best-effort Task Response Latency

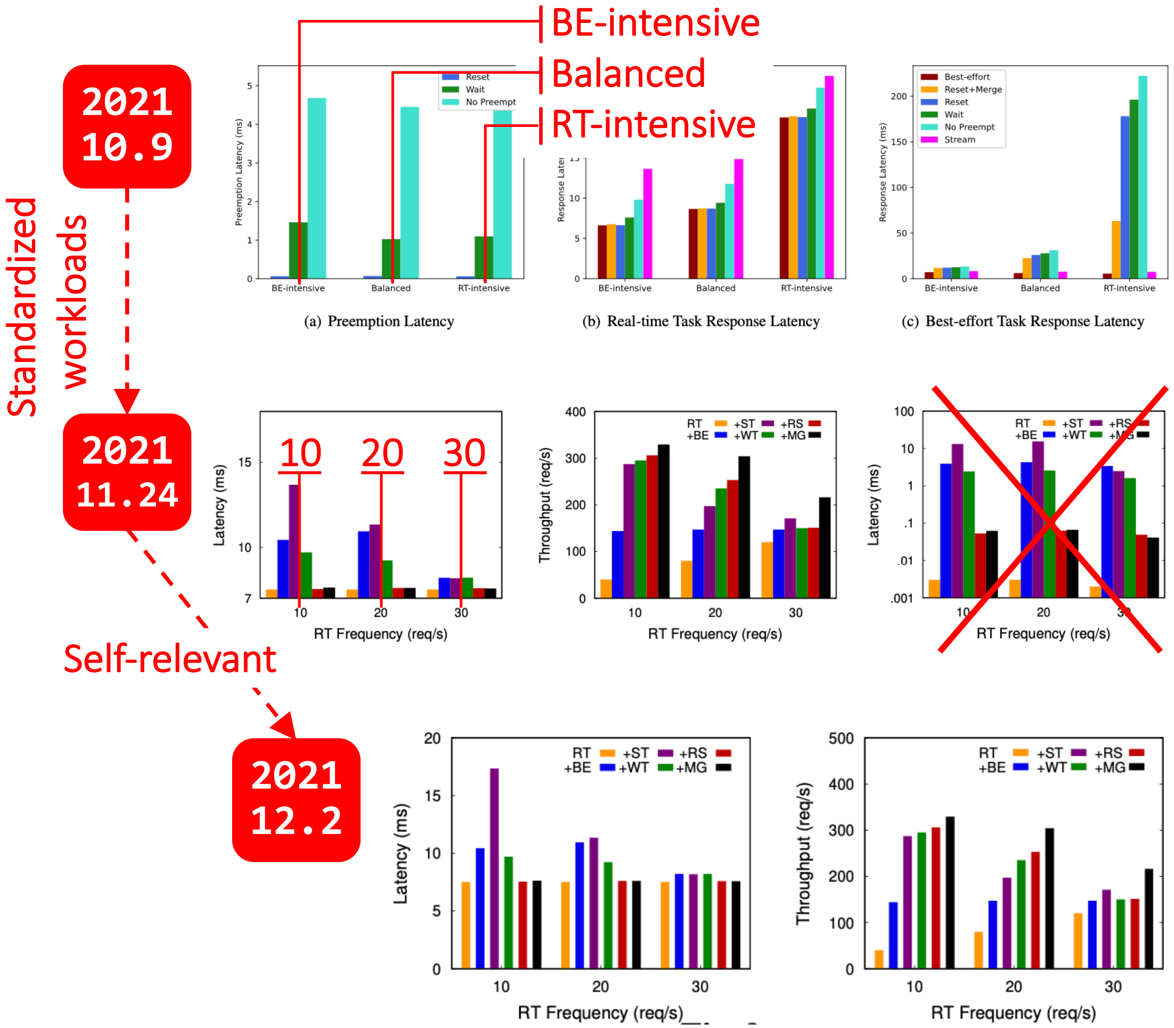
Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution

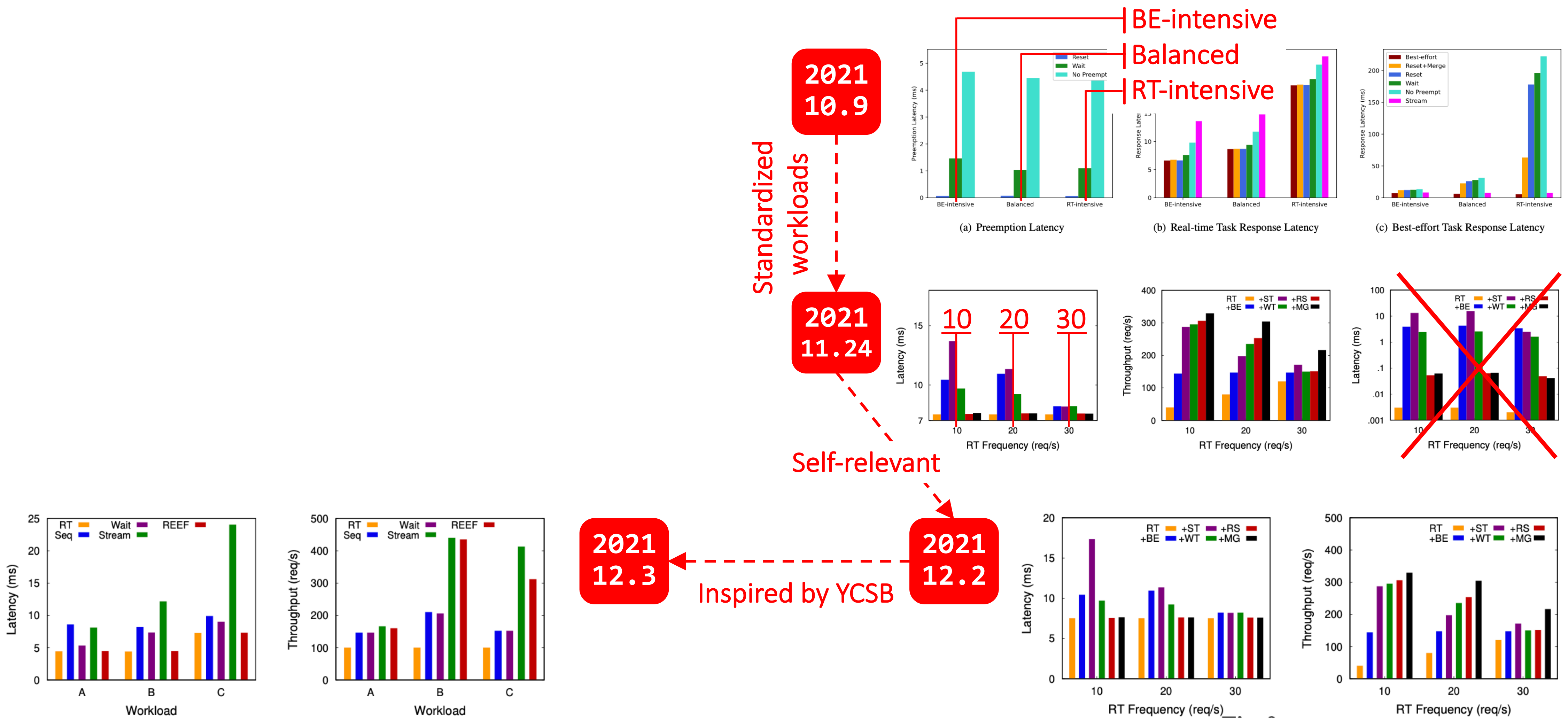


Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution



Evaluation-centric Research



Evaluation-centric Research

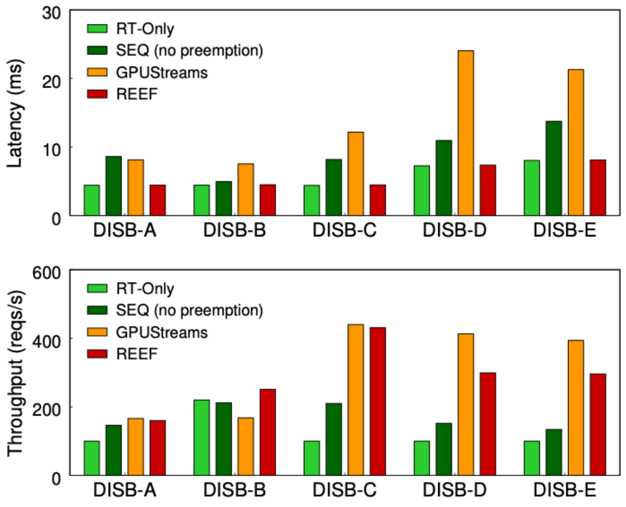


Fig. 10: *Comparison of (a) end-to-end real-time task latency, and (b) overall throughput (including both real-time and best-effort tasks) using different scheduling approaches.*

2021 12.9

Format

2021 12.5

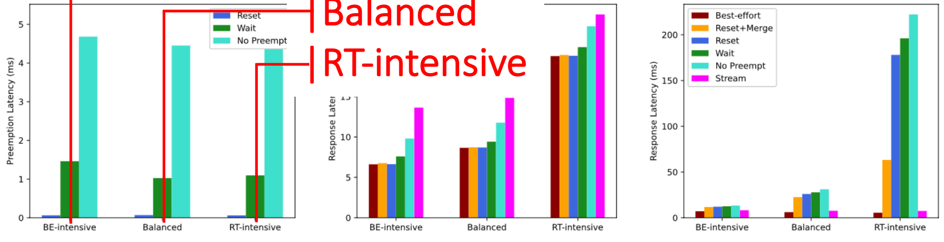
- Diverse workloads
- Clear baseline

2021 10.9

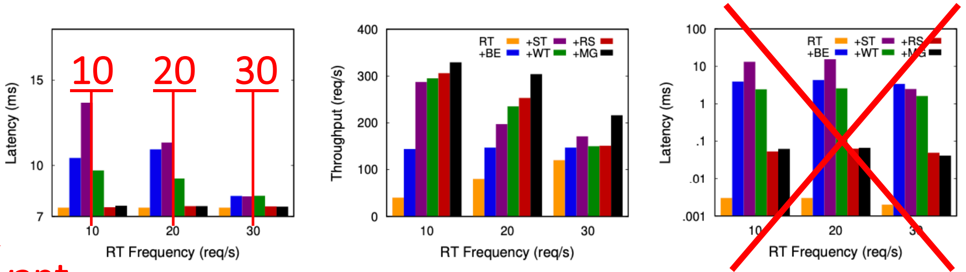
Standardized workloads

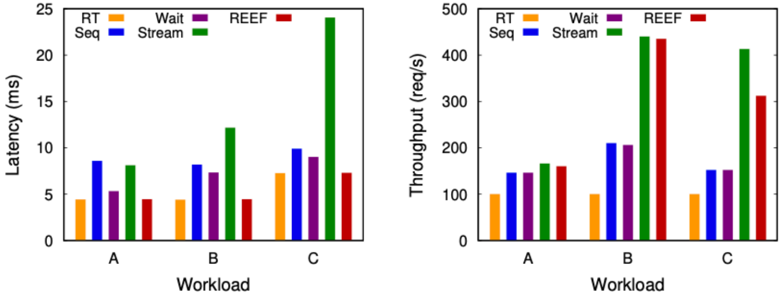
2021 11.24

Self-relevant



(a) Preemption Latency (b) Real-time Task Response Latency (c) Best-effort Task Response Latency

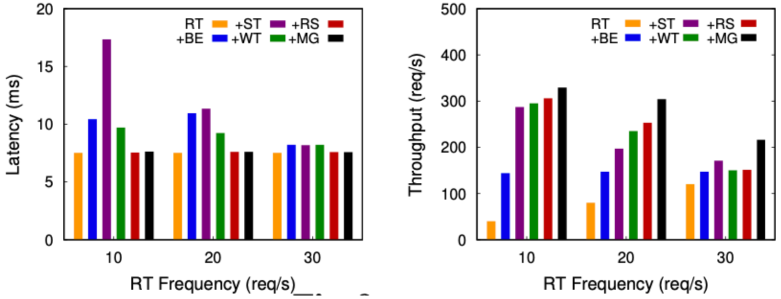




2021 12.3

Inspired by YCSB

2021 12.2



Evaluation-centric Research

OSDI'22 submission

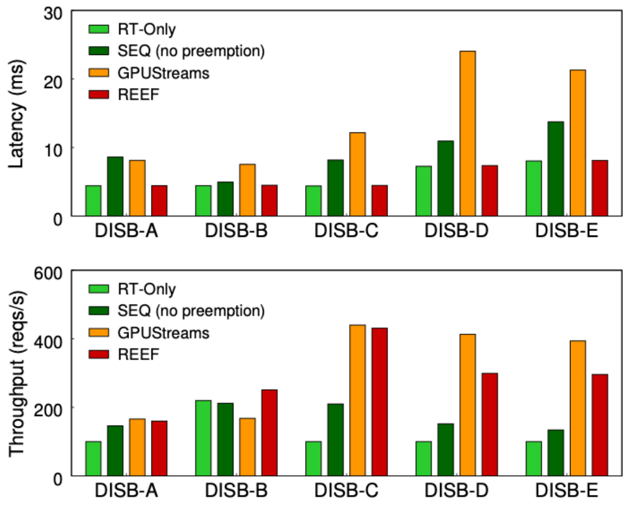


Fig. 10: *Comparison of (a) end-to-end real-time task latency, and (b) overall throughput (including both real-time and best-effort tasks) using different scheduling approaches.*

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution

OSDI '22 Home

Strengths

- interesting problem domain and a nice set of ideas
- comprehensively covers various implementation issues with optimizations to improve performance
- thorough evaluation

---

that be for REEF? The evaluation is pretty thorough, but it certainly shows REEF in a positive light without trying to put it into scenarios where it might struggle. It can be more helpful to show the full spectrum to readers so that we know when to apply REEF and when it may not be suitable.

Evaluation-centric Research

OSDI'22 submission

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution

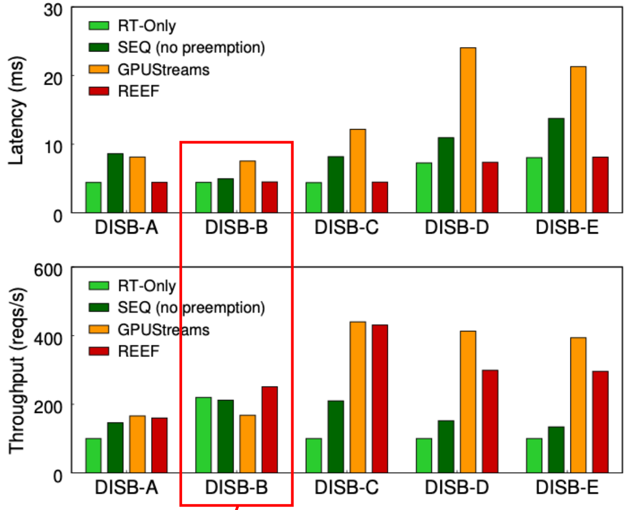


Fig. 10: *Comparison of (a) end-to-end real-time task latency, and (b) overall throughput (including both real-time and best-effort tasks) using different scheduling approaches.*

Table 2: *DISB workload description.* `#/model` *denotes the number of clients and their DNN models.*

DISB	**A**	**B**	**C**	**D**	**E**
Num. of RT-clients Frequency (reqs/s)	1/VGG 100	1/VGG 220	1/VGG 100	5/ALL 20	5/ALL 20
Num. of BE-clients	1/RNET	1/RNET	5/ALL	5/ALL	5/ALL

OSDI '22 Home

Strengths

- interesting problem domain and a nice set of ideas
- comprehensively covers various implementation issues with optimizations to improve performance
- thorough evaluation

that be for REEF? The evaluation is pretty thorough, but it certainly shows REEF in a positive light without trying to put it into scenarios where it might struggle. It can be more helpful to show the full spectrum to readers so that we know when to apply REEF and when it may not be suitable.

Evaluation-centric Research

OSDI'22 submission

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution

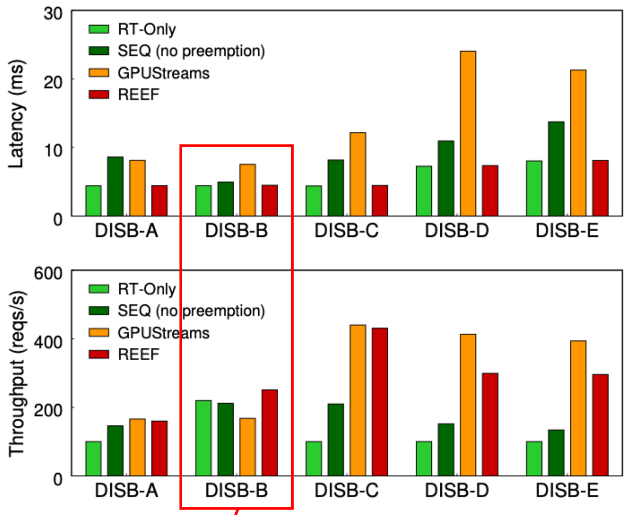


Fig. 10: *Comparison of (a) end-to-end real-time task latency, and (b) overall throughput (including both real-time and best-effort tasks) using different scheduling approaches.*

Table 2: *DISB workload description.* `#/model` *denotes the number of clients and their DNN models.*

DISB	A	B	C	D	E
Num. of RT-clients Frequency (reqs/s)	1/VGG 100	1/VGG 220	1/VGG 100	5/ALL 20	5/ALL 20
Num. of BE-clients	1/RNET	1/RNET	5/ALL	5/ALL	5/ALL

OSDI '22 Home

Strengths

- interesting problem domain and a nice set of ideas
- comprehensively covers various implementation issues with optimizations to improve performance
- thorough evaluation

that be for REEF? The evaluation is pretty thorough, but it certainly shows REEF in a positive light without trying to put it into scenarios where it might struggle. It can be more helpful to show the full spectrum to readers so that we know when to apply REEF and when it may not be suitable.

Claim more explicitly

Evaluation-centric Research

OSDI'22 submission

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution

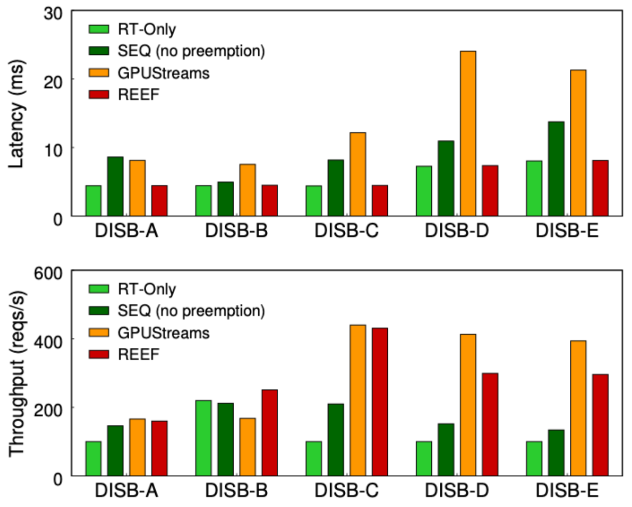


Fig. 10: *Comparison of (a) end-to-end real-time task latency, and (b) overall throughput (including both real-time and best-effort tasks) using different scheduling approaches.*

OSDI '22 Home

Significant weaknesses

1. The micro-benchmarks and the models could be used in many more configurations in the evaluation. This would make a more convincing case for the system.

. . .

2. Why not use a diurnal trace of user requests, of which there are many, to determine a somewhat realistic arrival pattern for real-time workloads? You could then design benchmarks by varying the kind of background workloads (synthetically of course) based on the usage scenario of the background task.

. . .

4. DISB Workload Description: Since the benchmark is new and introduced only in this paper, we need more details on this. For instance, it is unclear what a random workload is. Also, it is not clear what the underlying models used are. It would also be good for these benchmarks to be made publicly available.

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

2021
8.25

GPU cache

- SOTA: static cache + degree-based policy
- Idea#1: Dynamic cache + *approx.* prefetching
- Idea#2: CPU/GPU hybrid extracting

CPU/GPU feature store: Caching and Prefetching

- Caching policy (GPU): training set + 1-hop + high-degree (static) / approx. prefetching (
  - Intermediate buffer size: predictable (memory consumption for sampling-based tra
- Hybrid extractor and samples
  - GPU-DS (redundant): good locality, fast extractor
    - Array??
  - CPU-DS (dedup): fast loading (small data size)
    - CSC??
  - How to merge? Hybrid (data+ptr)? Locality?
- (The first) Dynamic (pre)fetching
  - Limitation of static caching @R3 (PaGraph)
    - Depends on high skewness of graph (power-law)
    - The cache hit rate is low?? @R3
  - NOT heuristic (e.g., LRU), no replacement
    - classic replacement algorithms typically rely on heuristics and empirical obser
    - Our approach is simple, well-grounded, robust, and performant.
  - OB: the direction of sampling and training is reversed
  - Approximate prefetching: sampling 1-hop and prefetching the rest
    - Precision: **Larger fan-out for first layer (leaf) and Smaller fan-out for la**
      - e.g., For GraphSage, sampling 10 immediate neighbors and 10x25 2-ho
      - Large pruning prefetching space and high ratio of data prefetched
    - Efficiency: the more prefetching, the less moving)

Motivation

Observation

Key tech

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

2021
8.30

宋小牛

August 30, 2021 at 09:09

宋

Re: Weekly report 2021-08-23~08-29

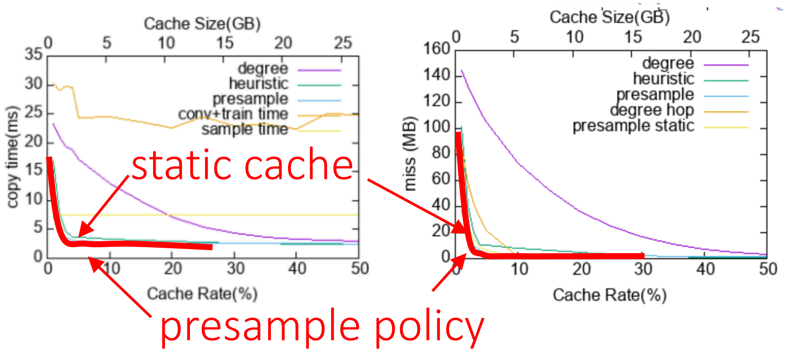
To: 分布式系统方向 IPADS, 陈榕

Objective: GNN Cache
Key Results:

- presample and several other policy implementation

Last Week:

- develop:
  - implement several cache policy presample, subgraph degree, presample full neighbour: [f5b2378, 565a043, 4b74272]
  - fix legacy bug: crash when cache 100% [816258b]
- evaluate fine-grained performance over cache rate
  - left: papers100; right: friendster



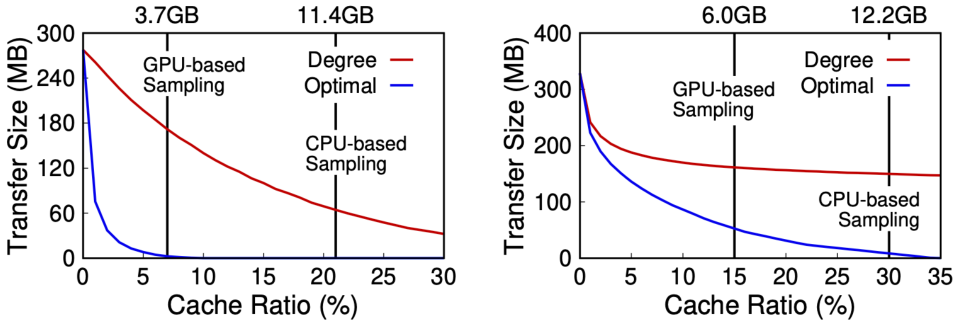
- try to get explaination of why degree gets better or worse, summary:
  - one hop degree does not match sample probability
  - but its hard to build a metric.
  - **proposed guideline:** presample is close to optimal and robust, so the choice between ~~degree and~~ presample is not necessary?
- try to deduce the optimal cache policy

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

New Story

Motivating experiments



Perforamnce Issues: 1) Capacity & 2) Efficiency

- Efficiency: Caching Policy
- SOTA: degree-based policy
  1. Narrow assumption of graph structures
  2. Unaware of sampling algorithms

> The second challenge is *how to achieve optimal cache efficiency for diverse GNN datasets and sampling algorithms.*

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

New Story

6 GPU-based Feature Caching

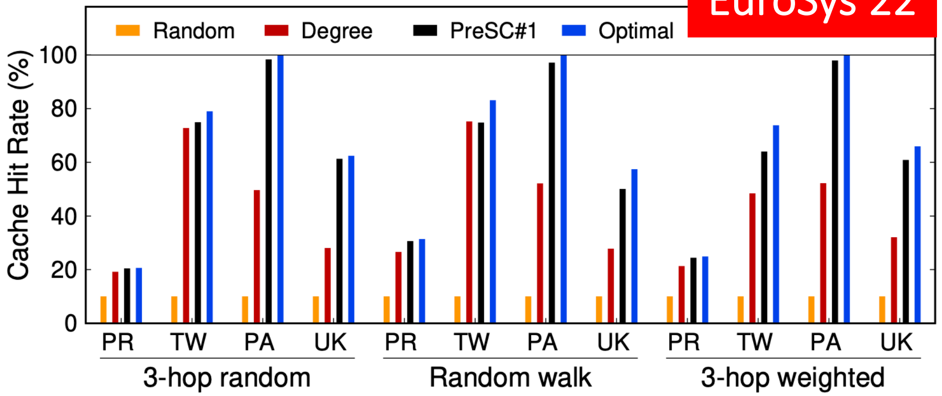
6.1 A General Caching Scheme

6.2 Analysis of Caching Policy

6.3 A Pre-sampling Based Caching Policy

- Efficiency
- Robustness

EuroSys'22



Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

2021
8.01

4 Overview of FGNN

4.1 Our approach: Factored GNN

4.2 Framework

4.3 Challenges

5 Caching

5.1 Static cache with adaptive pre-filling

5.2 Dynamic cache with approximated prefetching

6 Role-based Pipelining

???

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

2021
9.09

4 Overview of FGNN

4.1 Overview of Factored Design

4.2 Framework

5 Role-based Pipelining

5.1 Sampler / Extractor / Trainer

5.2 Load balance

5.3 Running on a single GPU

5.4 Supporting heterogenous environment

6 GPU-based Feature Caching

6.1 General caching policy

6.2 Memory allocation for the cache

6.3 Integration into training process

6.4 Cache data management

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

2021
9.26

4 Overview of FGNN

Opportunity: inter-task locality

Our approach: a factored design

5 FGNN Architecture

5.1 Programming Model

5.2 Hybrid Execution

6 GPU-based Feature Caching

6.1 General caching policy

6.2 Memory allocation for the cache

6.3 Integration into training process

6.4 Cache data management

Evaluation-centric Research

Table 5: End-to-end time breakdown in seconds for one epoch of DGL with GPU sampling. For all models, we use 8000 as the start batch size. For GCN, we uses a 3-layer model. The fanout from root to leaf are 15, 10, 5. For GraphSAGE, we use a 2-layer model. The fanout from root to leaf are 10, 25. For PinSAGE], we use a 3-layer model. The fanout for all layers are 5. In each layer, every node does 4 random walks with length of 3. And every node selects the top 5 nodes with highest frequency from the random walk traces as its neighbors. The hidden embedding in all models is 256 dimensions. For UK-2006 dataset, it gets a runtime failure because the graph structure in DGL is out of GPU memory. S stands for sample time. I stands for the time calculating the cache index. Q stands for the time queuing and copying graph to share memory. C stands for the time converting to DGL data structure. T stands for the training time

Model	Dataset	Sample	Extract	Train	Cache Pct.	Hit Rate	Sample(S+I+Q)	Extract	Train(T+C)
GCN	Products	0.35	2.82	0.74	1.00	1.00	0.40(0.29+0.03+0.08)	0.24	0.94(0.91+0.03)
	Papers	1.19	10.70	2.64	0.20	0.99	1.00(0.69+0.17+0.14)	0.90	2.85(2.67+0.18)
	Twitter	0.74	8.64	1.06	0.24	0.89	0.39(0.26+0.07+0.06)	1.09	1.09(1.02+0.07)
	UK-2006	FAIL	FAIL	FAIL	0.15	0.67	0.59(0.39+0.08+0.11)	3.96	2.23(2.06+0.17)
GraphSAGE	Products	0.14	1.93	0.18	1.00	1.00	0.21(0.15+0.02+0.04)	0.11	0.16(0.14+0.02)
	Papers	0.56	6.08	1.09	0.24	0.99	0.48(0.31+0.10+0.06)	0.46	0.90(0.79+0.11)
	Twitter	0.38	4.27	0.38	0.32	0.89	0.17(0.11+0.03+0.03)	0.63	0.34(0.30 + 0.04)
	UK-2006	FAIL	FAIL	FAIL	0.16	0.69	0.28(0.18+0.05+0.05)	2.02	0.78(0.68+0.10)
PinSAGE	Products				1.00	1.00	0.21(0.16+0.02+0.04)	0.12	1.58(1.55+0.03)
	Papers				0.19	0.97	0.63(0.47+0.08+0.08)	0.64	5.19(4.99+0.20)
	Twitter				0.25	0.86	0.29(0.21+0.03+0.04)	0.78	1.78(1.71+0.08)
	UK-2006	FAIL	FAIL	FAIL	0.11	0.52	0.67(0.49+0.07+0.11)	4.17	3.63(3.44+0.19)

stage-by-stage breakdown

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

2021 9.29

4 Overview of FGNN

Opportunity: inter-task locality

Our approach: a factored design

Challenge: load imbalance

5 FGNN Architecture

5.1 Programming Model

5.2 Hybrid Execution

5.3 Dynamic Scheduling

Adaptive switch btw. Sampler & Trainer

Running on a single GPU

New Story

6 GPU-based Feature Caching

6.1 A general caching scheme

6.2 GPU-based feature caching via pre-sampling

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

2021
9.29

Table 5: End-to-end time breakdown in seconds for one epoch of DGL with GPU sampling. For all models, we use 8000 as the start batch size. For GCN, we uses a 3-layer model. The fanout from root to leaf are 15, 10, 5. For GraphSAGE, we use a 2-layer model. The fanout from root to leaf are 10, 25. For PinSAGE], we use a 3-layer model. The fanout for all layers are 5. In each layer, every node does 4 random walks with length of 3. And every node selects the top 5 nodes with highest frequency from the random walk traces as its neighbors. The hidden embedding in all models is 256 dimensions. For UK-2006 dataset, it gets a runtime failure because the graph structure in DGL is out of GPU memory. S stands for sample time. I stands for the time calculating the cache index. Q stands for the time queuing and copying graph to share memory. C stands for the time converting to DGL data structure. T stands for the training time

Model	Dataset	Sample	Extract	Train	Cache Pct.	Hit Rate	Sample(S+I+Q)	Extract	Train(T+C)
GCN	Products	0.35	2.82	0.74	1.00	1.00	0.40(0.29+0.03+0.08)	0.24	0.94(0.91+0.03)
	Papers	1.19	10.70	2.64	0.20	0.99	1.00(0.69+0.17+0.14)	0.90	2.85(2.67+0.18)
	Twitter	0.74	8.64	1.06	0.24	0.89	0.39(0.26+0.07+0.06)	1.09	1.09(1.02+0.07)
	UK-2006	FAIL	FAIL	FAIL	[illegible]	0.67	[illegible]	3.96	2.23(2.06+0.17)
GraphSAGE	Products	0.14	[illegible]	[illegible]	1.00	1.00	[illegible]	0.11	0.16(0.14+0.02)
	Papers	0.56	6.08	1.09	0.24	0.99	0.48(0.31+0.10+0.06)	0.46	0.90(0.79+0.11)
	Twitter	0.38	4.27	0.38	0.32	0.89	0.17(0.11+0.03+0.03)	0.63	0.34(0.30 + 0.04)
	UK-2006	FAIL	FAIL	FAIL	[illegible]	[illegible]	[illegible]	2.02	0.78(0.68+0.10)
PinSAGE	Products				1.00	1.00	0.21(0.16+0.02+0.04)	0.12	1.58(1.55+0.03)
	Papers				0.19	0.97	0.63(0.47+0.08+0.08)	0.64	5.19(4.99+0.20)
	Twitter				0.25	0.86	0.29(0.21+0.03+0.04)	0.78	1.78(1.71+0.08)
	UK-2006	FAIL	FAIL	FAIL	0.11	0.52	0.67(0.49+0.07+0.11)	4.17	3.63(3.44+0.19)

stage-by-stage breakdown

4 Overview of FGNN

Opportunity: inter-task locality

Our approach: a factored design

Challenge: load imbalance

5 FGNN Architecture

5.1 Programming Model

5.2 Hybrid Execution

5.3 Dynamic Scheduling

Adaptive switch btw. Sampler & Trainer

Running on a single GPU

New Story

6 GPU-based Feature Caching

6.1 A general caching scheme

6.2 GPU-based feature caching via pre-sampling

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

Table 5: The time breakdown (in seconds) of one epoch for DGL and FGNN. For UK dataset, it gets a runtime failure because the graph structure in DGL is out of GPU memory. S, M, and C indicate the time in the Sample stage for graph sampling, marking cached vertices, and copying samples to the host memory, respectively. X and T indicate the time in the Train stage for transforming the inputs to DGL format and model training, respectively.

GNN Model	Dataset	DGL			FGNN		
		Sample	Extract	Train	Sample = S + M + C	Extract (Ratio, Hit%)	Train = X + T
GCN	PR	0.35	2.81	1.22	0.39 = 0.29 + 0.01 + 0.09	0.19 (100%,100%)	1.18 = 1.15 + 0.03
	PA	1.20	10.70	4.00	0.96 = 0.68 + 0.10 + 0.18	0.61 (21%, 99%)	3.84 = 3.69 + 0.15
	TW	0.74	9.44	1.48	0.37 = 0.26 + 0.03 + 0.08	0.80 (25%, 89%)	1.51 = 1.45 + 0.06
	UK	OOM	OOM	OOM	0.56 = [illegible] + 0.03 + [illegible]	3.24 (13%, 67%)	3.15 = 2.98 + 0.16
GraphSAGE	PR	0.13	1.92	[illegible]	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	0.25 = 0.23 + 0.02
	PA	0.56	6.06	1.25	0.46 = 0.31 + 0.06 + 0.08	0.34 (25%, 99%)	1.10 = 1.00 + 0.10
	TW	0.38	4.65	0.44	0.16 = 0.11 + 0.01 + 0.03	0.43 (33%, 89%)	0.42 = 0.38 + 0.04
	UK	OOM	OOM	OOM	0.26 = 0.18 + 0.02 + [illegible]	1.60 (16%, 69%)	1.02 = 0.92 + 0.10
PinSAGE	PR	0.16	1.56	1.75	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	1.73 = 1.70 + 0.03
	PA	0.53	5.00	6.14	0.61 = 0.47 + 0.04 + 0.09	0.40 (20%, 97%)	5.96 = 5.78 + 0.18
	TW	0.23	4.97	2.57	0.28 = 0.21 + 0.02 + 0.05	0.55 (26%, 86%)	2.53 = 2.47 + 0.07
	UK	OOM	OOM	OOM	0.65 = 0.48 + 0.03 + 0.13	3.68 (11%, 52%)	7.00 = 6.80 + 0.19

stage-by-stage breakdown

2021 10.3

5 FGNN Architecture

5.1 Programming Model

5.2 Hybrid Execution

5.3 Dynamic Scheduling

EuroSys'22

New Story

$$\frac{T_t}{T_s} = K \quad and \quad \left\lceil \frac{N_g}{K+1} \right\rceil = N_s,$$

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

2021 10.6

5 FGNN Architecture

5.1 Programming Model

5.2 Hybrid Execution

5.3 Dynamic Scheduling

EuroSys'22

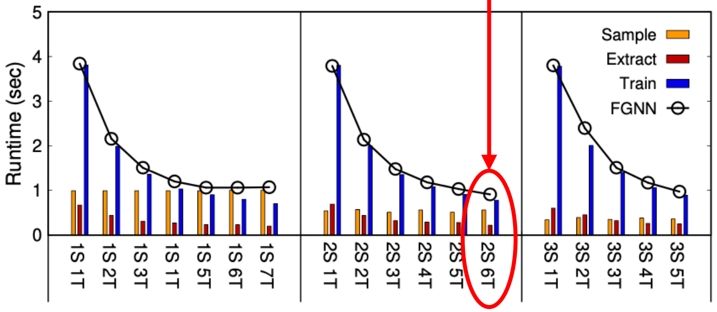
New Story

$$\frac{T_t}{T_s} = K \quad and \quad \left\lceil \frac{N_g}{K+1} \right\rceil = N_s,$$

Table 5: The time breakdown (in seconds) of one epoch for DGL and FGNN. For UK dataset, it gets a runtime failure because the graph structure in DGL is out of GPU memory. S, M, and C indicate the time in the Sample stage for graph sampling, marking cached vertices, and copying samples to the host memory, respectively. X and T indicate the time in the Train stage for transforming the inputs to DGL format and model training, respectively.

stage-by-stage breakdown

GNN Model	Dataset	DGL			FGNN		
		Sample	Extract	Train	Sample = S + M + C	Extract (Ratio, Hit%)	Train = X + T
GCN	PR	0.35	2.81	1.22	0.39 = 0.29 + 0.01 + 0.09	0.19 (100%,100%)	1.18 = 1.15 + 0.03
	PA	1.20	10.70	4.00	0.96 = 0.68 + 0.10 + 0.18	0.61 (21%, 99%)	3.84 = 3.69 + 0.15
	TW	0.74	9.44	1.48	0.37 = 0.26 + 0.03 + 0.08	0.80 (25%, 89%)	1.51 = 1.45 + 0.06
	UK	OOM	OOM	OOM	0.56 = [illegible] + 0.03 + [illegible]	3.24 ([illegible], 67%)	3.15 = 2.98 + 0.16
GraphSAGE	PR	0.13	1.92	[illegible]	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	0.25 = 0.23 + 0.02
	PA	0.56	6.06	1.25	0.46 = 0.31 + 0.06 + 0.08	0.34 (25%, 99%)	1.10 = 1.00 + 0.10
	TW	0.38	4.65	0.44	0.16 = 0.11 + 0.01 + 0.03	0.43 (33%, 89%)	0.42 = 0.38 + 0.04
	UK	OOM	OOM	OOM	0.26 = 0.18 + 0.02 + [illegible]	1.60 (16%, 69%)	1.02 = 0.92 + 0.10
PinSAGE	PR	0.16	1.56	1.75	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	1.73 = 1.70 + 0.03
	PA	0.53	5.00	6.14	0.61 = 0.47 + 0.04 + 0.09	0.40 (20%, 97%)	5.96 = 5.78 + 0.18
	TW	0.23	4.97	2.57	0.28 = 0.21 + 0.02 + 0.05	0.55 (26%, 86%)	2.53 = 2.47 + 0.07
	UK	OOM	OOM	OOM	0.65 = 0.48 + 0.03 + 0.13	3.68 (11%, 52%)	7.00 = 6.80 + 0.19



GNN Model	Dataset	DGL	PyG	SGNN	FGNN
GCN	PR	1.33	11.91	0.22	0.33 (3S)
	TW	3.86	12.15	1.80	0.47 (2S)
	PA	4.56	14.82	2.46	0.84 (2S)
	UK	OOM	15.04	OOM	1.47 (2S)
GraphSAGE	PR	0.79	8.17	0.07	0.11 (4S)
	TW	1.81	8.18	0.35	0.20 (2S)
	PA	2.47	9.68	0.85	0.30 (2S)
	UK	OOM	9.86	2.01	0.61 (1S)
PinSAGE	PR	0.86	×	0.30	0.40 (1S)
	TW	2.38	×	0.98	0.58 (1S)
	PA	2.79	×	1.65	1.05 (1S)
	UK	OOM	×	OOM	1.81 (1S)

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

Table 5: The time breakdown (in seconds) of one epoch for DGL and FGNN. For UK dataset, it gets a runtime failure because the graph structure in DGL is out of GPU memory. S, M, and C indicate the time in the Sample stage for graph sampling, marking cached vertices, and copying samples to the host memory, respectively. X and T indicate the time in the Train stage for transforming the inputs to DGL format and model training, respectively.

GNN Model	Dataset	DGL			FGNN		
		Sample	Extract	Train	Sample = S + M + C	Extract (Ratio, Hit%)	Train = X + T
GCN	PR	0.35	2.81	1.22	0.39 = 0.29 + 0.01 + 0.09	0.19 (100%,100%)	1.18 = 1.15 + 0.03
	PA	1.20	10.70	4.00	0.96 = 0.68 + 0.10 + 0.18	0.61 (21%, 99%)	3.84 = 3.69 + 0.15
	TW	0.74	9.44	1.48	0.37 = 0.26 + 0.03 + 0.08	0.80 (25%, 89%)	1.51 = 1.45 + 0.06
	UK	OOM	OOM	OOM	0.56 = [illegible] + 0.03 + [illegible]	3.24 (13%, 67%)	3.15 = 2.98 + 0.16
GraphSAGE	PR	0.13	1.92	[illegible]	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	0.25 = 0.23 + 0.02
	PA	0.56	6.06	1.25	0.46 = 0.31 + 0.06 + 0.08	0.34 (25%, 99%)	1.10 = 1.00 + 0.10
	TW	0.38	4.65	0.44	0.16 = 0.11 + 0.01 + 0.03	0.43 (33%, 89%)	0.42 = 0.38 + 0.04
	UK	OOM	OOM	OOM	0.26 = 0.18 + 0.02 + [illegible]	1.60 (16%, 69%)	1.02 = 0.92 + 0.10
PinSAGE	PR	0.16	1.56	1.75	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	1.73 = 1.70 + 0.03
	PA	0.53	5.00	6.14	0.61 = 0.47 + 0.04 + 0.09	0.40 (20%, 97%)	5.96 = 5.78 + 0.18
	TW	0.23	4.97	2.57	0.28 = 0.21 + 0.02 + 0.05	0.55 (26%, 86%)	2.53 = 2.47 + 0.07
	UK	OOM	OOM	OOM	0.65 = 0.48 + 0.03 + 0.13	3.68 (11%, 52%)	7.00 = 6.80 + 0.19

stage-by-stage breakdown

2021
10.6

5 FGNN Architecture

5.1 Programming Model

5.2 Hybrid Execution

5.3 Dynamic Scheduling

EuroSys'22

New Story

$$\frac{T_t}{T_s} = K \quad and \quad \left\lceil \frac{N_g}{K+1} \right\rceil = N_s,$$

Dynamic executor switching

New Story

$$\mathcal{P} = \begin{cases} \frac{M_r \times T_t}{N_t} - T_{t'} & \text{if } N_t > 0 \\ +\infty & \text{if } N_t = 0 \end{cases},$$

Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

Table 5: The time breakdown (in seconds) of one epoch for DGL and FGNN. For UK dataset, it gets a runtime failure because the graph structure in DGL is out of GPU memory. S, M, and C indicate the time in the Sample stage for graph sampling, marking cached vertices, and copying samples to the host memory, respectively. X and T indicate the time in the Train stage for transforming the inputs to DGL format and model training, respectively.

GNN Model	Dataset	DGL			FGNN		
		Sample	Extract	Train	Sample = S + M + C	Extract (Ratio, Hit%)	Train = X + T
GCN	PR	0.35	2.81	1.22	0.39 = 0.29 + 0.01 + 0.09	0.19 (100%,100%)	1.18 = 1.15 + 0.03
	PA	1.20	10.70	4.00	0.96 = 0.68 + 0.10 + 0.18	0.61 (21%, 99%)	3.84 = 3.69 + 0.15
	TW	0.74	9.44	1.48	0.37 = 0.26 + 0.03 + 0.08	0.80 (25%, 89%)	1.51 = 1.45 + 0.06
	UK	OOM	OOM	OOM	0.56 = [illegible] + 0.03 + 0.14	3.24 (13%, 67%)	3.15 = 2.98 + 0.16
GraphSAGE	PR	0.13	1.92	[illegible]	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	0.25 = 0.23 + 0.02
	PA	0.56	6.06	1.25	0.46 = 0.31 + 0.06 + 0.08	0.34 (25%, 99%)	1.10 = 1.00 + 0.10
	TW	0.38	4.65	0.44	0.16 = 0.11 + 0.01 + 0.03	0.43 (33%, 89%)	0.42 = 0.38 + 0.04
	UK	OOM	OOM	OOM	0.26 = 0.18 + 0.02 + [illegible]	1.60 (16%, 69%)	1.02 = 0.92 + 0.10
PinSAGE	PR	0.16	1.56	1.75	0.20 = 0.15 + 0.01 + 0.04	0.10 (100%,100%)	1.73 = 1.70 + 0.03
	PA	0.53	5.00	6.14	0.61 = 0.47 + 0.04 + 0.09	0.40 (20%, 97%)	5.96 = 5.78 + 0.18
	TW	0.23	4.97	2.57	0.28 = 0.21 + 0.02 + 0.05	0.55 (26%, 86%)	2.53 = 2.47 + 0.07
	UK	OOM	OOM	OOM	0.65 = 0.48 + 0.03 + 0.13	3.68 (11%, 52%)	7.00 = 6.80 + 0.19

stage-by-stage breakdown

2021 10.6

5 FGNN Architecture

5.1 Programming Model

5.2 Hybrid Execution

5.3 Dynamic Scheduling

EuroSys'22

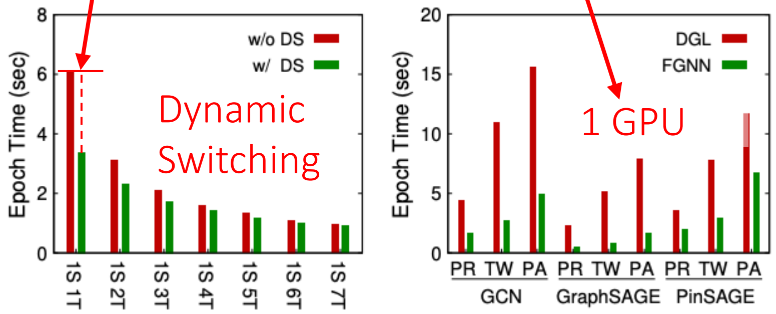
New Story

$$\frac{T_t}{T_s} = K \quad and \quad \left\lceil \frac{N_g}{K+1} \right\rceil = N_s,$$

Dynamic executor switching

New Story

$$\mathcal{P} = \begin{cases} \frac{M_r \times T_t}{N_t} - T_{t'} & \text{if } N_t > 0 \\ +\infty & \text{if } N_t = 0 \end{cases},$$



Evaluation-centric Research

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

2021
10.6

5 FGNN Architecture

5.1 Programming Model

5.2 Hybrid Execution

5.3 Dynamic Scheduling/Switching

$$\frac{T_t}{T_s} = K \quad and \quad \left\lceil \frac{N_g}{K+1} \right\rceil = N_s, \quad \mathcal{P} = \begin{cases} \frac{M_r \times T_t}{N_t} - T_{t'} & \text{if } N_t > 0 \\ +\infty & \text{if } N_t = 0 \end{cases}$$

EuroSys 2022 Home

2. Dynamic switching is a novel idea that sounds interesting, but is poorly motivated since it doesn't actually help their considered workloads.

   The authors don't demonstrate in any of their datasets/models a scenario where adaptive/dynamic switching occurs "naturally". Instead, they need to artificially create this scenario in section 7.8 by forcing an unbalanced sampler/trainer scenario that their system wouldn't choose in the first place. Given that training workloads have predictable iterable steady state behaviour, this is expected. Perhaps this technique would have more value in more unpredictable workload settings (e.g., non-homogenous hardware, workloads contending for GPU resources and slowing down trainers/samplers).

Conclusion & Thanks

1. Motivate your work
2. Support your observation
3. Revise your implementation
4. Realize your limit/limitation
5. Find new contribution
6. Change your story

Evaluation-*centric* Systems Research

More …?

ACKNOWLEDGMENT

Thanks to all authors of papers that contribute cases for this talk, including OSDI'22/'21/'20, EuroSys'22, ATC'21/'19, NSDI'21, SOCC'21. [see also https://ipads.se.sjtu.edu.cn/~rchen]