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
Outline

• Background and Motivation
• GLIST Design
• Evaluation
• Conclusion
Background of Graph Learning

- Social Network
- Molecular structures
- Knowledge Graph
- Road Map

Graph Data is Everywhere!

User

Relation

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

Embedding Vector for Rick

Embedding Vector for Alice
Predicting Relationship

Rick and Alice may know each other!

Y = 0.7

N = 0.3

Rick and Alice may know each other!
GNN-based GL Workload Characterization

- Graph learning workloads are bottlenecked by I/O

SSD IO Time > 50%
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

Embedding → Prediction

Alice may know Rick!
In-Storage Graph Learning Paradigm

Conventional GL Paradigm

In-Storage GL Paradigm

Embedding → Prediction → Alice may know Rick!

 recommender Daemon

Embedding → Prediction → Alice may know Rick!

 recommender Daemon
System Overview

The GLIST System

GLIST User Library
System Overview

The GLIST System

- GLIST User Library
- GLIST Runtime

Graph Learning Applications

GLIST API

GLIST User Library

Request Path

Data Path

NVMe Driver

NVMe

GLA

Flash Controllers

Nand Flash Array

GLIST Runtime

Request Scheduler

I/O Dispatcher

GL-TL

GLIST Runtime

DRAM

Page Cache

GL Trans. Table

Host

Graph Learning Applications

GLIST API

GLIST User Library

Request Path

Data Path

NVMe Driver

GLIST HW Platform

ARM Processor

PCI-E

FPGA
System Overview

The GLIST System

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

**Graph Learning Applications**

- GLIST API
- GLIST User Library
- Request Path
- Data Path
- NVMe Driver

**GLIST HW Platform**

- ARM Processor
- PCI-E
- FPGA
- Request Scheduler
- I/O Dispatcher
- GL-TL
- GLIST Runtime
- Nand Flash Array
- DRAM
- Page Cache
- GL Trans. Table
- NVMe
- GLA
- Flash Controllers
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 $V_5$
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.
System Overview

Graph Learning Applications

GLIST API

GLIST User Library

Request Path

Data Path

NVMe Driver

The GLIST System

GLIST User Library

GLIST Runtime

In-Storage Graph Learning Accelerator

Host

Graph Learning Applications

GLIST API

GLIST User Library

Request Path

Data Path

NVMe Driver

GLIST HW Platform

PCI-E

FPGA

ARM Processor

Request Scheduler

I/O Dispatcher

GL-TL

GLIST Runtime

DRAM

Page Cache

GL Trans. Table

NAND Flash Array

NVMe

GLA

Flash Controllers
Request 1: Analysis \( V_1 \{V_1, V_2, V_3\} \)
Request 2: Analysis \( V_4 \{V_4, V_5\} \)
Request 3: Analysis \( V_3 \{V_1, V_2, V_3\} \)

Page 1

- \( V_0 \)
- \( V_1 \)
- \( V_2 \)
- \( V_3 \)

Page 2

- \( V_4 \)
- \( V_5 \)
- \( V_6 \)
- \( V_7 \)

Block 0

- \( V_2 \)
- \( V_3 \)

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 Page 1
- Read Page 2
- Read Page 1

Schedule
- Request 1
- Request 3
- Request 2

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

Read Page 1
Read Page 2

BW Util Improved!
System Overview

The GLIST System

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

Sample
\[ S_v = \text{Sample}^k(Nb(v)) \]

Aggregate
\[ h^{(k)}_v = \text{Aggregate}(\{h^{(k-1)}_u\}_{u \in N_v}) \]

Combine
\[ h^{(k)}_v = \text{Combine}(h^{(k)}_v) \]
Outline

• Background and Motivation
• GLIST Design
• Evaluation
• Conclusion
## Experimental Setup

### Hardware

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>DRAM</th>
<th>SSD</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>2 * Xeon E5 2690v3</td>
<td>64GB</td>
<td>1TB PCIe SSD</td>
<td>-</td>
</tr>
<tr>
<td>V100</td>
<td>2 * Xeon E5 2690v3</td>
<td>64GB</td>
<td>1TB PCIe SSD</td>
<td>NVIDIA V100</td>
</tr>
<tr>
<td>GLIST</td>
<td>ARM Dual Cortex A9</td>
<td>1GB</td>
<td>1TB NAND flash</td>
<td>-</td>
</tr>
</tbody>
</table>

### Software

- Ubuntu 18.04, DGL[1]

### Benchmark

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Models</th>
<th>Datasets</th>
<th>#Vertices</th>
<th>#Edges(Per graph)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>twitter [9]</td>
<td>61,578,417</td>
<td>1,468,365,182</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ogbl-wikikg2 [6]</td>
<td>2,500,604</td>
<td>17,137,181</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOCLiveJournal1 [7, 8]</td>
<td>65,608,366</td>
<td>1,806,067,135</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ogbg-ppa [6]</td>
<td>158,100</td>
<td>2,266.1</td>
</tr>
</tbody>
</table>
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.
GLIST reduces average energy consumption by 98.7% and 98.0%, respectively than CPU and GPU based GL systems, and is more efficient.
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].
The request scheduling strategy greatly exploits temporal data locality exists among GL requests.
• 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.
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

Email: licangyuan20@mails.ucas.ac.cn


