Peridot: Accelerating Out-of-Core GCN Data Reuse Pattern and Co-Design on GPU

Shakya Jayakody, Jun Wang

Department of Electrical and
Computer Engineering

MLArchSys 2024

UCF UNIVERSITY OF CENTRAL FLORIDA

Research problem

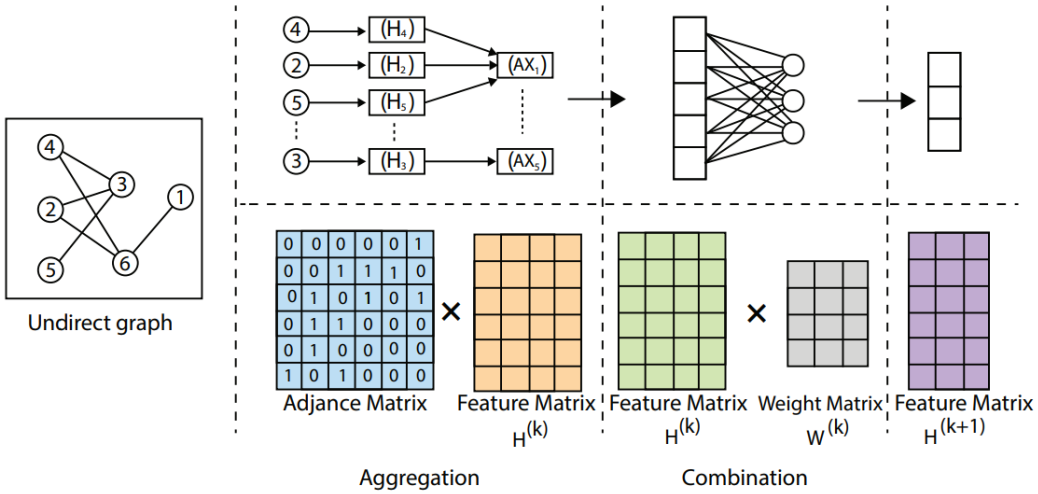


Fig. 1. Chain Matrix Multiplication in a Graph Convolutional Network Layer During the Aggregation and Combination Phases

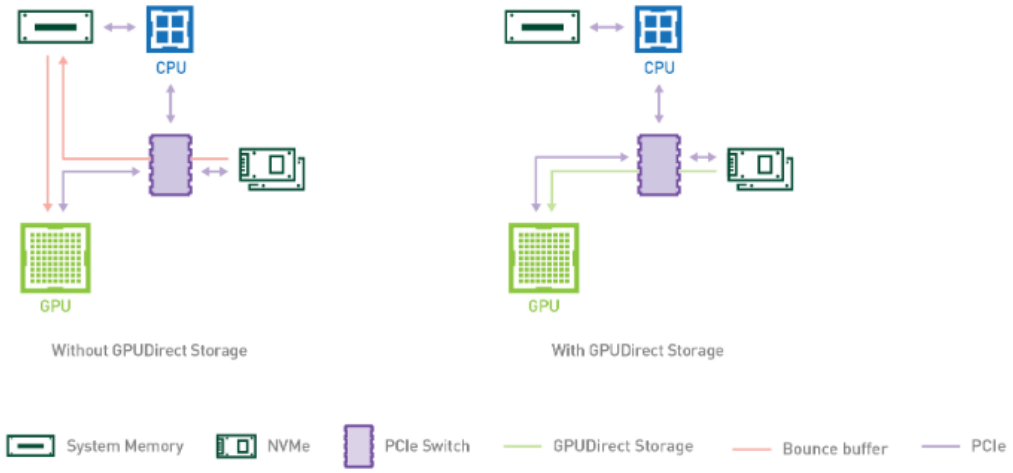
In the aggregation and combination phase of Graph Convolutional Networks (GCN), chain matrix multiplication is employed, which is highly data and memory intensive. Insufficient memory or out-of-core memory issues pose significant challenges for large GCN modules.

Need for Efficient Data Handling in GCN

During the combination phase of GCN processing, each node's features are updated based on its own features and the aggregated features of its neighbors. This phase is computationally intensive for several reasons:

1. **Large Volume of Data**
2. **High Computational Requirements**
3. **Dynamic Data Access Patterns**
4. **Scalability**

Background: GPU Direct Storage (GDS)



NVIDIA. (n.d.). GPUdirect Storage, from https://developer.nvidia.com/blog/gpudirect-storage/

- **Reduced Latency**: By eliminating the detour through CPU memory, GDS significantly cuts down the time it takes for data to reach the GPU. This is crucial for applications requiring real-time processing and analysis.
- **Increased Throughput**: Direct data paths allow for higher data throughput rates. This is beneficial in environments where large volumes of data need to be processed quickly, such as in video processing or complex simulations.

GPU Direct Storage (GDS) (Continued)

- **Enhanced Scalability**: GDS makes it easier to scale applications by removing the memory bottlenecks associated with large-scale data transfers between storage and GPUs. This is essential for expanding the capabilities of systems without compromising on performance.
- **Improved Resource Utilization**: By reducing the load on the CPU and its memory, other processes can use these resources more effectively, leading to overall improved system efficiency.

Disadvantages:

- **Sending small data**: GDS is optimized for high throughput and larger transfers, so sending many small packets individually can lead to inefficiencies.
- **Initialization Overhead**: Setting up a direct data path between storage and the GPU in GDS can result in higher relative latency for small data packets due to the setup overhead.

Contribution

- **Automatic Data Compression Technique** : This technique utilizes operator hints to detect sparsity based on a tiered memory hierarchy and selects the appropriate sparse matrix compression algorithms for matrices A and B, respectively.
- **Automatic Data Transfer Technique** : This method selects a low-latency, zero-copy path through a GPU-directed storage approach. It enables direct loading of sparse matrices to the GPU, bypassing the CPU to minimize CPU involvement. Additionally, verify whether the CPU is used to compress the matrix before transfer.
- **Metadata and Data Decoupling Technique**: This approach is designed for out-of-core SpGEMM computations to optimize space allocation within a tiered memory system, including High Bandwidth Memory (HBM), GDS, and host DRAM.

Peridot: Integrating GDS with GCNs and our solution for efficient memory management

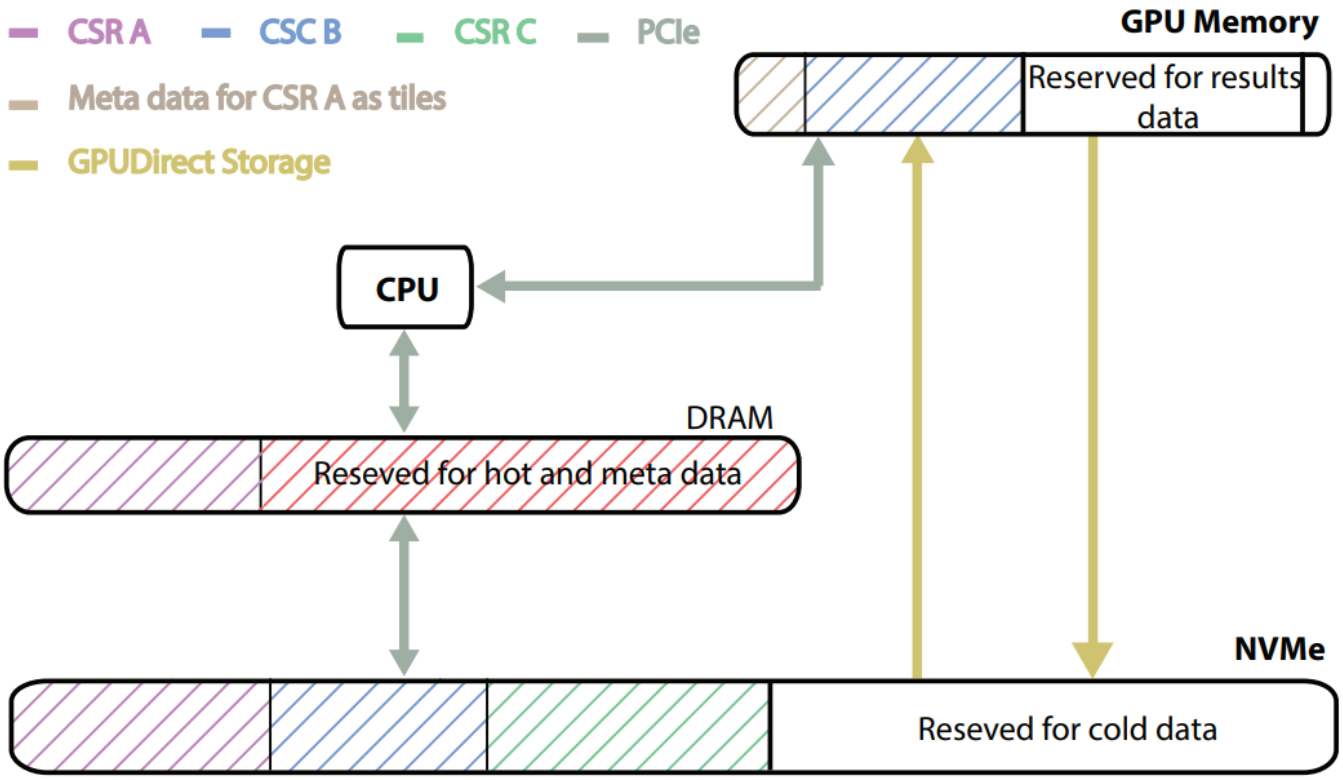


Fig. 2. Peridot memory utilization

Peridot: Integrating GDS with GCNs and our solution for efficient memory management (Continued)

Peridot's hierarchical memory management system offers multiple advantages:

1. **Reduced DRAM Usage**: Minimizes DRAM consumption, crucial for accelerating chain matrix multiplication processes.
2. **Lower Latency**: Utilizes GDS for large GCN data transfers, significantly reducing latency.
3. **CPU Offloading**: By offloading data transfer tasks to GDS, Peridot efficiently frees up CPU resources for other processes.
4. **Optimized Memory Utilization**: Systematic exploration and optimization of memory allocation for CSR A and CSC B formats ensure maximal memory efficiency.

Methodology

Experiment evaluation:

1. Baseline
2. Peridot
3. Peridot without GDS

Results

TABLE I
MEMORY REQUIREMENTS AND GPU MEMORY CONSTRAINTS FOR SUITESPARSE GRAPH DATASETS [5]

Dataset	No. Vertices	No. Edges	Memory Req. (GB)	GPU Mem. Restrict. (GB)
kmer_V2a	55.04M	117.21M	4.63	4
kmer_U1a	67.71M	138.77M	5.52	4
mycielskian18	196.6K	300.93M	9.63	8
kmer_P1a	139.35M	297.82M	11.76	10
kmer_A2a	170.72M	360.58M	14.27	12
kmer_V1r	214M	465.41M	18.317	16

TABLE II
PERFORMANCE COMPARISON OF GCN DATA HANDLING TECHNIQUES

Dataset	Baseline		Peridot		Peridot without GDS	
	Bandwith (MB/s)	Latency (ms)	Bandwith (MB/s)	Latency (ms)	Bandwith (MB/s)	Latency (ms)
kmer_V2a	3952.42	1054.15	6566.6	529	4761.79	729.502
kmer_U1a	Out of core	–	Out of core	–	Out of core	–
mycielskian18	3514.33	2285.49	7104.39	1017	4739.62	1524.42
kmer_P1a	3504.27	3773.47	7307.5	1207	4782.98	1844.07
kmer_A2a	3660.48	4526.35	7355.86	1455	4893.4	2187.19
kmer_V1r	3745.9	5104.49	7132.88	1926	4820.03	2850.17

Conclusion

- **Performance Improvements**: The integration of advanced data transfer techniques and memory management strategies in Peridot leads to substantial improvements in processing large-scale graph data.
- **Increased Bandwidth and Reduced Latency**: Demonstrated across multiple datasets, highlighting the efficiency of the system.
- **Effective Use of GDS**: The application of GDS technology within Peridot shows its capability to handle intensive data transfers effectively.
- **Addressing Key Challenges**: Specifically targets and overcomes the issues of memory limitations and data transfer delays, which have traditionally hindered the scalability of graph processing applications.

Related work

- A Systematic Survey of General Sparse Matrix-Matrix Multiplication [1]
  - The survey categorizes SpGEMM algorithms into row-wise, column-wise, and hybrid approaches, detailing their applicability and effectiveness across CPUs, GPUs, and distributed systems.
  - Highlights advancements in parallel computing that improve SpGEMM performance, with a focus on optimizing for the sparse nature of matrices to enhance computational efficiency and reduce memory latency.
- Sextans: A Streaming Accelerator for General-Purpose Sparse-Matrix Dense-Matrix Multiplicatio [2]
  - Sextans, a novel hardware accelerator optimized for sparse-matrix dense-matrix multiplication (SpMM), using a streaming architecture to enhance data flow and reduce memory latency.
  - Sextans supports a variety of sparse matrix formats and densities, ensuring its applicability across diverse scientific and data analytics domains.
- Accelerating sparse matrix–matrix multiplication with GPU Tensor Cores [3]
  - The study delves into optimizing memory access patterns and computational strategies for sparse matrices, ensuring efficient use of GPU Tensor Cores despite challenges from irregular data structures.
  - Demonstrates the potential and benefits of adapting sparse computations to Tensor Cores, laying a foundation for future work in optimizing sparse linear algebra operations with advanced GPU features.

Acknowledgement

This work was sponsored in part by the U.S. National Science Foundation (NSF).

Reference

[1] Gao, J., Ji, W., Chang, F., Han, S., Wei, B., Liu, Z., & Wang, Y. (2023). A systematic survey of general sparse matrix-matrix multiplication. *ACM Computing Surveys*, *55*(12), 1-36.

[2] Song, L., Chi, Y., Sohrabizadeh, A., Choi, Y. K., Lau, J., & Cong, J. (2022, February). Sextans: A streaming accelerator for general-purpose sparse-matrix dense-matrix multiplication. In *Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (pp. 65-77).

[3] Zachariadis, O., Satpute, N., Gómez-Luna, J., & Olivares, J. (2020). Accelerating sparse matrix–matrix multiplication with GPU Tensor Cores. *Computers & Electrical Engineering*, *88*, 106848.