CPU-PCGCN: Efficient Processing of Convolutional Graph Networks on CPU Architectures

Nicolás Meseguer-Iborra, José L. Abellán and Manuel E. Acacio

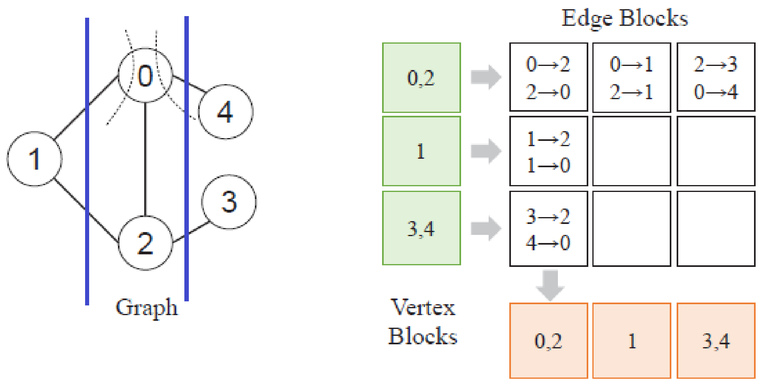
University of Murcia, Spain

Objective

This work introduces CPU-PCGCN, an implementation of PCGCN [1] using the CPU, achieving up to 3.94 times speed increase compared to the base GCN implementation.

Introduction

The acceleration in GCNs is mainly achieved due to: 1) Partitioning of the graph [2] and 2) Determination of the edge-blocks and their sparsity. Once processed, we will employ a dual computation method. Depending on the sparsity we will compute using either a sparse or a dense algorithm.



CPU-PCGCN Algorithm

Symbols: input graph: $G = (V, E)$, layers: $l = \{1, ..., L\}$, subgraphs: $\{S_k = (V_k, E_k, SS_k) | k = 1, ..., K\}$, vertices in subgraph k: V_k, edges in subgraph k: E_k, sparsity of the subgraph k: SS_k, edges between subgraph i and j: $E_{i,j}$, features of layer l: h^l (h^0 indicates the input features), weight of layer l: w^l

Ensure: Partition $G \rightarrow \{S_k | k = 1, ..., K\}$

// calculate a L layer CPU-PCGCN model

for $l = 1, ..., L$ **do**

  $a^l = h^{l-1} \times w^l$ ▷ Combination

  Split $a^l \rightarrow \{a^l_k | k = 1, ..., K\}$ according to subgraphs:

  // execute graph propagation

  for $k = 1, , K$ **do**

    // dual-mode computing

    if $SS_k > args.threshold$ **then**

      $h^l_k = \sum_{i=1}^{K} f_{spmm}(E_{k,i}, a^l_i)$ ▷ Sparse

    else

      $h^l_k = \sum_{i=1}^{K} f_{mm}(E_{k,i}, a^l_i)$ ▷ Dense

    end if

    // combine hidden states subgraph k

    $h^l = concat(h^l_k)$

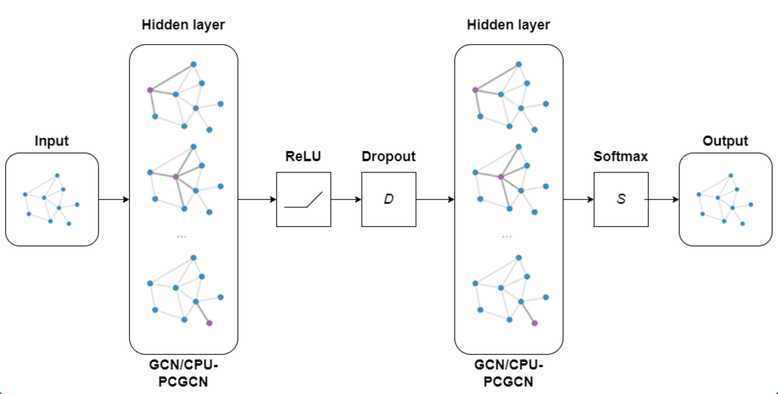
  end for

end for

return h^L;

Model Architecture

For a better understanding of CPUPCGCN, we provide a high-level overview. The model consists of two hidden layers known as GCN or CPU-PCGCN. Both layers are architecturally identical but differ in how they compute input parameters. The first layer operates on the complete graph in a compressed format, while the second layer works on partitions generated from the graph, also in a compressed format.

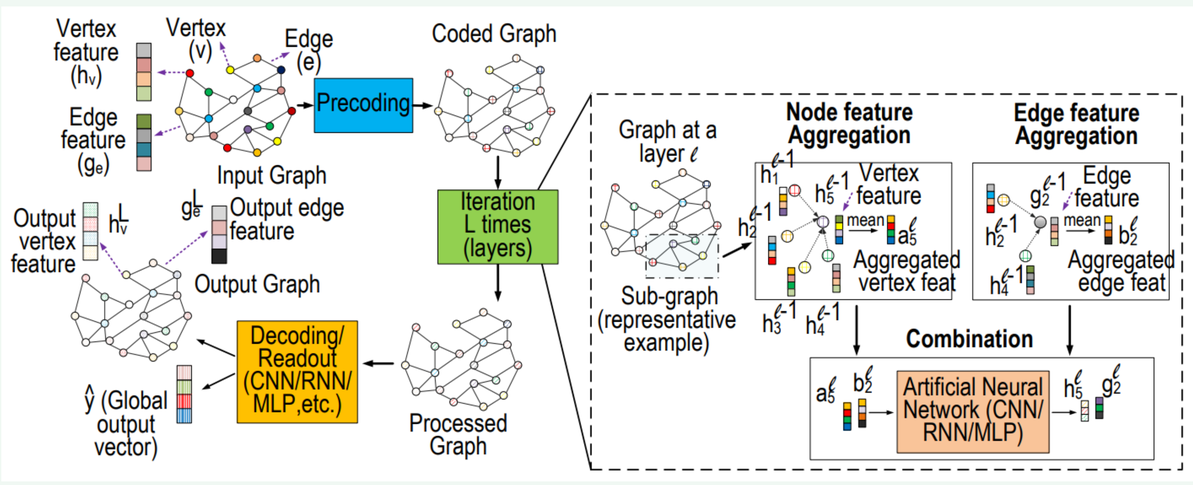


GNNs and GCNs

Graph Neural Networks

Two phases: **Aggregation**, collecting the features of neighboring vertices , and **Combination**, the new vertex features are updated using a DNN.

Graph Convolutional Networks



A single convolution transforms and aggregates information from neighbouring nodes. However they face two limitations: 1) Identity Matrix A and 2) Symmetric Normalization

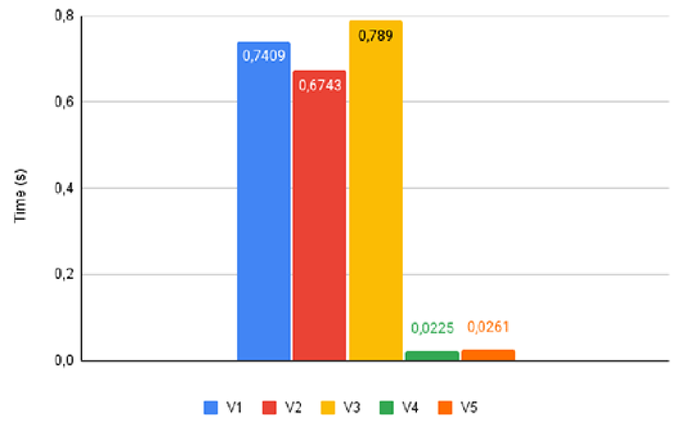
Results

Datasets	#vertex	#edge	#feature	#label	clustering coefficient
cora	2.7K	5.4K	1.4K	7	0.24
citeseer	3.3K	4.7K	3.7K	6	0.14
pubmed	19.7K	108.3K	500	3	0.06
PaRMAT-0.01	996	8K	259	30	NM
PaRMAT-0.06	1K	30K	731	61	NM
PaRMAT-0.12	100	600	48	6	NM
PaRMAT-1	996	800K	6.8K	405	NM

Time (s)							
Datasets	GCN (D)	GCN (S)	CPU-2	CPU-4	CPU-8	CPU-16	speedup
cora	88.74	81.53	46.74	48.58	56.96	72.53	**1.74x**
citeseer	82.61	69.42	46.11	49.58	56.90	74.97	**1.50x**
pubmed	1,259.93	795.184	248.56	303.22	347.44	446.47	**3.19x**
PaRMAT-0.01	30.70	28.86	19.78	21.71	25.36	33.12	**1.45x**
PaRMAT-0.06	43.01	37.81	21.67	23.70	26.31	34.06	**1.74x**
PaRMAT-0.12	3.76	2.41	1.9	2.04	2.63	4.54	**1.26x**
PaRMAT-1	1,171.2	1,050.3	266.48	229.98	262.54	387.72	**3.94x**

Time (s)					
Datasets	*best-time*	CPU-4-20	CPU-4-40	CPU-16-20	CPU-16-40
cora	46.74	53.17	50.65	74.81	71.27
citeseer	46.11	48.84	48.88	74.18	74.47
pubmed	248.56	287.19	288.13	445.91	446.56
PaRMAT-0.01	19.78	21.47	21.27	32.57	32.78
PaRMAT-0.06	21.67	23.28	23.55	33.78	34.38
PaRMAT-0.12	1.9	2.04	2.06	4.72	4.77
PaRMAT-1	229.98	224.13	220.90	398.34	403.47

Time (s)					
Phases	V1	V2	V3	V4	V5
Splitting a^l	0.098	0.0959	0.185	0.004	0.0065
Graph propagation	0.001	0.578	0.604	0.018	0.0196
Concat	0.642				
Torch conversion	NA	NA	NA	$\simeq 0$	$\simeq 0$
Total (s)	0.74	0.67	0.789	0.022	0.026



References

[1] Chao Tian, Lingxiao Ma, Zhi Yang, and Yafei Dai. Pcgcn: Partition-centric processing for accelerating graph convolutional network. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 936–945, 2020.

[2] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM JOURNAL ON SCIENTIFIC COMPUTING*, 20(1):359–392, 1998.

Acknowledgments

This work was supported by grants TED2021-130233B-C33 and RYC2021-031966-I both funded by MCIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR. Nicolás Meseguer was supported by grant 21803/FPI/22 from Fundación Séneca.