PYTORCH CONFERENCE

PyG Performance Optimization for CPU

Graph Neural Networks Performance Optimization

Mingfei Ma[1], Matthias Fey[2]
Szarmach Michal[1], Ding Ke[1]
Yanbing Jiang[1], Jain Sanchit[1]

1. Intel Corporation
2. kumo.ai

C2

Abstract

Accelerating PyG CPU performance with faster sparse aggregation.

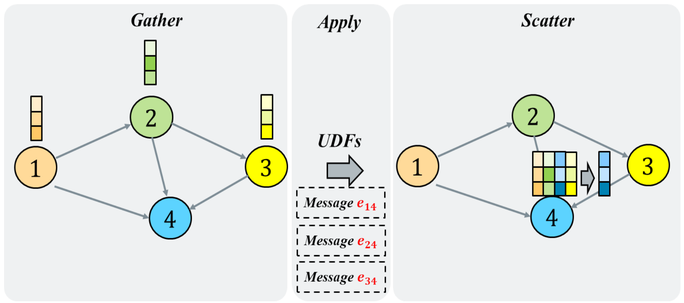
PyG is a library built upon PyTorch to easily write and train Graph Neural Networks, which heavily relies on the mechanism of Message Passing for information aggregation, we have optimized critical bottlenecks of Message Passing from PyTorch, including:

1. Scatter Reduce: maps to classic PyG use case when the EdgeIndex is stored in COO memory format.
2. SpMM Reduce: maps to the usage case when the EdgeIndex is stored in CSR memory format.

Message Passing

Message Passing in 3 steps:

- Gather: collect Edge-level information of adjacent nodes and edges.
- Apply: update the collected values with User-defined functions (UDFs).
- Scatter: aggregate to node-level information.

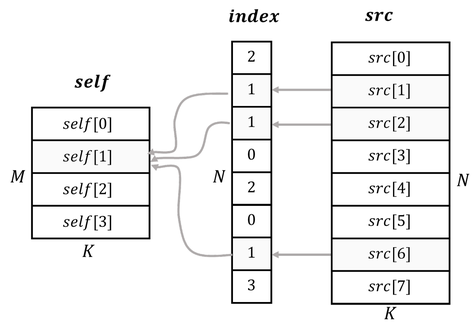


Performance is highly related to the memory format of EdgeIndex which records which pairs of nodes are connected:

- EdgeIndex in COO: physically stored in a [2, #edges] tensor which maps each connection of src->dst. Performance hotspot is scatter reduce.
- EdgeIndex in CSR: physically stored in CSR (Compressed Sparse Row) tensor. Performance hotspot is SpMM reduce.

Scatter Reduce Optimization

The pattern of scatter reduce is parallel in nature, updating values of self using values from src at the entries specified by index. The difficulty is to solve the write conflicts when paralleling on M.

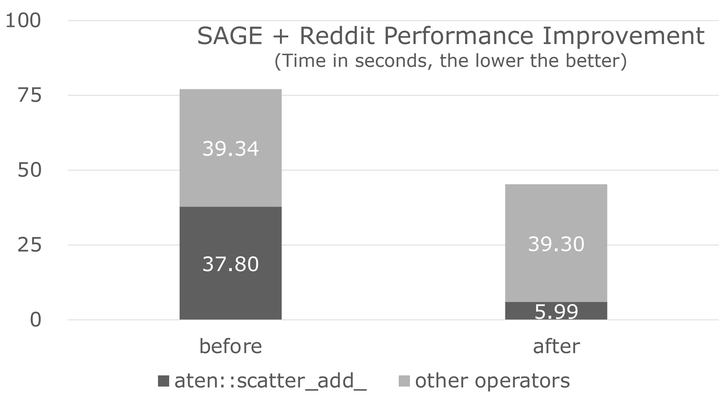


$self[1] = src[1] + src[2] + src[6]$

Optimization in 2 steps:

- Sorting: sort the indices to solve write conflicts.
- Reduction: parallel on the outer dimension of self and do vectorized reduction for each entry.

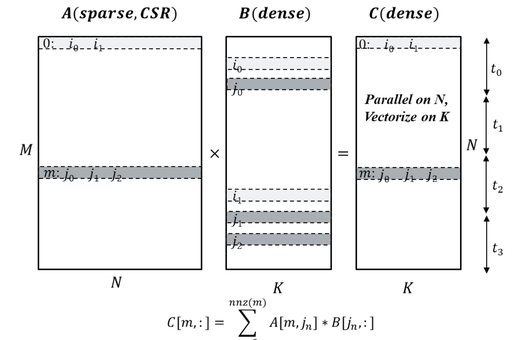
For its backward, e.g., gather. sorting will not be necessary, as the memory access pattern will not lead to write conflicts.



SAGE + reddit single socket inference performance is improved by 1.7x with scatter reduce optimization. CPU: Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz.

SpMM Reduce Optimization

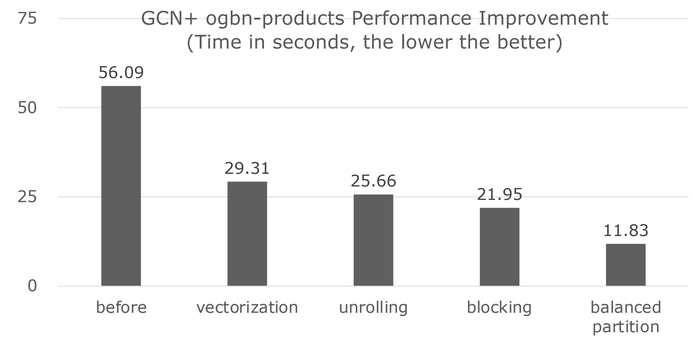
Sparse Matrix-Matrix Reduction is a fundamental operator in GNN, where A is sparse matrix in CSR format and B is a dense matrix, reduction type could be "sum", "mean" or "max".



$$C[m,:] = \sum_{n=0}^{nnz(m)} A[m, j_n] * B[j_n, :]$$

One challenge is to balance thread payload when paralleling along rows of sparse matrix A. Each row in A corresponds to a node and its number of connections may vary vastly, resulting into payload imbalance.

Optimizations: vectorization, unrolling, blocking and balanced partition.



GCN + ogbn-products single socket inference performance is improved by 4.3x with SpMM reduce optimization. Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz.

Acknowledgements

This research is part of a collaboration between Intel Corporation and kumo.ai

Take a photo to learn more:

QR Code Placeholder