GLIST: Towards In-Storage Graph Learning

Cangyuan Li [1,2], Ying Wang [1,2], Cheng Liu [1,2], Shengwen Liang [1,2], Huawei Li [1,2,3], Xiaowei Li [1,2]

[1]State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing
[2]University of Chinese Academy of Sciences, Beijing
[3]Peng Cheng Laboratory, Shenzhen

University of Chinese Academy of Sciences

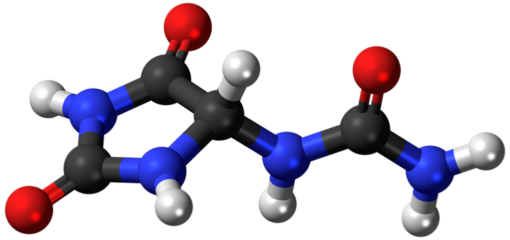
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

Outline

- Background and Motivation
- GLIST Design
- Evaluation
- Conclusion

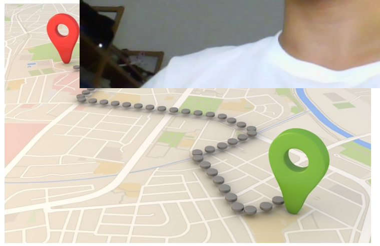
Background of Graph Learning

Social Network



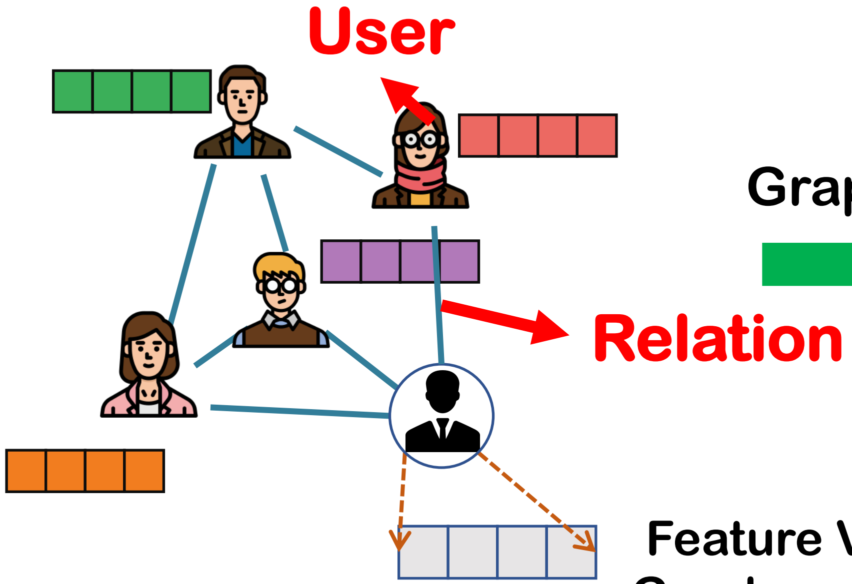
Molecular structures

Knowledge Graph



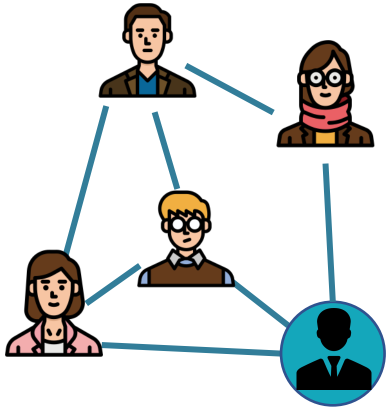
Road Map

Graph Data is Everywhere !

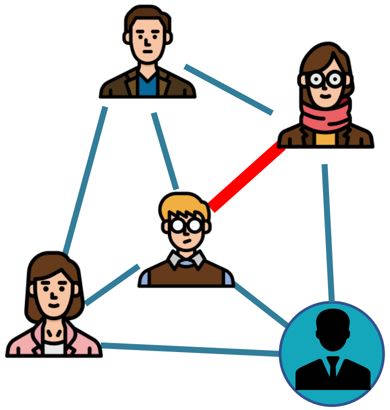


Feature Vector:
Gender , Age, ...

Graph Learning

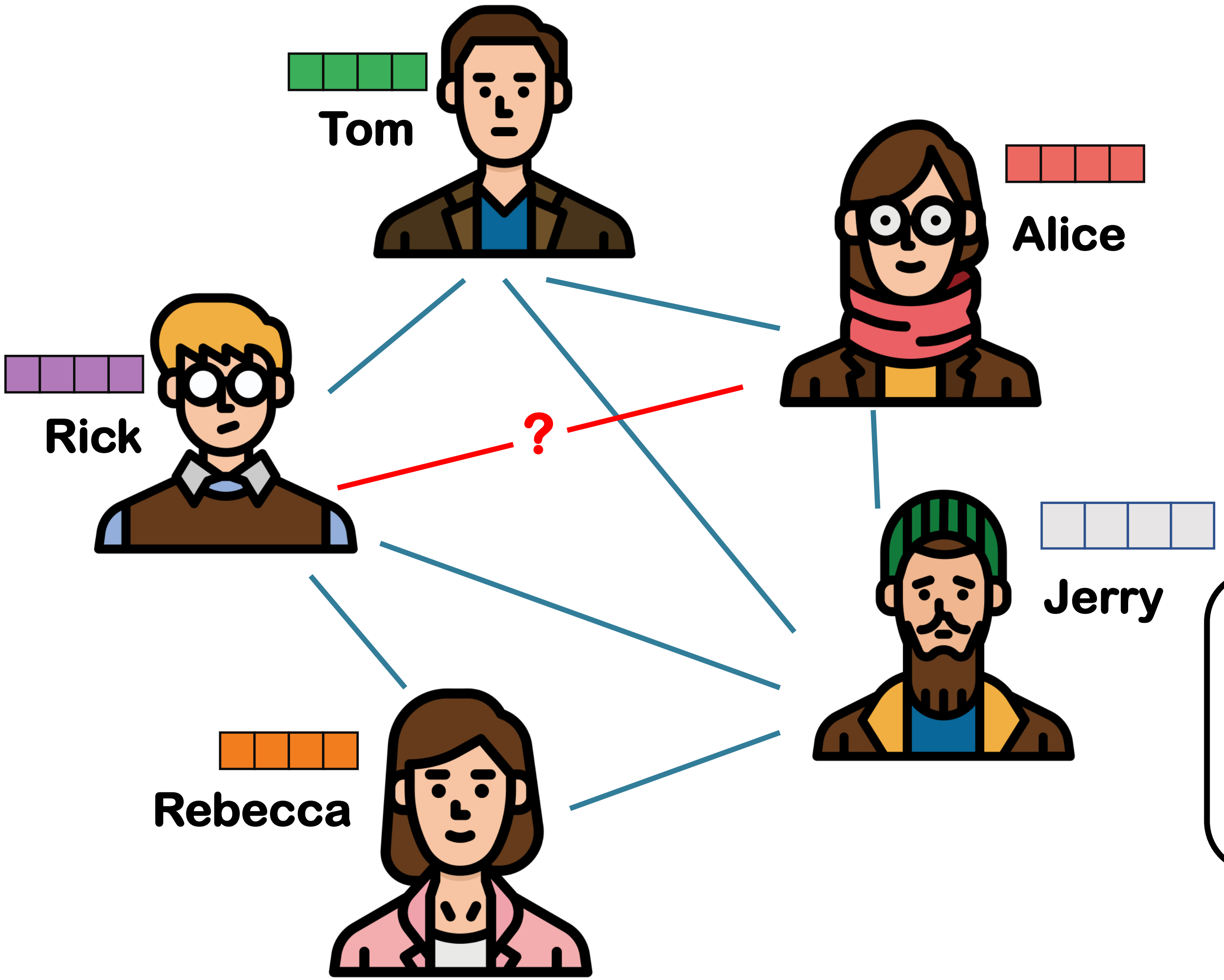


Classify Users



Relationship Prediction

Background of GNN-based GL

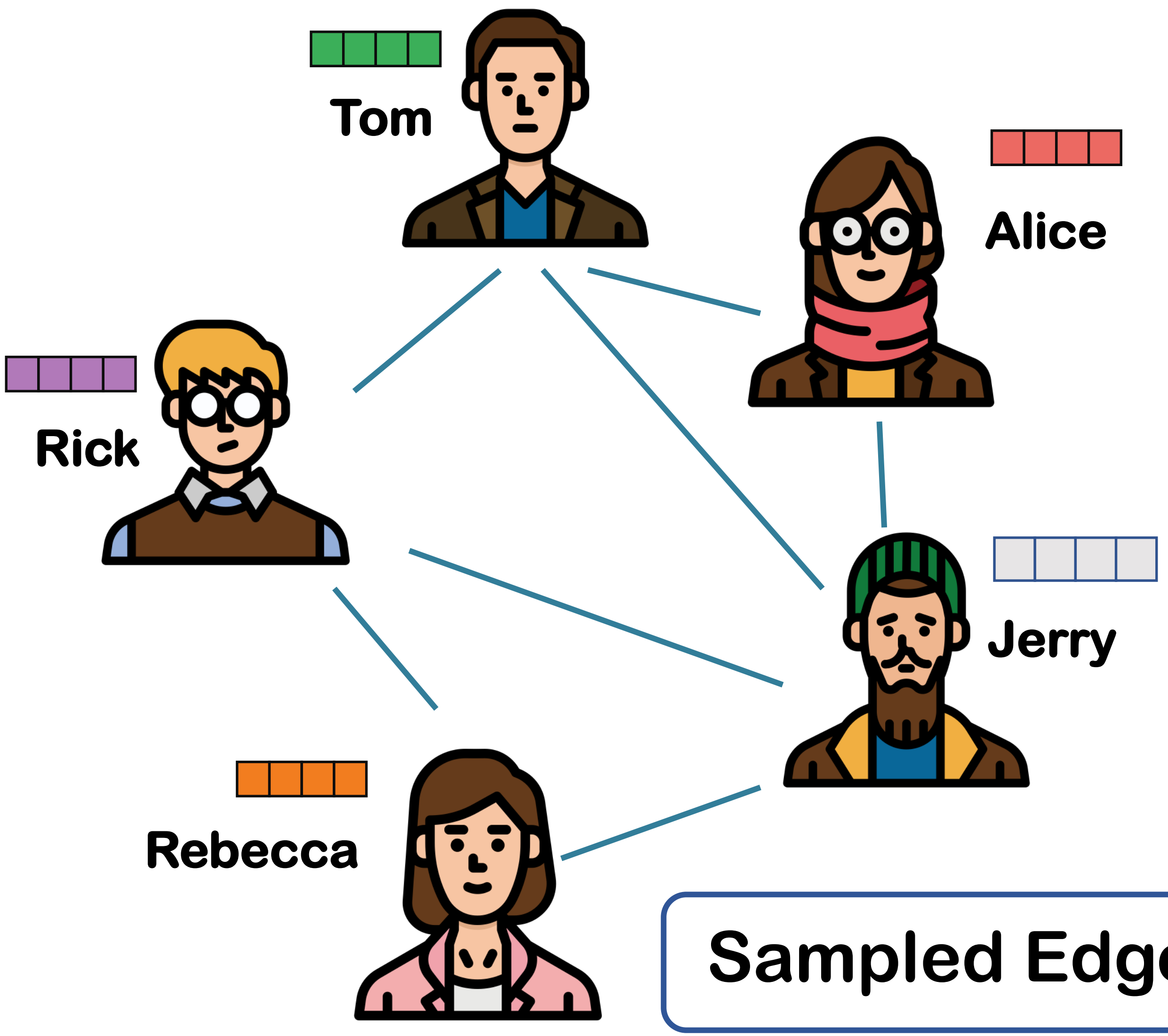


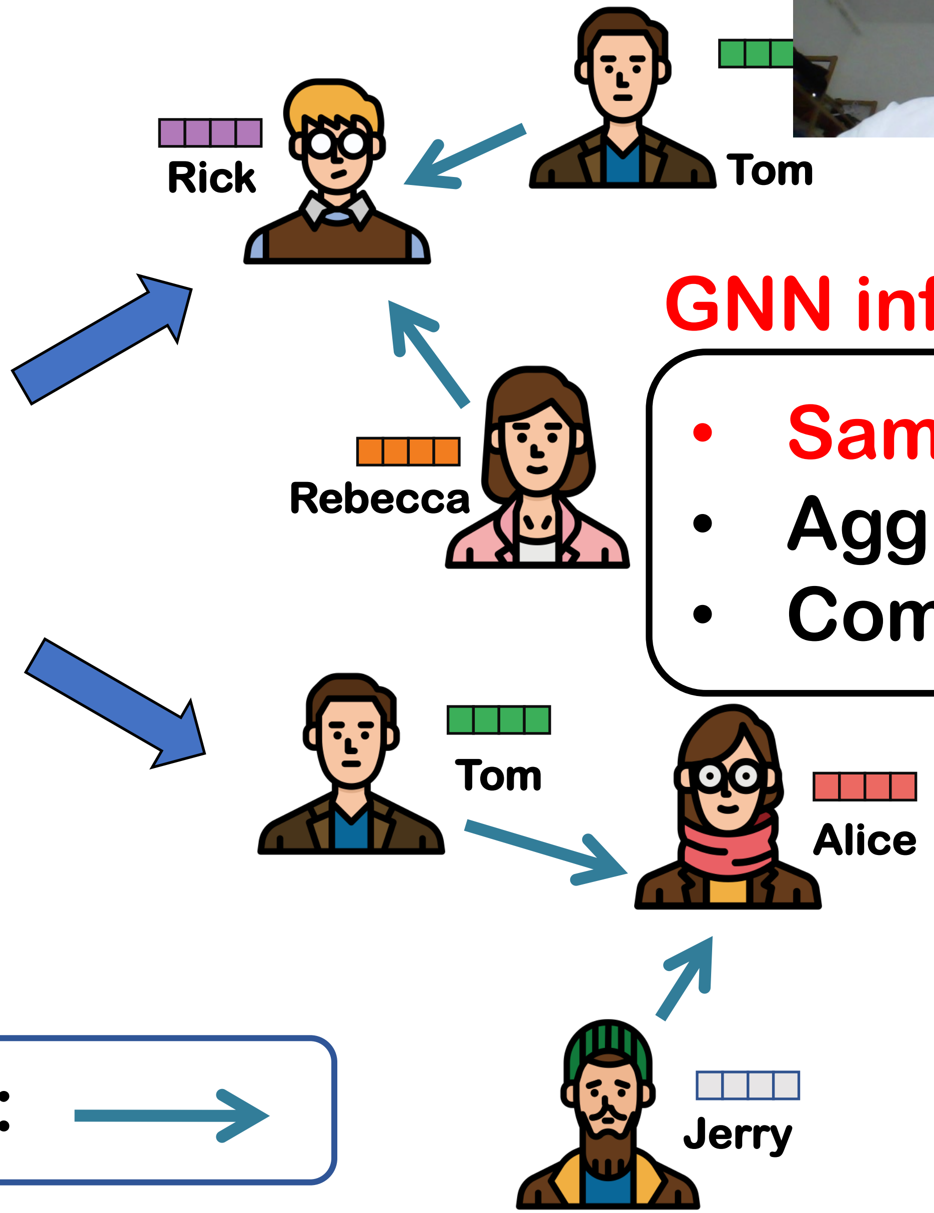
Does Rick know Alice?

Graph learning approach:

- Stage 1: Embedding
- Stage 2: Prediction

Obtain Embedding Vectors with GNN - 1



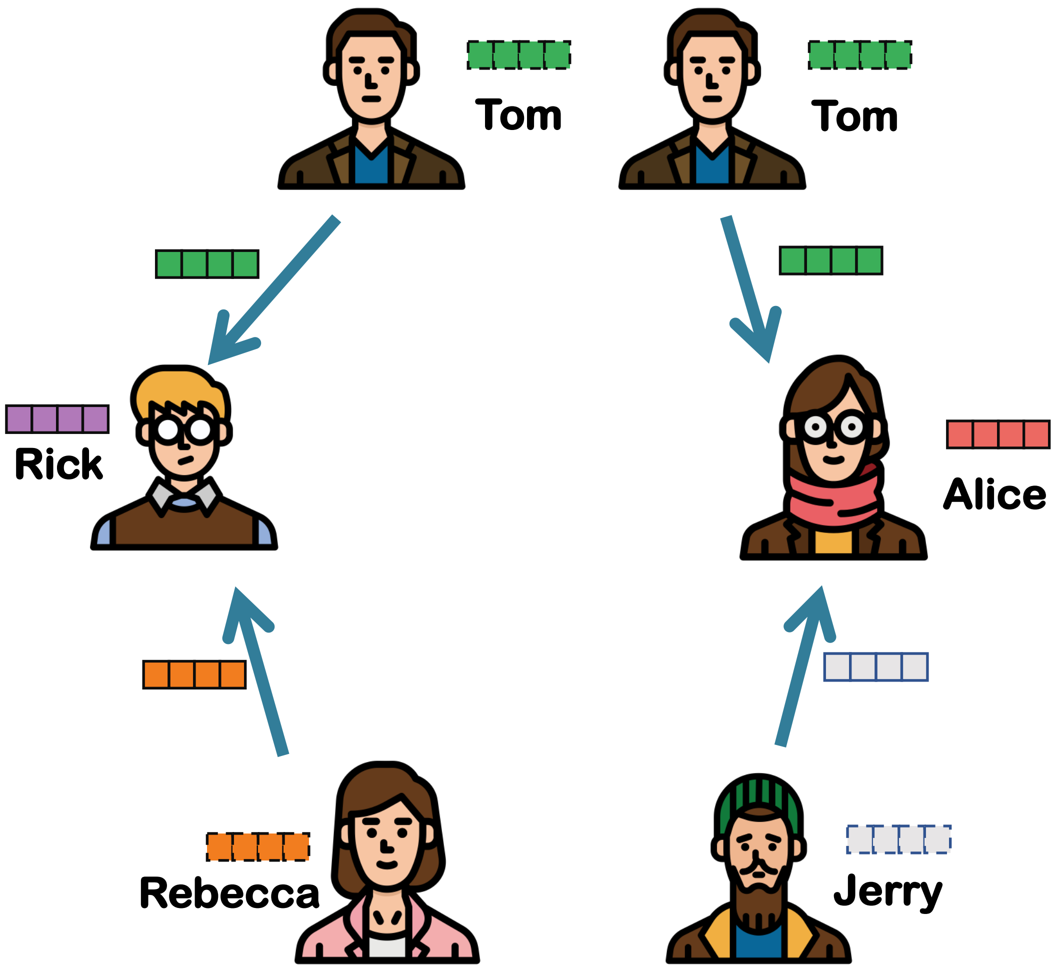


GNN inference

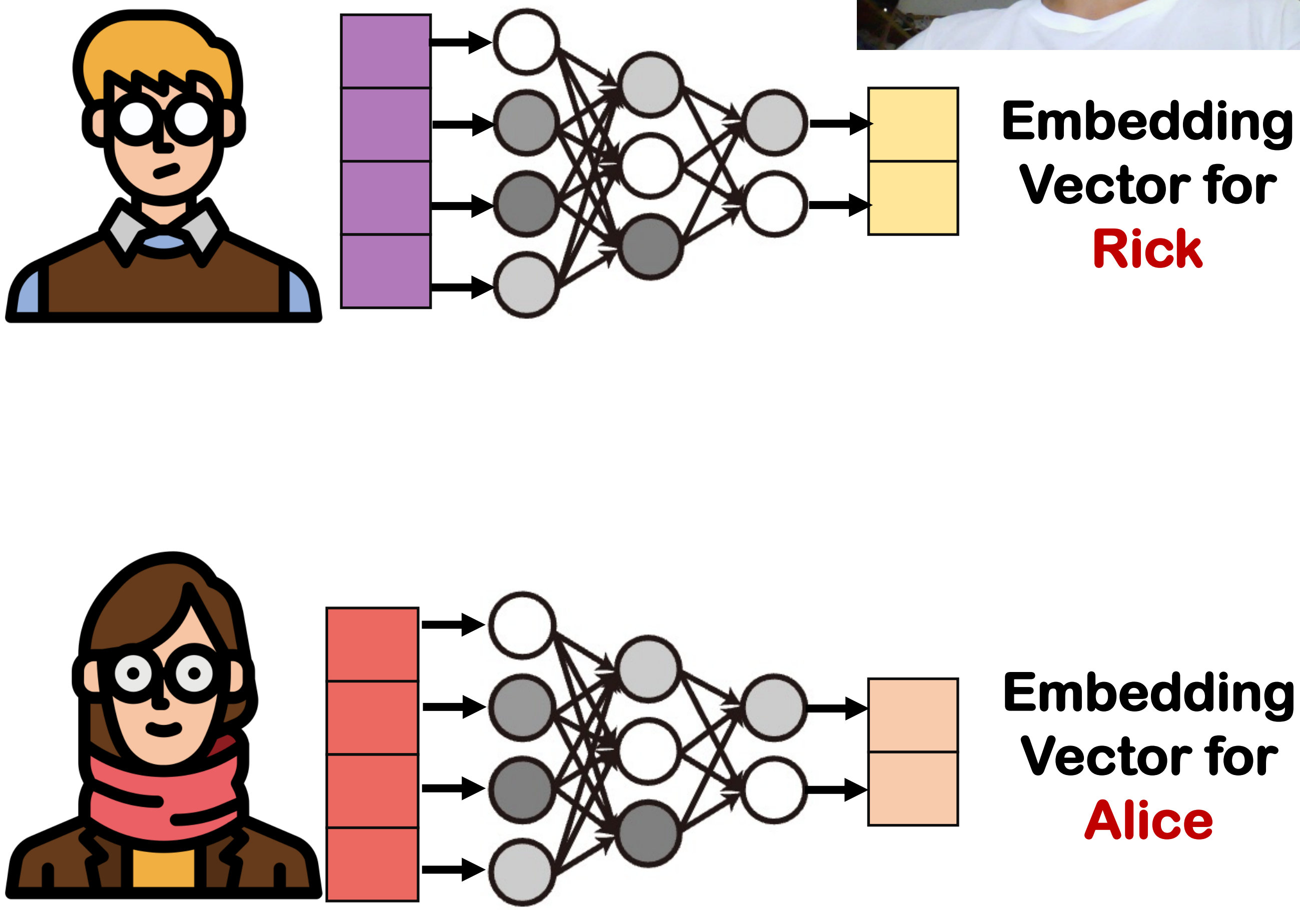
- Sample
- Aggregate
- Combine

Sampled Edge: →

Obtain Embedding Vectors with GNN - 2

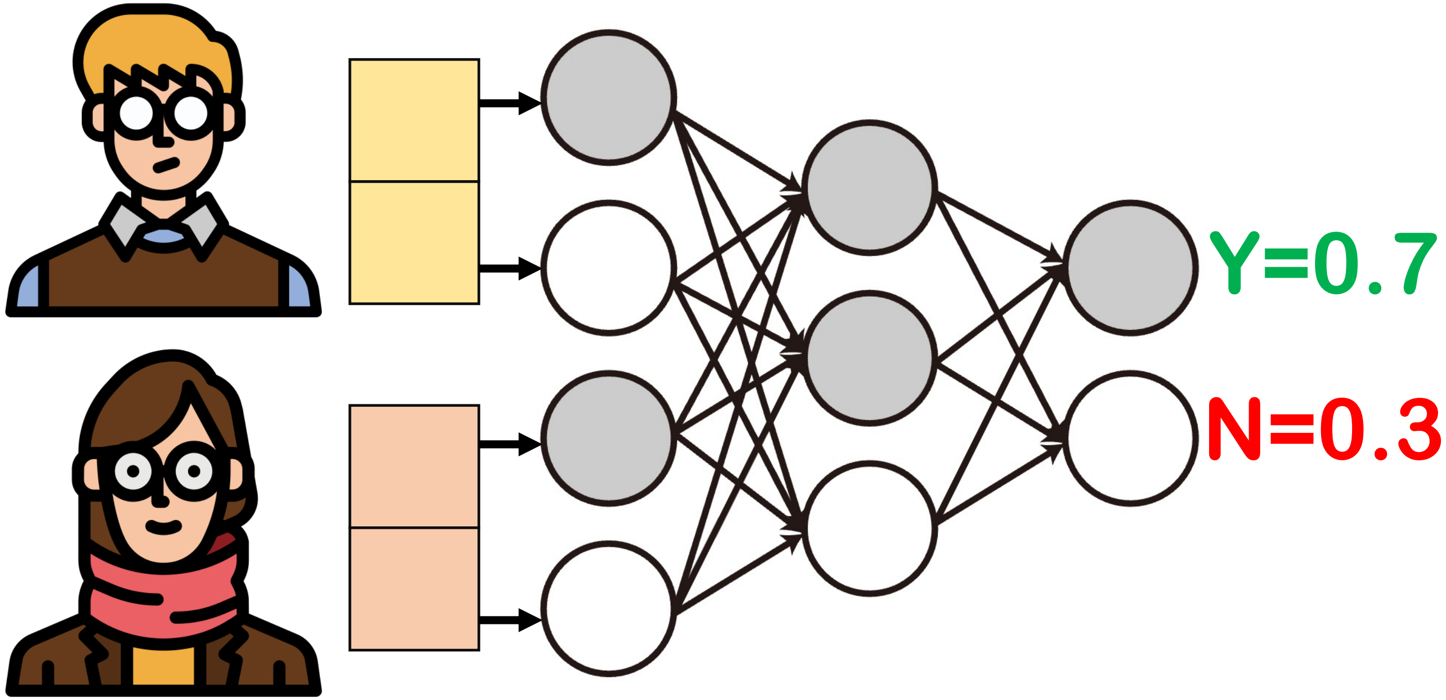


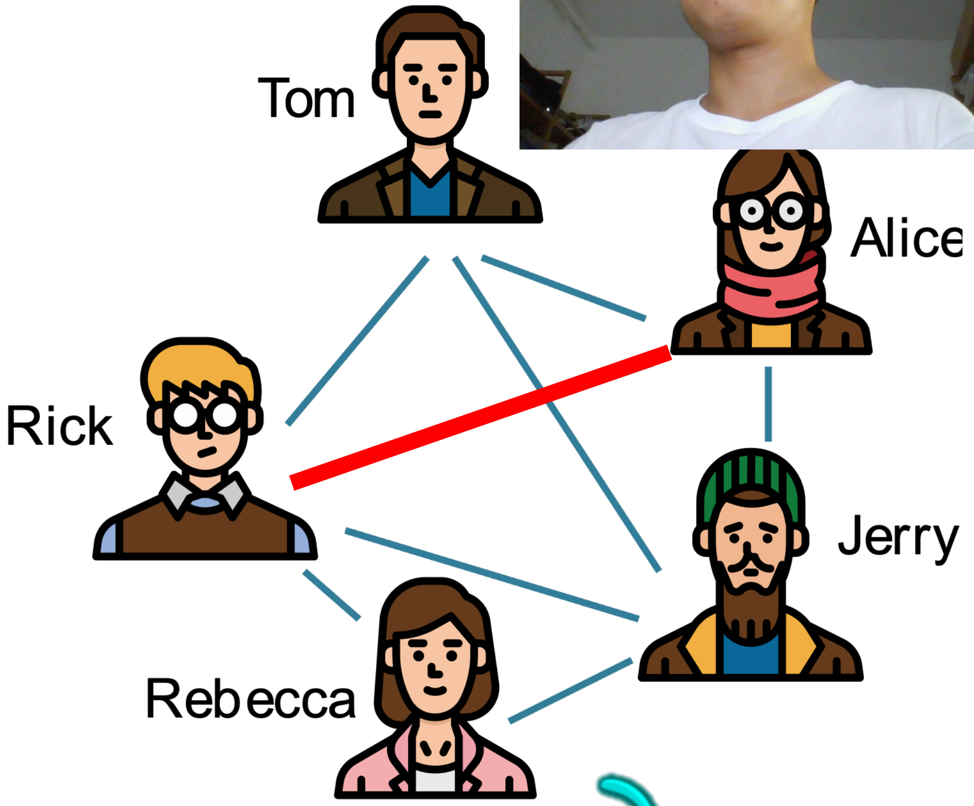
Aggregate



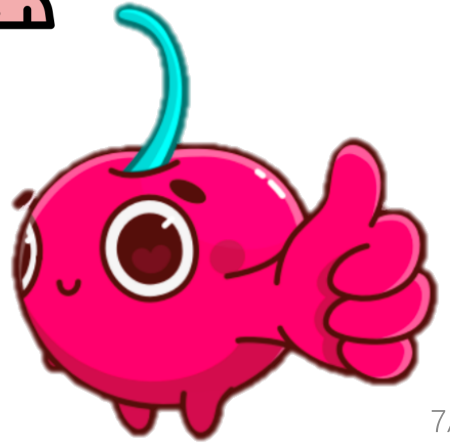
Combine

Predicting Relationship

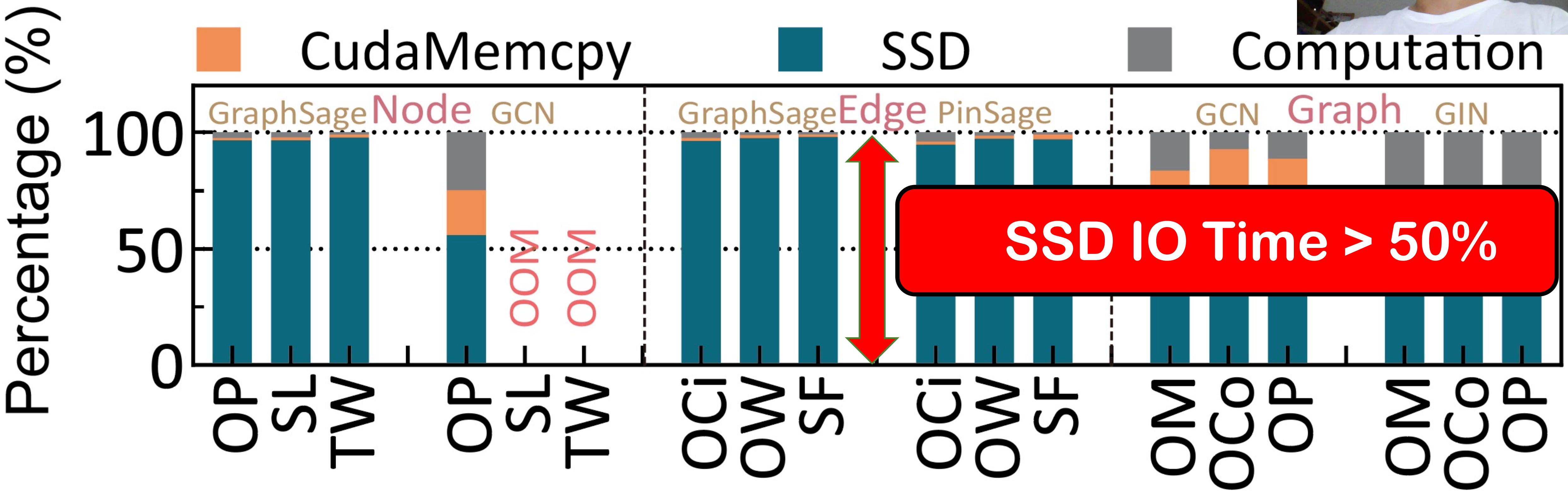




Rick and Alice may know each other !

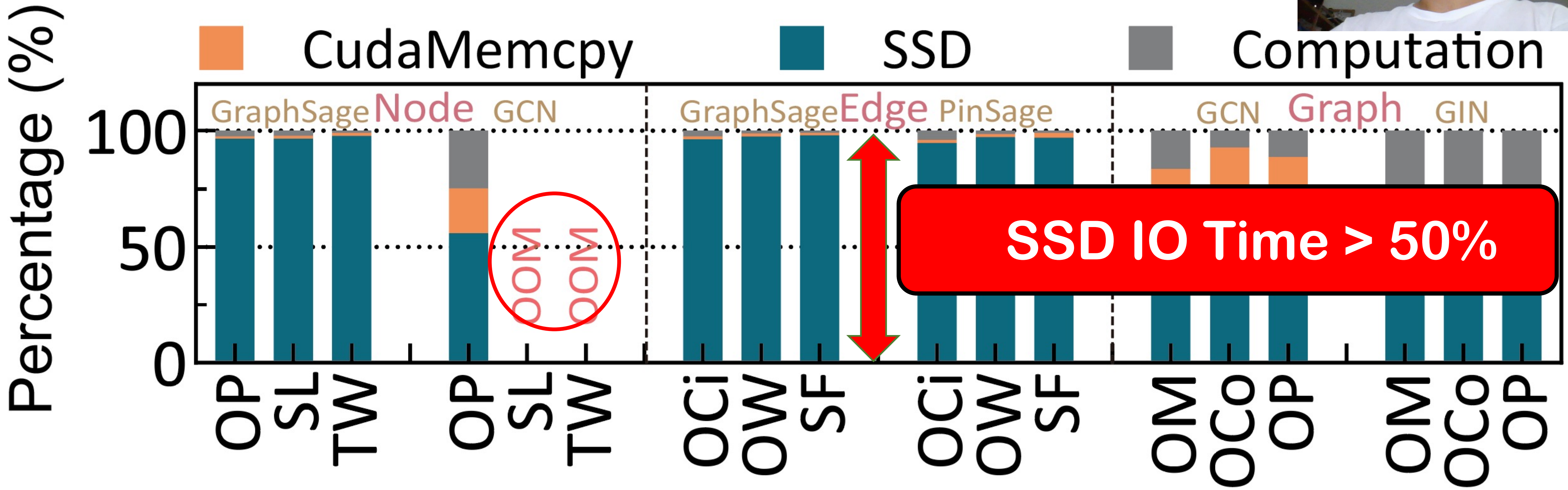


GNN-based GL Workload Characterization



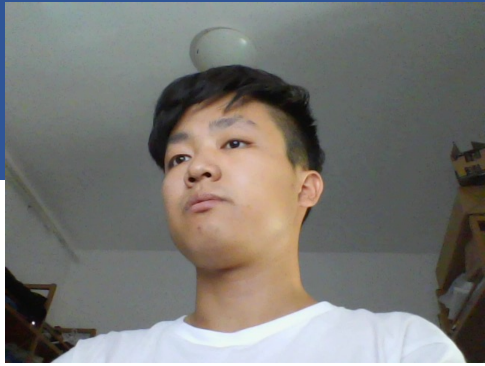
- **Graph learning workloads are bottlenecked by I/O**

GNN-based GL Workload Characterization



- Graph learning workloads are bottlenecked by I/O
- Graph learning on GPGPU is limited by memory capacity

Challenges and Solutions



- I/O bottleneck and memory constraints
- High performance graph learning
- Unstable data locality

Challenges and Solutions

- I/O bottleneck and memory constraints
- Move computation to storage

- High performance graph learning
- Use domain specific graph learning accelerator

- Unstable data locality
- Optimize data layout and schedule requests

Challenges and Solutions

- I/O bottleneck and memory constraints
- Move computation to storage
- High performance graph learning
- Use domain specific graph learning accelerator
- Unstable data locality
- Optimize data layout and schedule requests

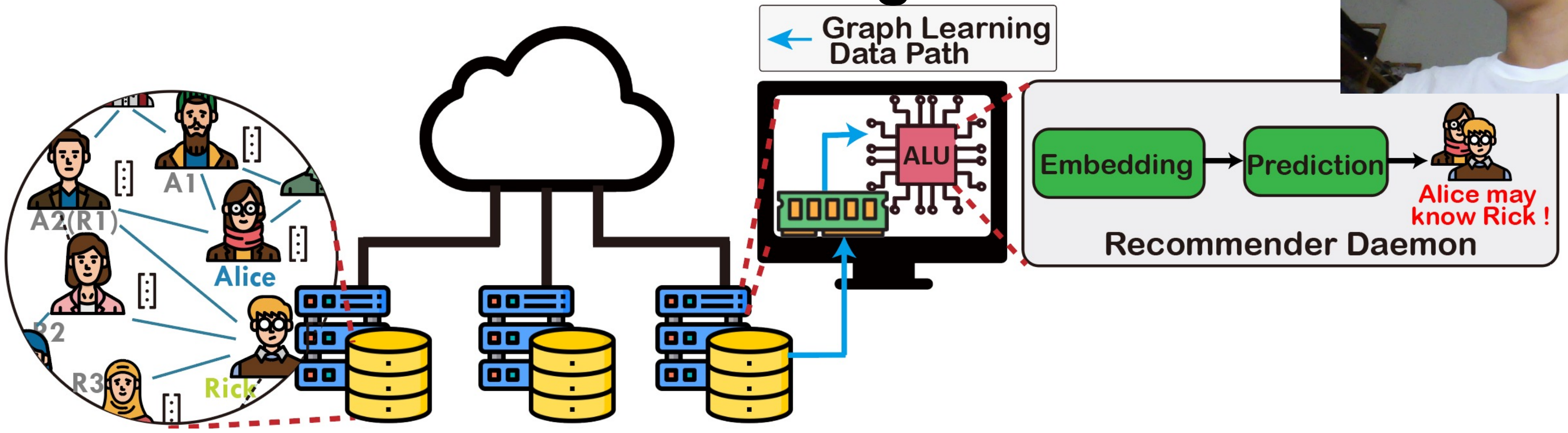
GLIST: Graph Learning In-STorage

Outline

- Background and Motivation
- GLIST Design
  - In-Storage Graph Learning Paradigm
  - System Overview
  - GLIST User Library
  - GLIST Runtime
  - In-Storage Graph Learning Accelerator
- Evaluation
- Conclusion

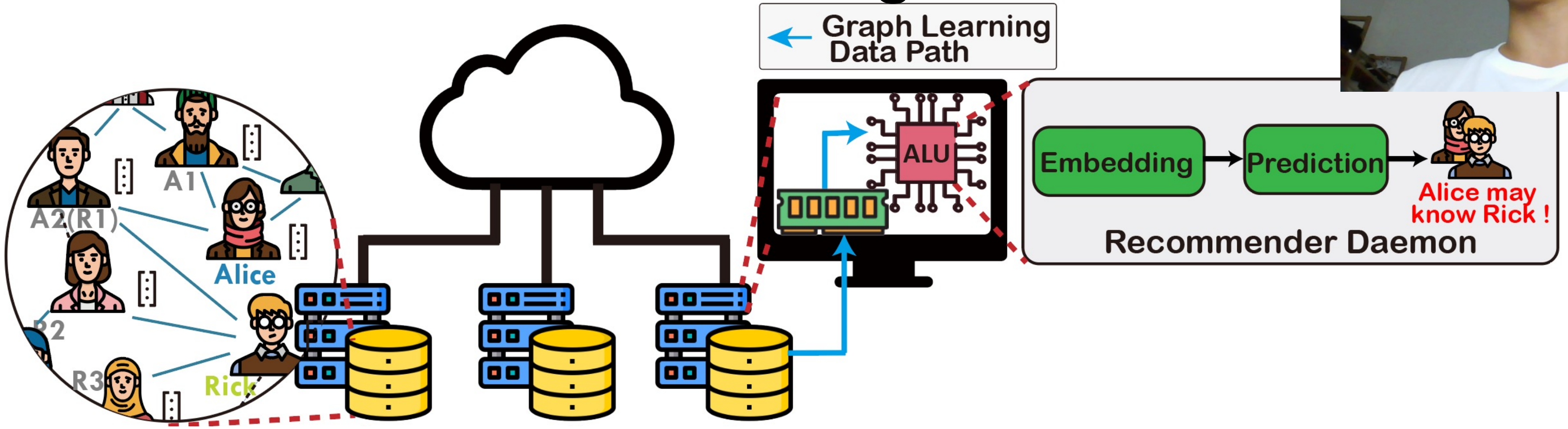
In-Storage Graph Learning Paradigm

Conventional GL Paradigm

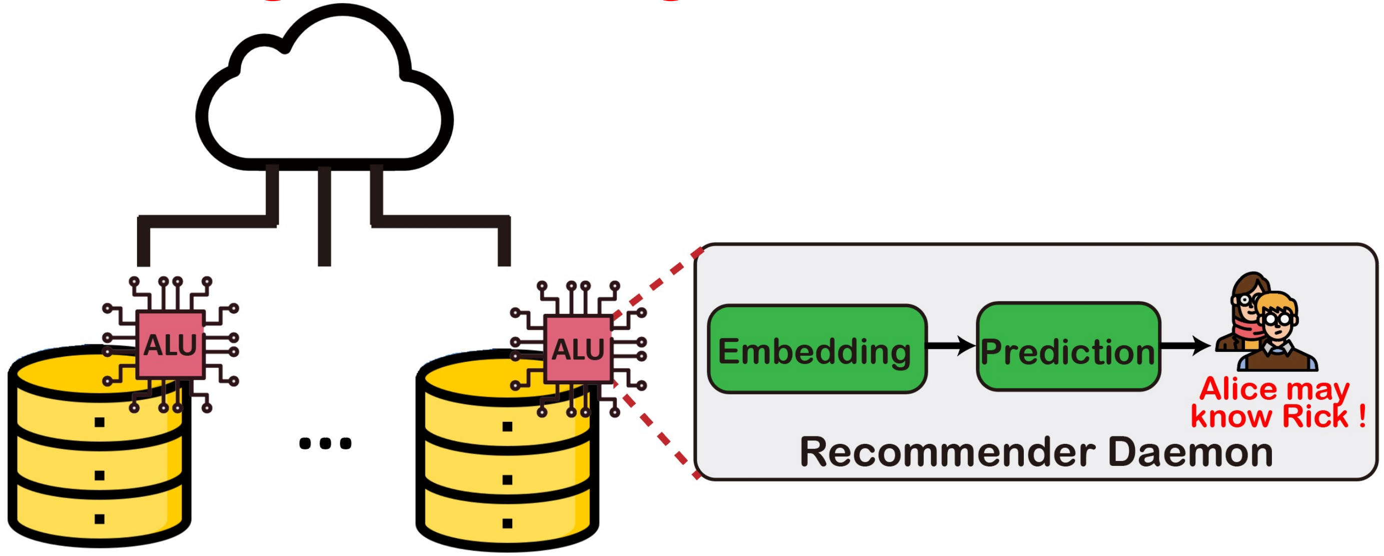


In-Storage Graph Learning Paradigm

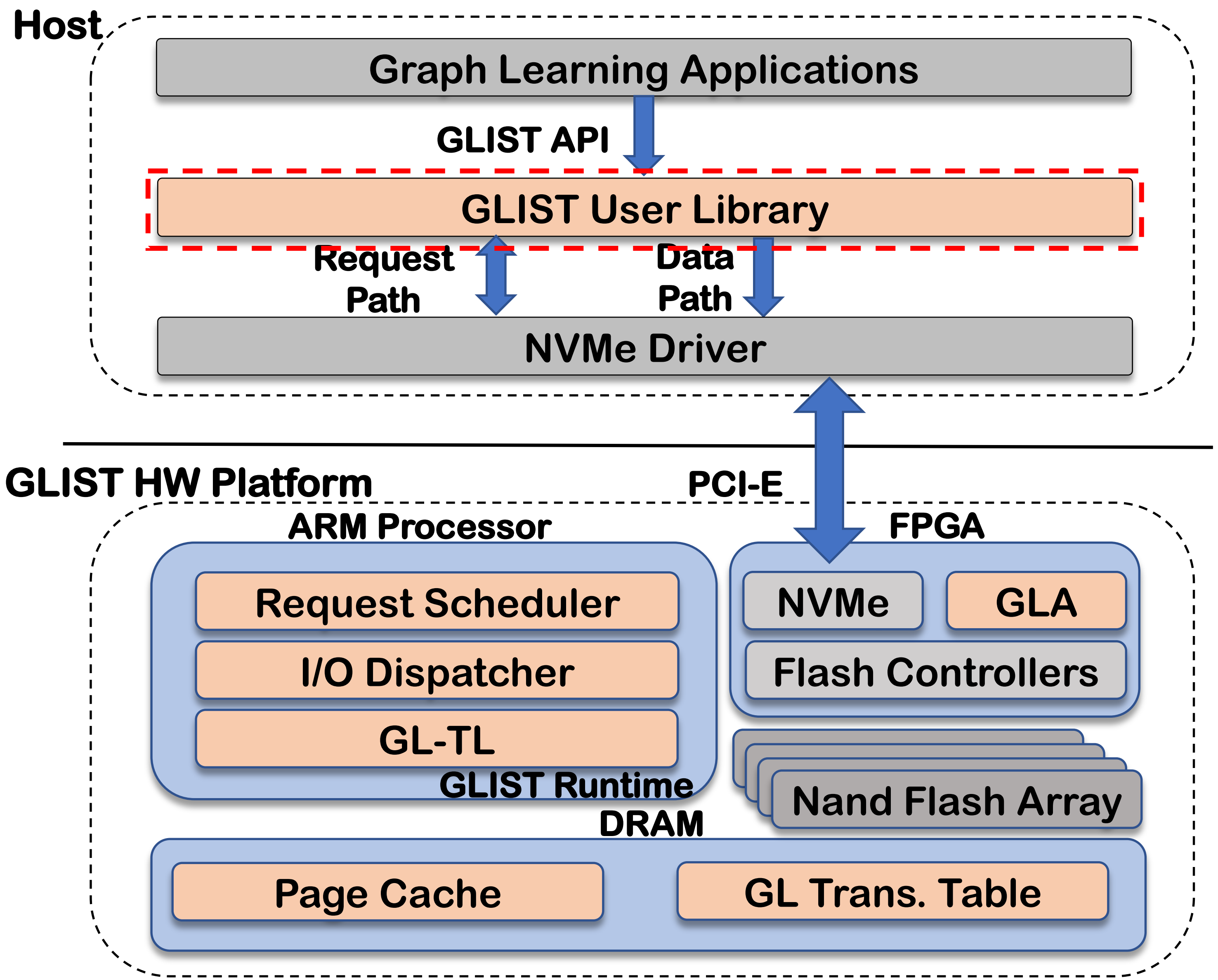
Conventional GL Paradigm



In-Storage GL Paradigm



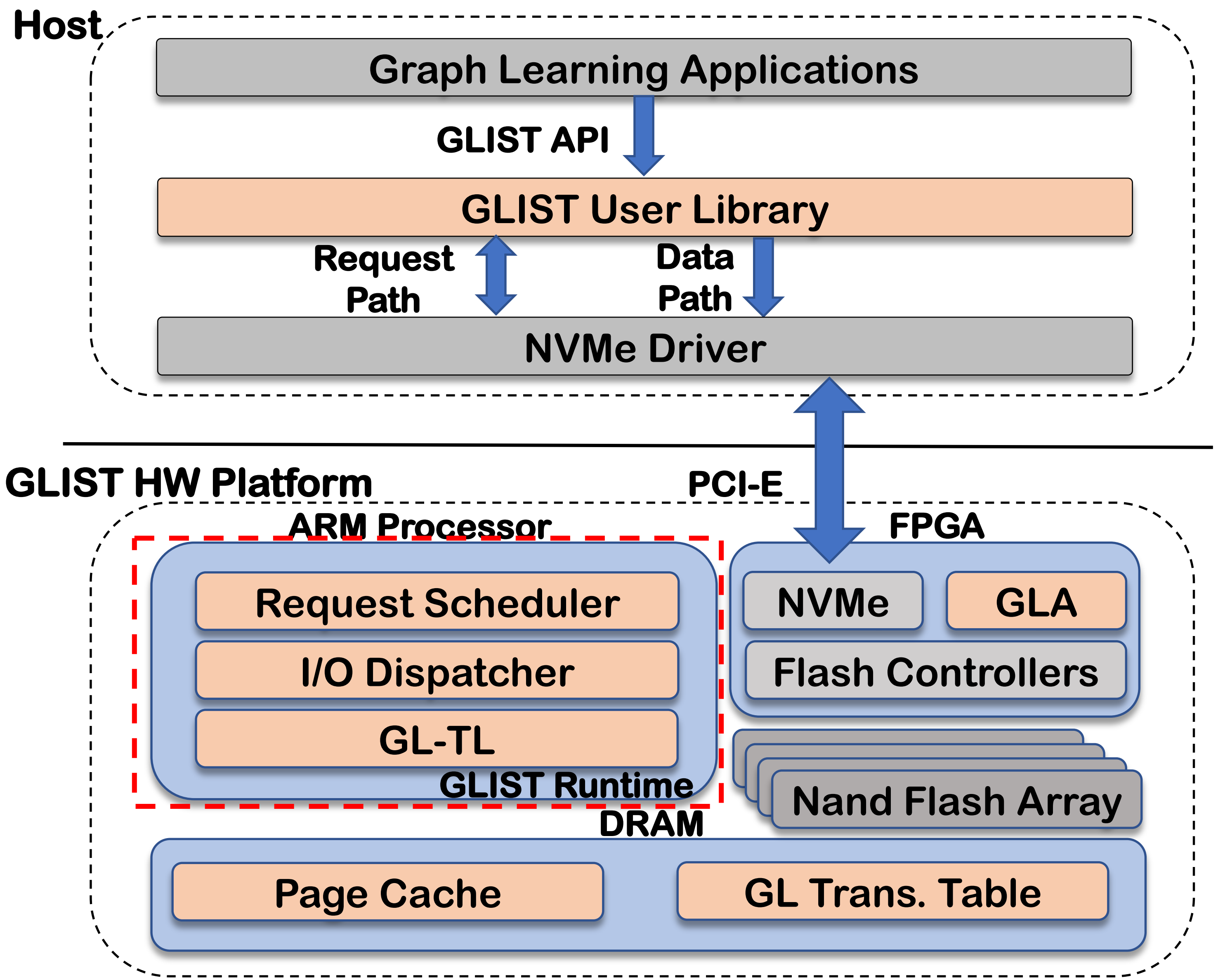
System Overview



The GLIST System

- GLIST User Library

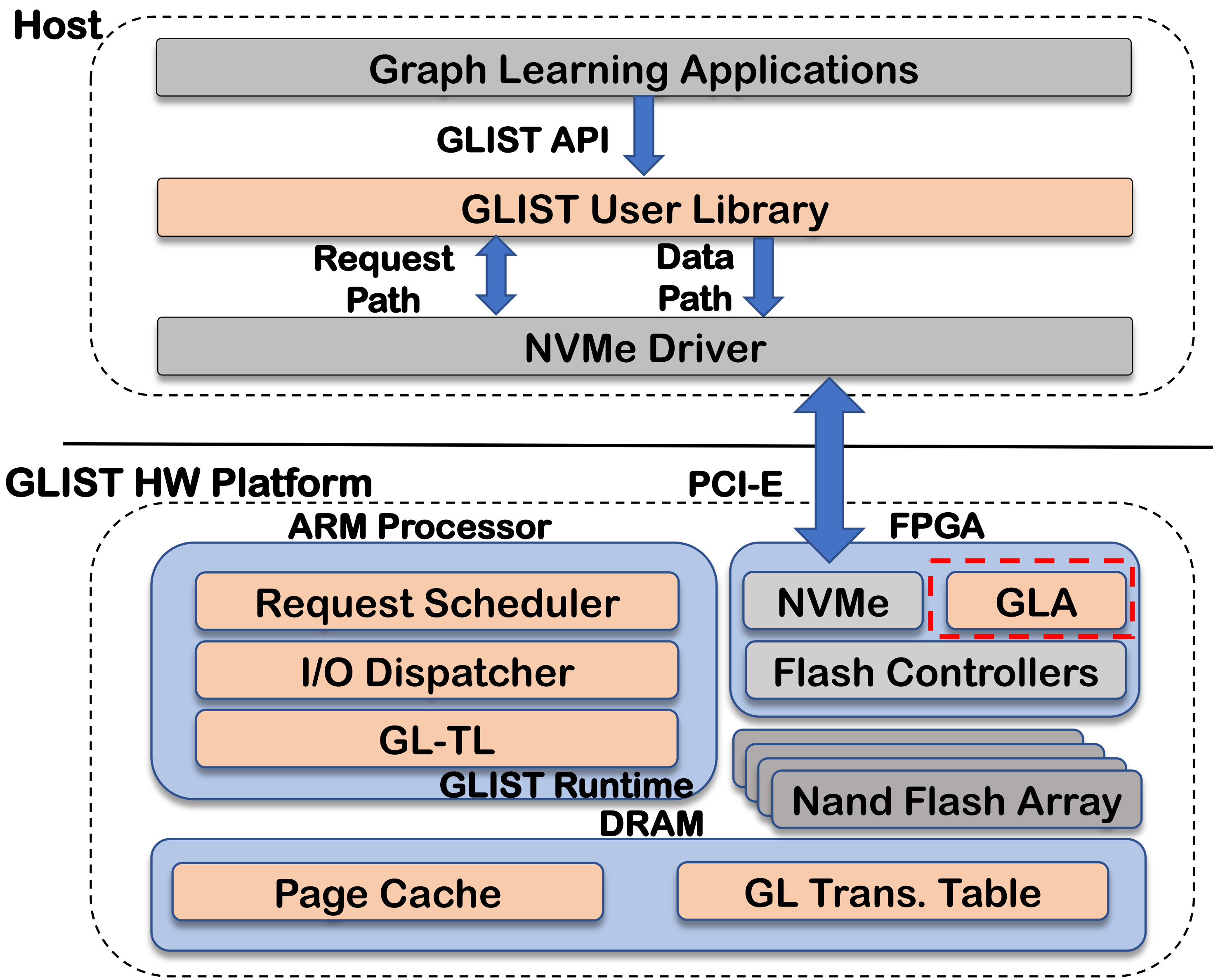
System Overview



The GLIST System

- GLIST User Library
- GLIST Runtime

System Overview



The GLIST System

- ✅ GLIST User Library
- ✅ GLIST Runtime
- ✅ In-Storage Graph Learning Accelerator

GLIST User Library

Graph Update

AddEdge(…),
RemoveEdge(…),…

Graph Registration

GraphRegister(…),
GraphUnregister(…),…

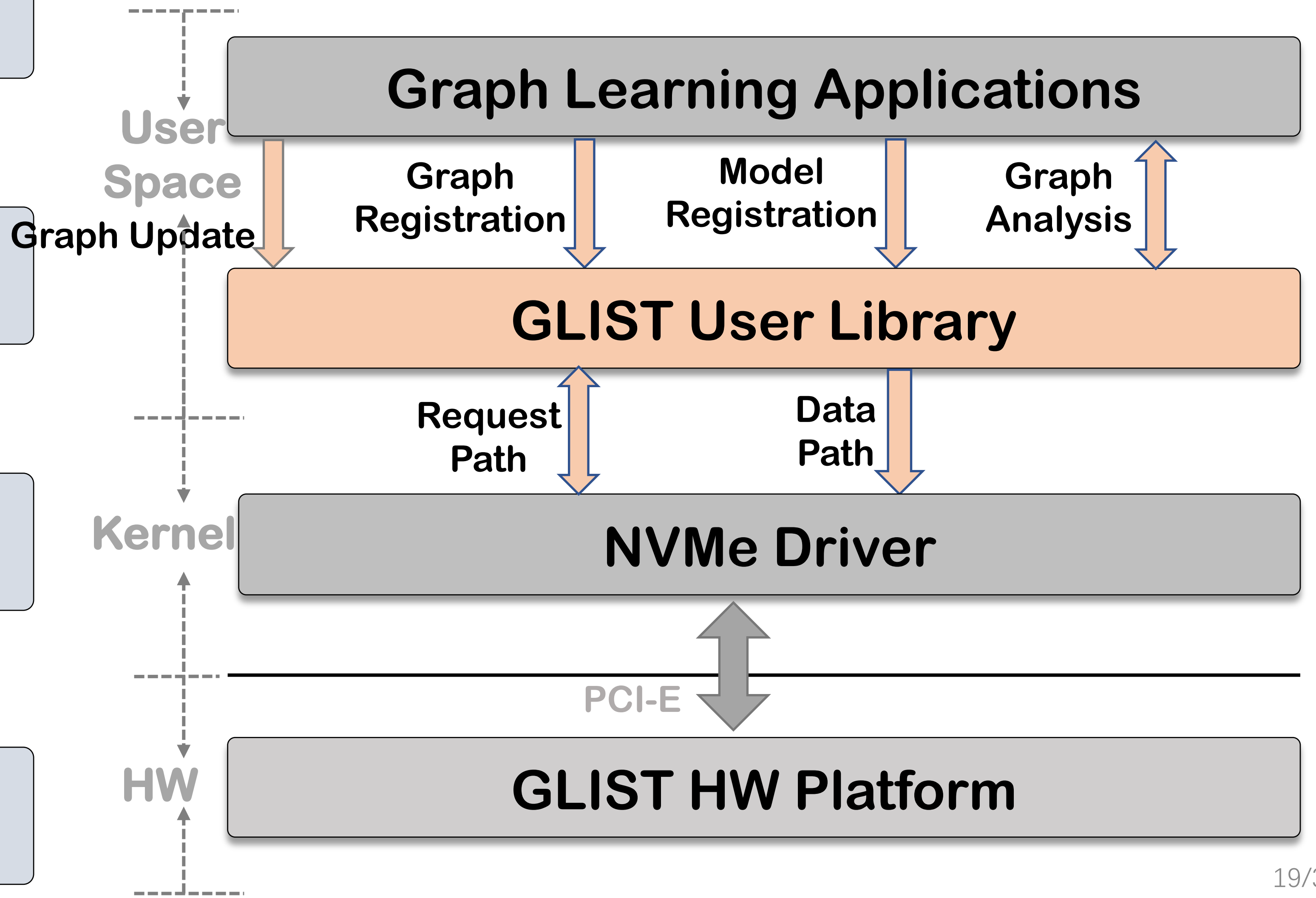
Model Registration

ModelRegister(…),
ModelUnregister(…),…

Graph Analysis

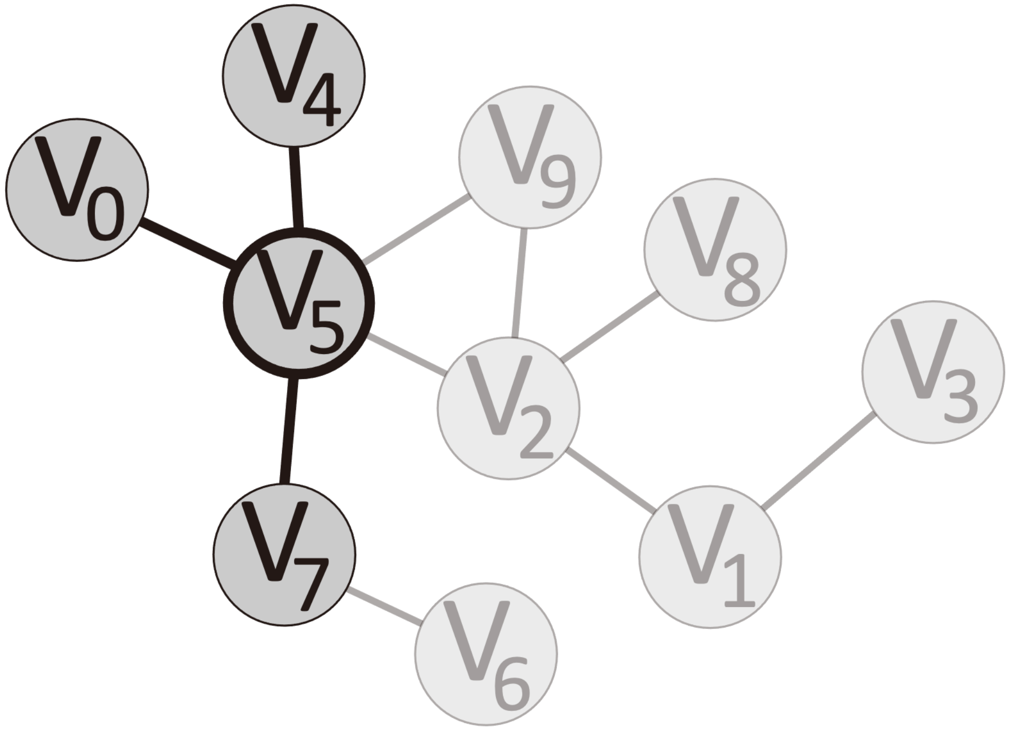
GraphAnalysis(…),
GetAnalysisResult(…),…

- **Application Interface**
- **Locality-Aware Optimization**

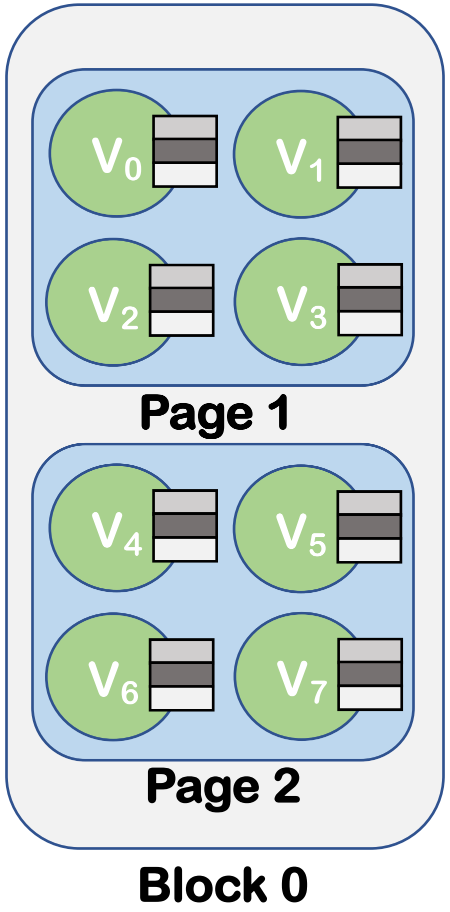


GLIST User Library – Graph Reorganization

Class of V_5?



Sample V_0, V_4, V_7 to analysis V_5



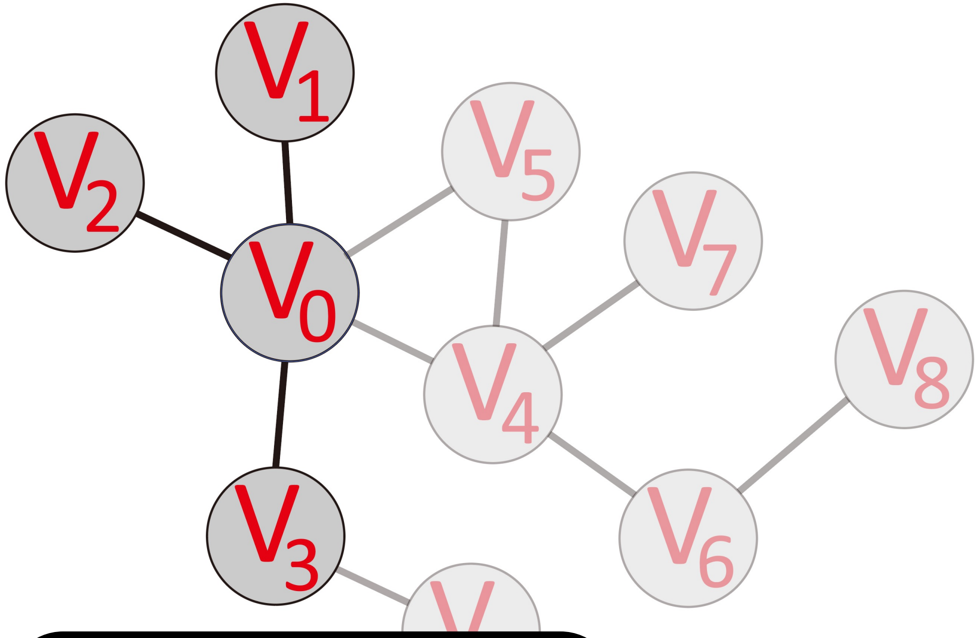
Read V_5
Read V_0
Read V_4
Read V_7

Read Page1
Read Page2

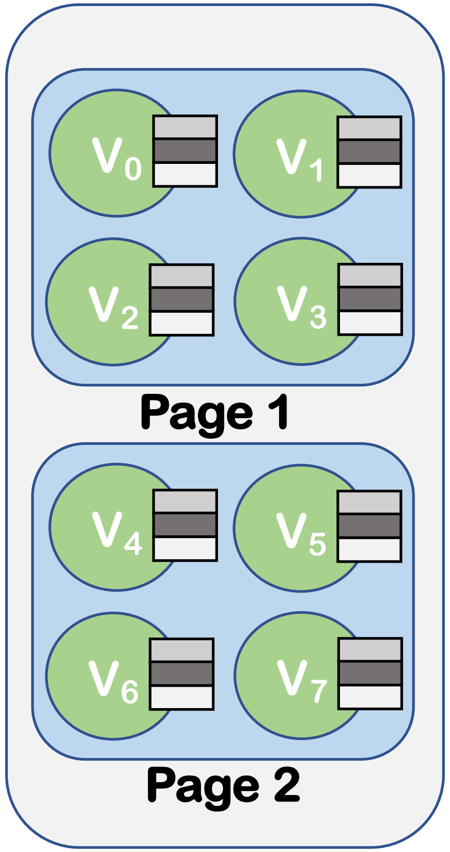
Low BW Util !

GLIST User Library – Graph Reorganization

Observation 1: Flash devices are operated at page level (16KB).
Observation 2: The size of each single property vector is usually far less than 16KB.



$V_4 \rightarrow V_1$
$V_0 \rightarrow V_2$
$V_7 \rightarrow V_3$
$V_5 \rightarrow V_0$



Block 0

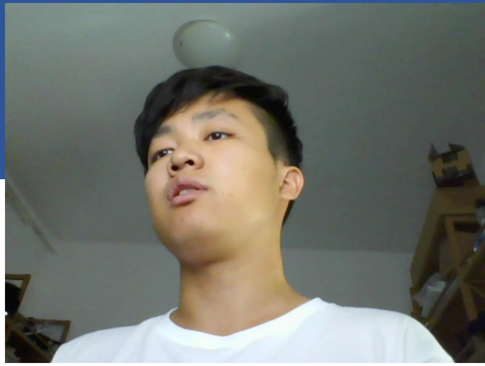
Read V_0
Read V_1
Read V_2
Read V_3

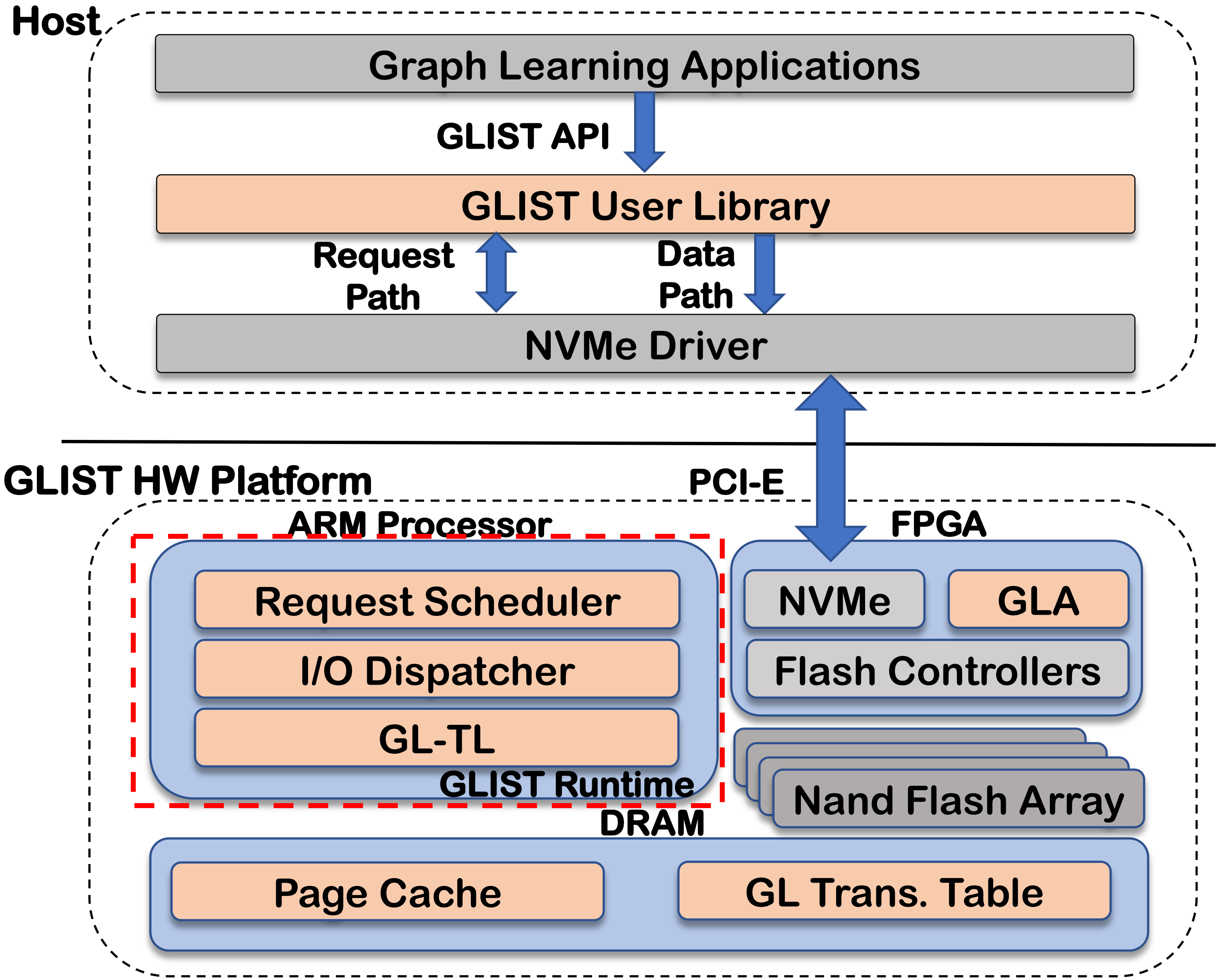
↓

Read Page1

BW Util Improved!

System Overview



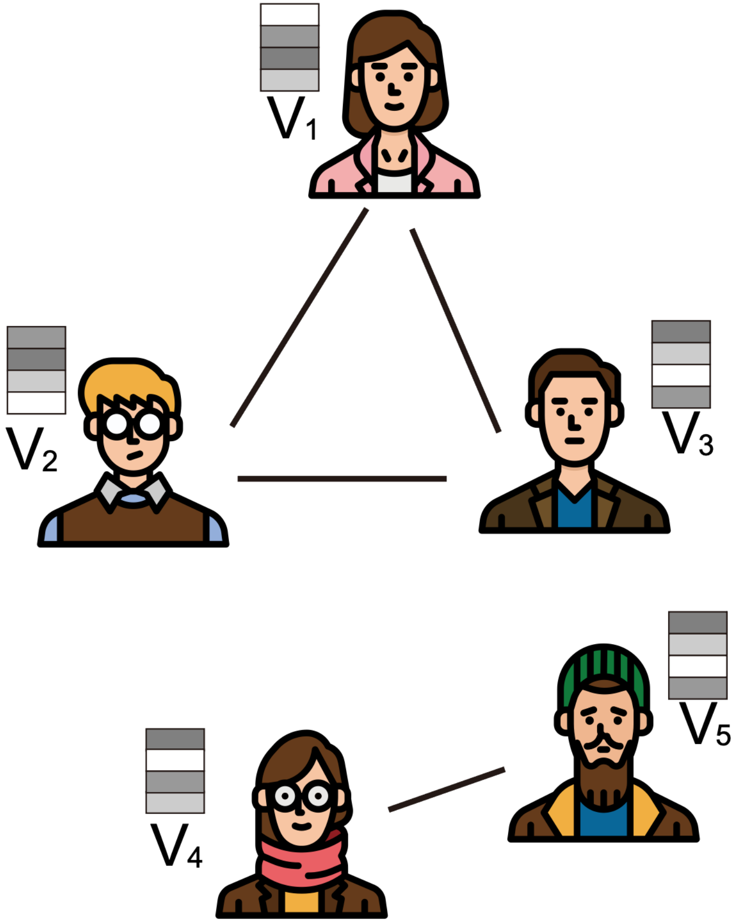


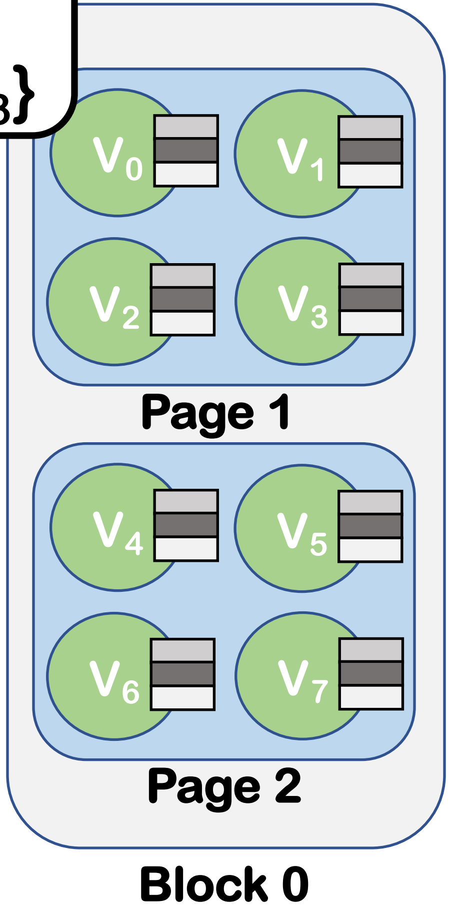
The GLIST System

- ☑ GLIST User Library
- ☑ GLIST Runtime
- ☑ In-Storage Graph Learning Accelerator

GLIST Runtime – Request Scheduling

Request 1: Analysis V_1 $\{V_1, V_2, V_3\}$
Request 2: Analysis V_4 $\{V_4, V_5\}$
Request 3: Analysis V_3 $\{V_1, V_2, V_3\}$



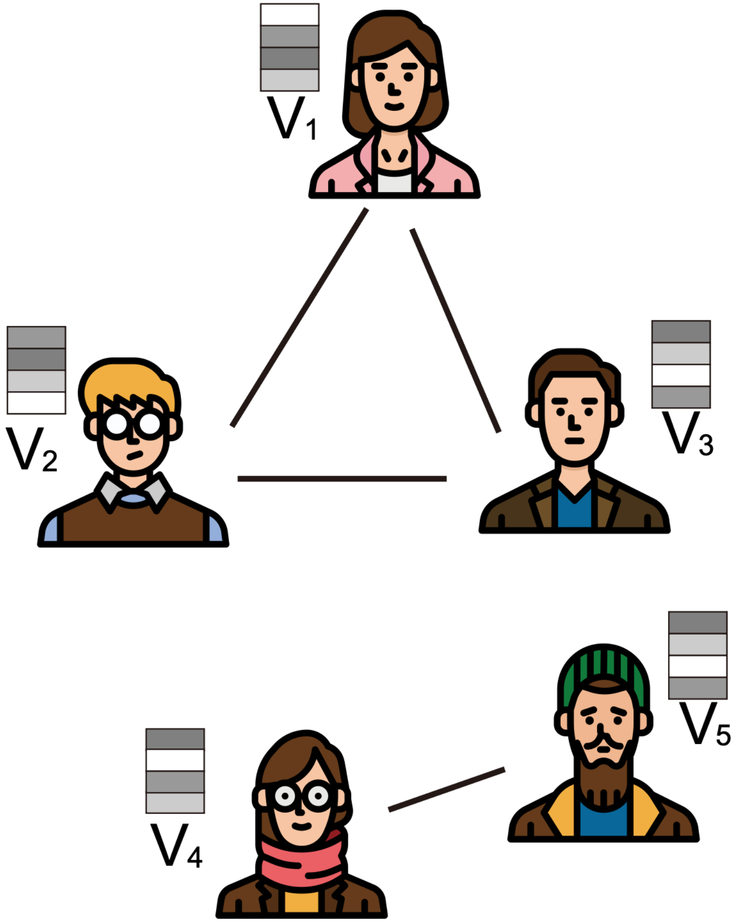


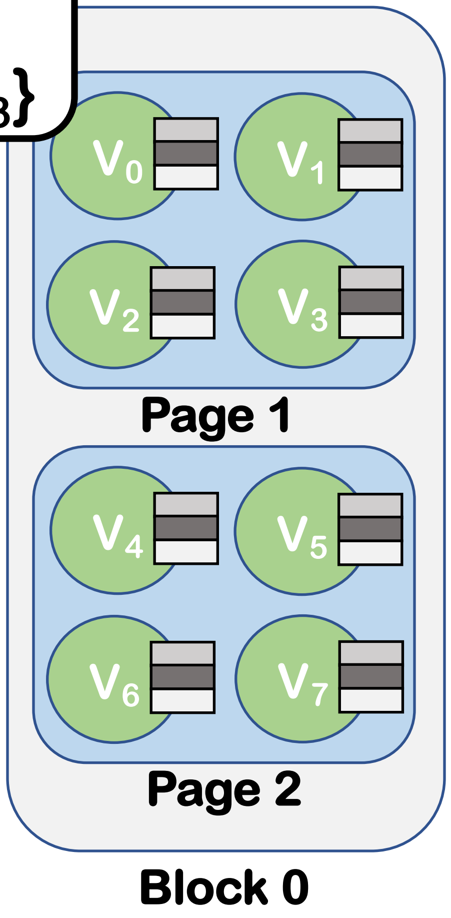
Read V_1
Read V_2
Read V_3
Read V_4
Read V_5
Read V_1
Read V_2
Read V_3

Read Page1
Read Page2
Read Page1

GLIST Runtime – Request Scheduling

Request 1: Analysis V_1 $\{V_1, V_2, V_3\}$
Request 2: Analysis V_4 $\{V_4, V_5\}$
Request 3: Analysis V_3 $\{V_1, V_2, V_3\}$





Read V_1
Read V_2
Read V_3
Read V_4
Read V_5
Read V_1
Read V_2
Read V_3

Read Page1
Read Page2
Read Page1

Schedule

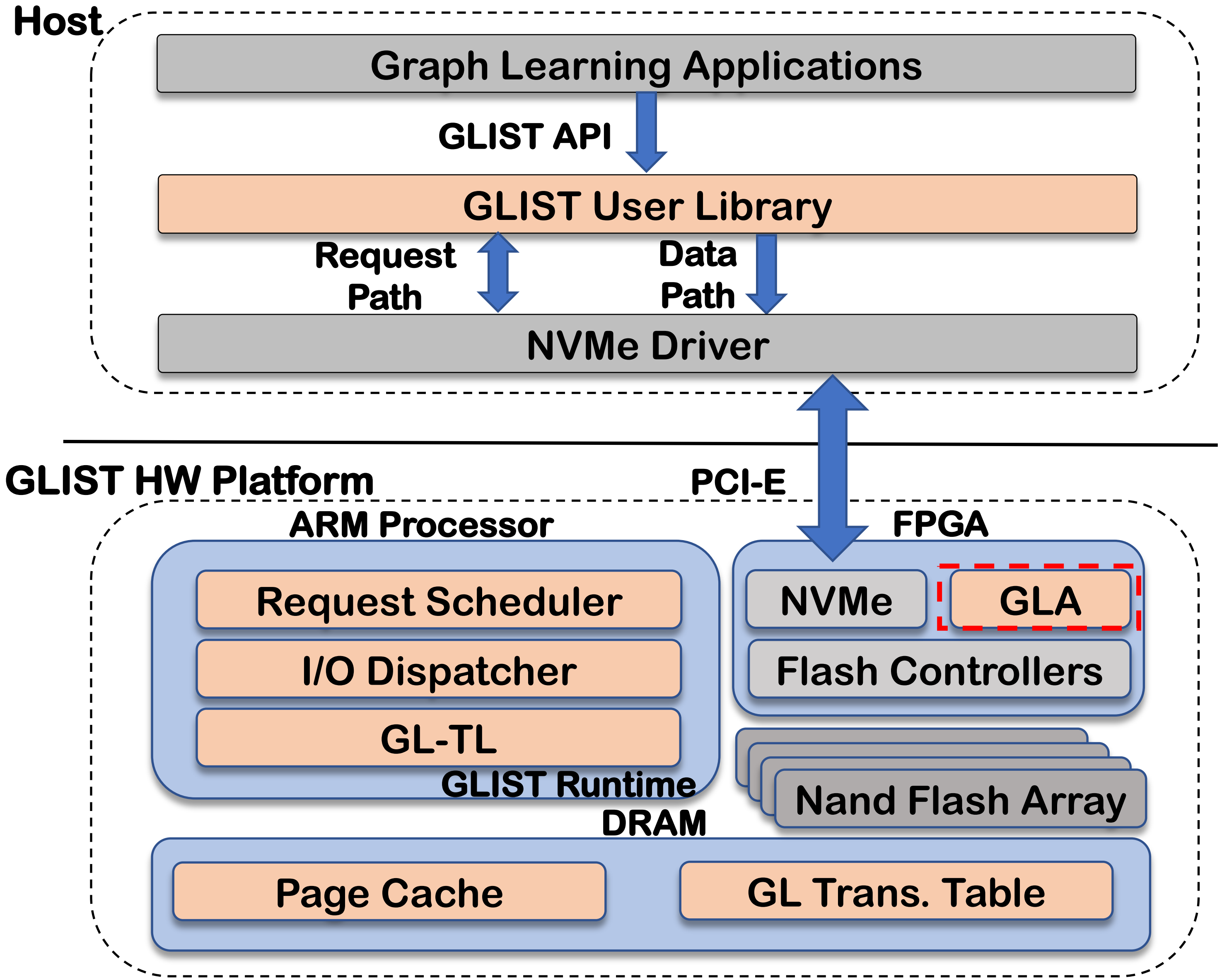
Request 1
Request 3
Request 2

Read V_1
Read V_2
Read V_3
Read V_4
Read V_5

Read Page1
Read Page2

BW Util Improved!

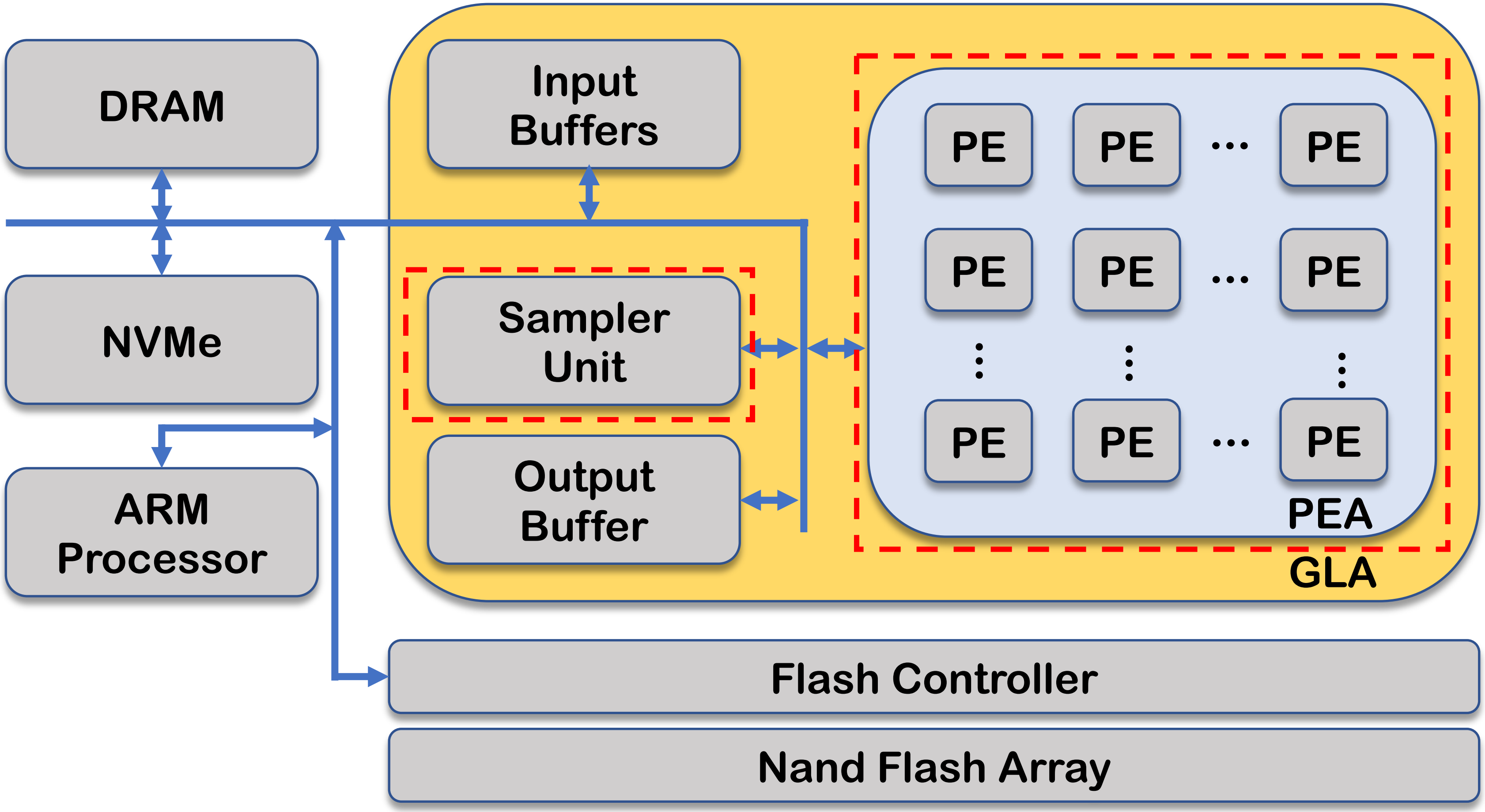
System Overview



The GLIST System

- GLIST User Library
- GLIST Runtime
- In-Storage Graph Learning Accelerator

In-Storage Graph Learning Accelerator



Sample

$$\mathbf{S}_v = \mathbf{Sample}^k(Nb(v))$$

Aggregate

$$\mathbf{h'}_v^k = \mathbf{Aggregate}(\{\mathbf{h}_u^{(k-1)}\}_{u \in N_v})$$

Combine

$$\mathbf{h}_v^k = \mathbf{Combine}(\mathbf{h'}_v^k)$$

Outline

- Background and Motivation
- GLIST Design
- Evaluation
- Conclusion

Experimental Setup

Hardware

	CPU	DRAM	SSD	GPU
CPU	2 * Xoen E5 2690v3	64GB	1TB PCIe SSD	-
V100	2 * Xoen E5 2690v3	64GB	1TB PCIe SSD	NVIDIA V100
GLIST	ARM Dual Cortex A9	1GB	1TB NAND flash	-

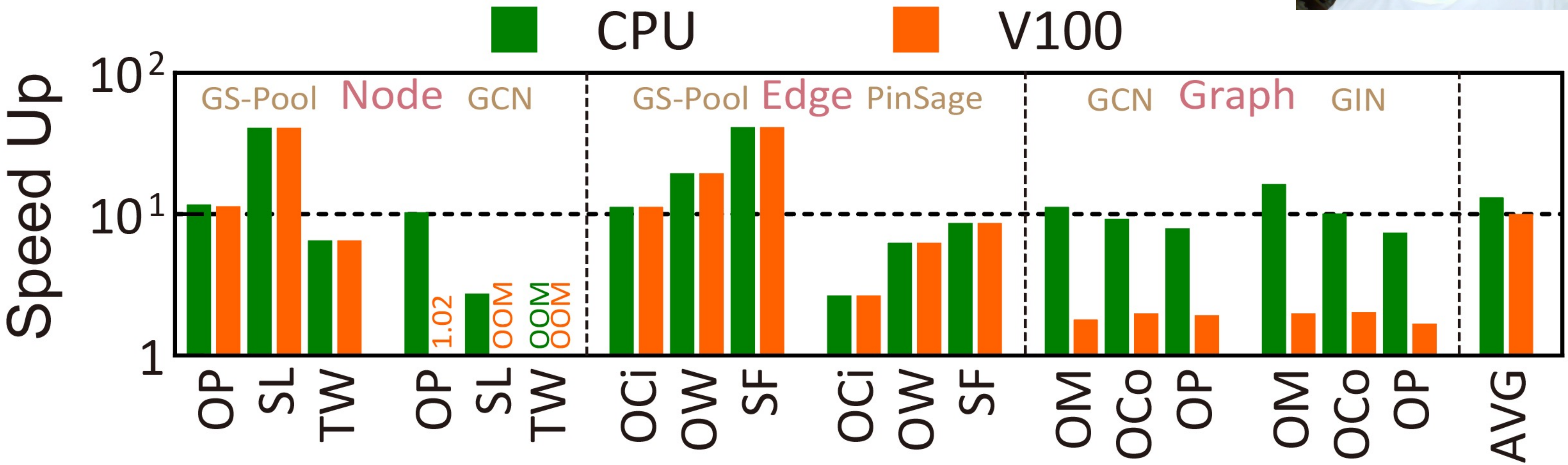
Software

Ubuntu 18.04, DGL[1]

Benchmark

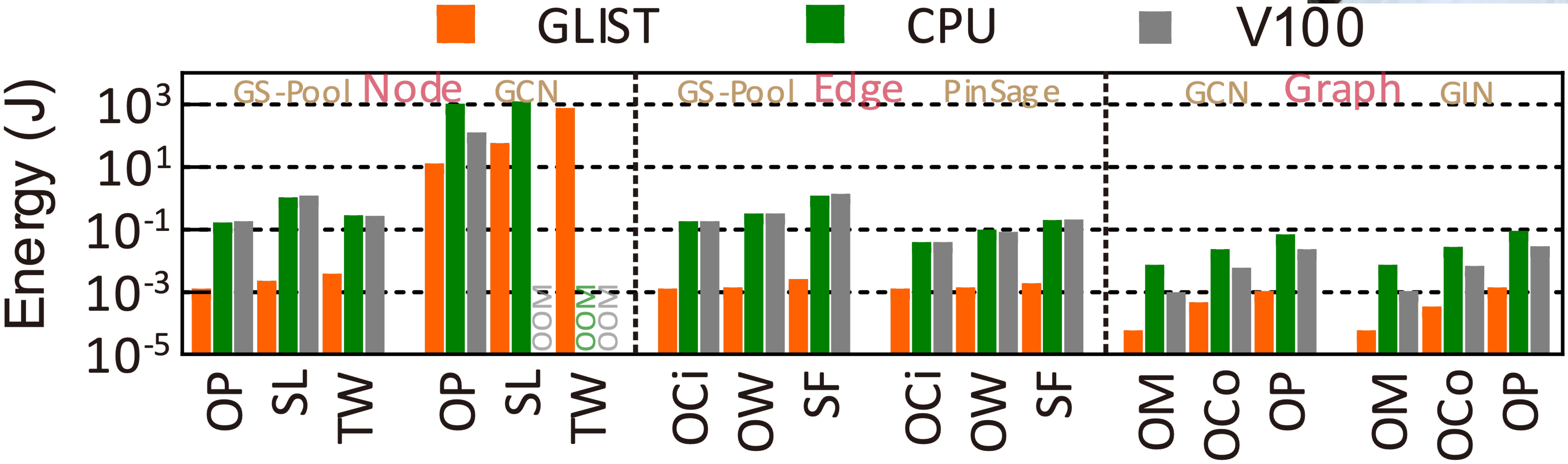
Tasks	Models	Datasets	#Vertices	#Edges(Per graph)
Node-Level Task	GCN [2] GS-Pool [3]	ogbn-products [6] soc-LiveJournal1 [7, 8] twitter [9]	2,449,029 4,847571 61,578,417	61,859,140 68,993,773 1,468,365,182
Edge-Level Task	GS-Pool [3] PinSage [4]	ogbn-papers100M [6] ogbl-citation2 [6] ogbl-wikikg2 [6] SOC-Friendster [10]	111,059,956 2,927,963 2,500,604 65,608,366	1,615,685,872 30,561,187 17,137,181 1,806,067,135
Graph-Level Task	GCN [2] GIN [5]	ogbg-molpcba [6] ogbg-code [6] ogbg-ppa [6]	437,929 452,741 158,100	28.1 124.2 2,266.1

Evaluation - GLIST Performance



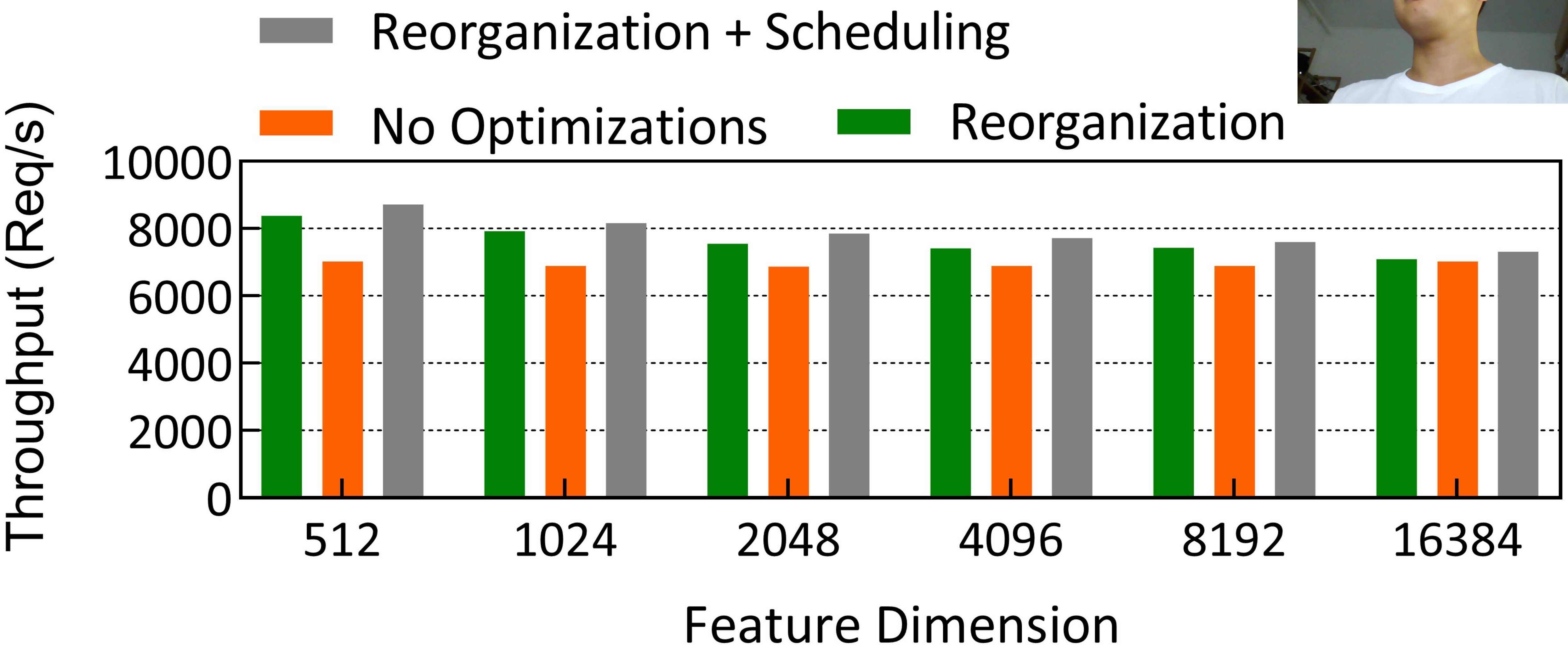
GLIST performs 13x and 10x faster on average than CPU and GPU based GL system due to well exploited data locality and high-performance GL accelerator.

Evaluation – GLIST Efficiency



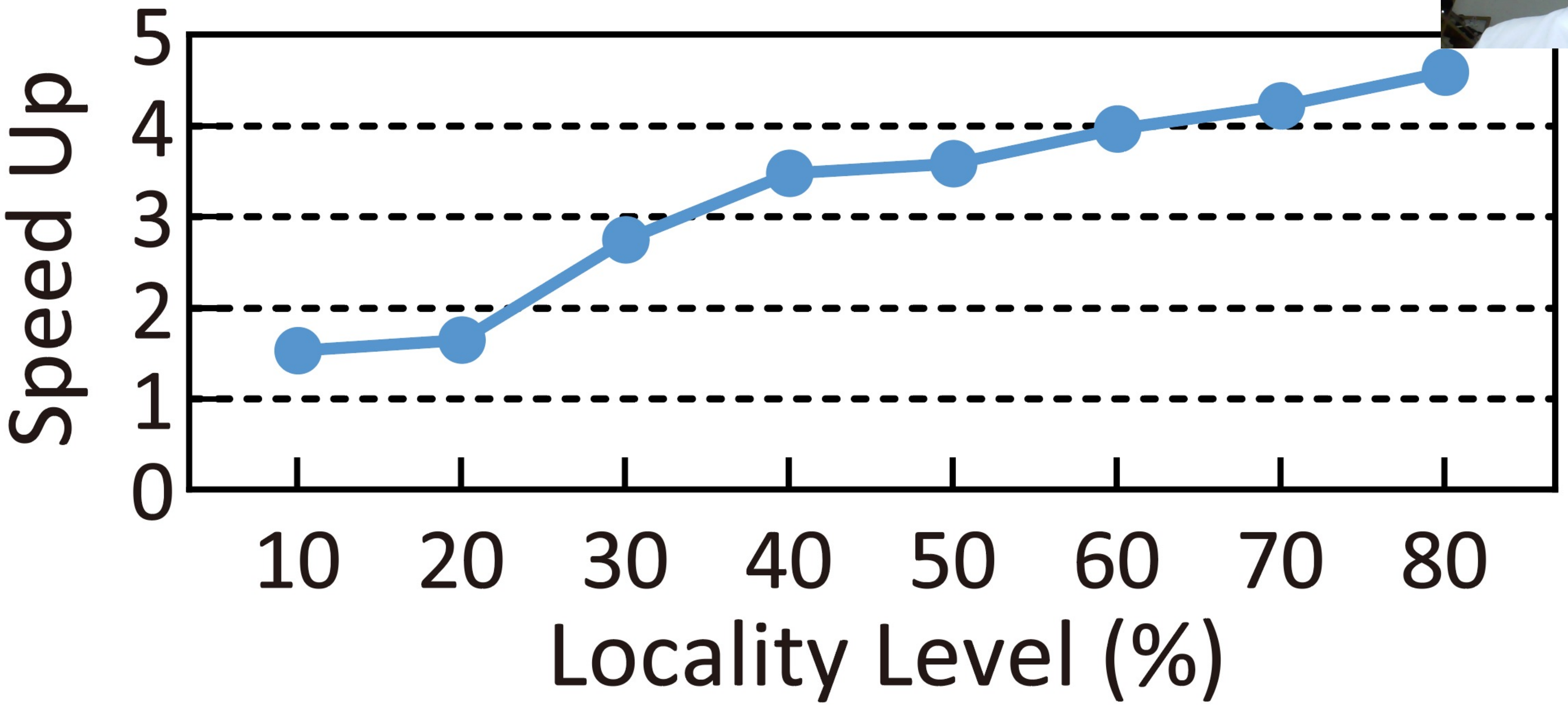
GLIST reduces average energy consumption by 98.7% and 98.0%, respectively than CPU and GPU based GL systems, and is more efficient.

Evaluation – GLIST Optimization



The optimized GLIST system benefits more from shorter property vector length and thankfully, few commonly used datasets have long feature vector [6,7,8,9,10].

Evaluation – GLIST Optimizations



The request scheduling strategy greatly exploits temporal data locality exists among GL requests.

Conclusion

- The GLIST design provides a guarantee for the high energy efficiency of the graph learning task.

- Proposed graph reorganization and request scheduling algorithm that greatly contribute to the performance and efficiency of GLIST by exploiting data locality.

- Built an FPGA-based prototype and performed various benchmarks on it.

Q&A

GLIST: Towards In-Storage Graph Learning

Cangyuan Li [1,2], Ying Wang [1,2], Cheng Liu [1,2], Shengwen Liang [1,2], Huawei Li [1,2,3], Xiaowei Li [1,2]

[1]State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing
[2]University of Chinese Academy of Sciences, Beijing
[3]Peng Cheng Laboratory, Shenzhen

中国科学院大学 University of Chinese Academy of Sciences

中国科学院计算技术研究所 INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

Email: licangyuan20@mails.ucas.ac.cn

Reference

[1] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J Smola, and Zheng Zhang. Deep graph library: Towards efficient and scalable deep learning on graphs. ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.

[2] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

[3] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Advances in neural information processing systems, pages 1024–1034, 2017.

[4] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACMSIGKDD International Conference on Knowledge Discovery & Data Mining, pages 974–983, 2018.

[5] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826, 2018.

[6] 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. arXiv preprint arXiv:2005.00687, 2020.

[7] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 44–54,2006.

[8] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.

stanford.edu/data, June 2014.

[9] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In WWW ’10: Proceedings of the 19th international conference on World wide web, pages 591–600, New York, NY, USA, 2010. ACM.

[10] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. Knowledge and Information Systems, 42(1):181–213, 20