가속기 구조에서의 그래프 신경망 성능 특성 분석

이훈종° 구건재
고려대학교 컴퓨터학과
hunjong@korea.ac.kr, gunjaekoo@korea.ac.kr

Performance Analysis of Graph Convolutional Networks on Accelerator Architectures

Hunjong Lee° Gunjae Koo
Department of Computer Science and Engineering, Korea University

요 약

Graph neural networks의 연산 과정은 높은 희소성을 갖는 행렬 연산을 포함하고 있다. 그러므로, GNN 처리를 기존의 조밀 행렬 및 낮은 희소성을 가지는 행렬 연산에 특화된 가속기에서 수행할 경우 매우 낮은 효율을 보일 것으로 예상할 수 있다. 이 논문에서는 GNN의 대표적인 응용 방법인 GCN 커널을 행렬 연산 가속기 구조 중의 하나인 TPU와 SIGMA에서 수행시키면서 GCN 커널이 보여주는 성능 저하 요소를 분석하였다. 분석 결과 기존의 가속기 구조에서는 매우 높은 희소성을 가지는 행렬 연산이 낮은 효율성은 보여주는 것을 밝혔으며 이는 데이터의 이동 및 zero 값에 따른 PE의 낮은 utilization에 의한 것임을 밝혀내었다.

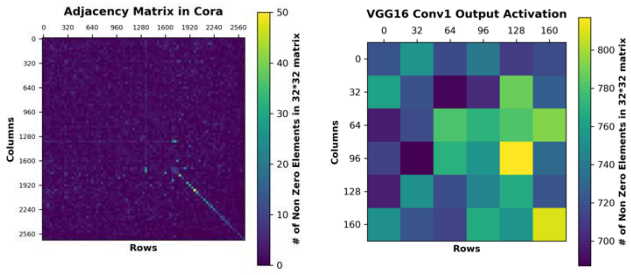
1. 서론

그래프 신경망(graph neural networks, GNNs)은 그래프 데이터 구조체에 인공 신경망 기술을 적용한 것으로서 비정형적인 연관 구조를 가지는 데이터 구조를 분석하고 예측하는 데에 사용될 수 있다 [1]. GNN은 기존의 정형적인 형태를 가지는 데이터에 적용되어 왔던 신경망 구조와 달리 비정형적인 연결 관계를 가지는 데이터 구조에 적용될 수 있으므로 사회 신경망 분석, 화학 구조 분석, 신약 개발 등의 여러 분야에 응용되어 큰 성과를 낼 수 있을 것으로 예상되고 있다. 그렇지만, GNN은 기존의 신경망에서 사용되는 데이터 구조와 달리 희소 행렬(sparse matrix)로 표현되는 연결 정보를 사용해야 하므로 기존의 신경망 처리에 사용되는 하드웨어 시스템을 그대로 사용할 경우 처리가 비효율적으로 수행되는 단점을 가지고 있다. 즉, 기존에 널리 사용되는 convolutional neural networks (CNNs)의 경우 조밀 행렬(dense matrix)의 형태를 가지는 데이터 구조를 주로 사용하므로 행렬 연산을 빠르게 계산할 수 있는 가속기 구조를 사용할 경우 효율적으로 신경망 처리 계산을 수행할 수 있다. 그렇지만, 그래프 신경망 처리는 희소 행렬 형태를 가지는 연결 정보를 사용하므로 기존의 가속기 구조는 비효율적이라고 예상할 수 있다. 본 논문에서는 CNN을 기반으로 개발된 가속기 구조를 사용하여 GNN 처리를 가속하는 경우를 성능적으로 분석하고자 한다.

Graph convolutional networks (GCNs)는 GNN 모델중에서 널리 사용되는 응용 방법이다. GCN은 크게 aggregation 부분과 combination 부분으로 나눌 수 있다 [2]. Combination 부분은 하나의 노드에서 처리된 특성 벡터에 fully-connected network를 적용한 것으로서 그래프 구조의 연결 특성과 무관하다. 그렇지만 aggregation 부분은 그래프 노드의 연결 정보를 가지고 있는 연결 행렬과 각 노드가 가지고 있는 특성 벡터의 행렬 곱 연산의 형태를 가지기 때문에 sparse matrix multiplication (SpMM)을 효율적으로 처리할 수 있는 하드웨어 구조가 요구된다.[9] 그렇지만, 기존의 가속기 구조는 강한 희소성을 가지는 행렬을 고려하여 설계되지 않았기 때문에 효율성이 떨어질 것이라고 예상된다. 그리하여, 이 논문에서는 행렬 곱셈 연산을 가속하는 가속기 구조에서 GCN의 aggregation 부분을 수행할 때의 특성을 분석하였다.

2. GCN 커널 및 가속기 구조 소개

본 논문에서는 GCN의 aggregation 부분을 대표적은 행렬 곱셈 가속기인 tensor processing unit (TPU)와 Bitmap 형태로 압축된 행렬 곱셈을 지원하는 SIGMA 구조를 사용하여 분석하였다[3][4]. 기본적으로 가속기에서 사용하는 데이터에 zero 값이 존재할 경우 가속기의 processing elements (PEs)의 자원이 아무런 이득없이 소모되며, 또한 zero 데이터를 메모리로부터 가져오기 위해서 가속기의 자원이 불필요하게 소모되어 가속기의 효율성이 낮아지게 된다.



[그림 1, Matrix Sparsity in 32*32 Matrix Block]

이 성과는 2022년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2021R1C101272)

2.1 GCN 커널 분석

앞에서 언급한대로 GCN의 aggregation 부분의 핵심 연산은 그래프의 연결을 표현하는 인접(adjacency) 행렬과 각 노드의 특성을 표현하는 특성 벡터의 행렬곱으로 표현될 수 있다. 표1에서 볼 수 있듯이 GCN에서 사용하는 데이터셋은 인접 행렬의 경우 모두 98% 이상의 높은 희소성을 가지고 있다. 즉, GCN에서 행렬을 구성하는 많은 요소들이 zero 값을 가지고 있으며 이를 기존의 행렬 곱셈을 지원하는 가속기 구조에서 수행할 경우 utilization이 극도로 낮게 나올 것이라고 예상할 수 있다. 또한 그림 1에서 볼 수 있듯이, DNN에 사용되는 입력 행렬(오른쪽)은 구성 요소끼리 군집된 형태를 보이는 반면, GCN에 사용되는 인접 행렬(왼쪽)은 구성 요소간의 거리가 불규칙하다는 것을 알 수 있다.

2.2 Systolic array 기반 가속기 구조

TPU는 행렬 곱셈을 가속하기 위해 설계되었으며 systolic array의 형태를 가지고 있다 [4][5]. Systolic array는 입력 행렬 혹은 weight를 PE의 내부 register를 이용하여 재사용이 가능하도록 설계하여 하드웨어의 효율성을 높였다. TPU에서는 weight를 재사용하고 각 입력에 대한 multiply-and-accumulation (MAC) 연산 결과를 인접 PE로 전달하여 accumulation 연산을 수행하는 구조를 가지고 있다. 그렇지만, TPU의 구조는 조밀 행렬의 연산에는 적합하지만, 입력 데이터가 많은 zero값을 가질 경우 PE에서의 효율성이 크게 떨어지게 된다.

2.3 희소 행렬 연산을 위한 가속기 구조

Zero값을 많이 가지는 행렬 연산을 기존의 가속기에서 수행할 경우 효율성이 크게 떨어지는 문제로 인해서 많은 연구자들이 희소 행렬 연산을 효율적으로 수행하기 위한 구조를 제시하였다. SIGMA는 희소 행렬 연산을 지원하는 가속기 구조를 가지고 있으며 희소 행렬 데이터를 연산하기 위해 압축된 형태의 행렬 데이터를 입력 받을 수 있는 구조로 설계가 되었다 [3]. 즉, SIGMA는 희소 행렬 데이터를 bitmap 방식을 이용하여 non-zero 데이터만을 메모리에 저장한 후에 실제 연산에 사용되는 구성 요소만을 PE로 전달하는 구조를 가지고 있다. Systolic array 방식과 마찬가지로 SIGMA에서도 PE 내부 registers를 이용하여 데이터를 재활용하여 가속기의 효율을 높일 수 있도록 설계가 되었다. SIGMA는 내부 메모리에 저장된 행렬 데이터를 distributed network를 이용하여 PE에 전달하고 여러 개의 PE에서 계산된 곱셈 결과를 reduction tree를 이용하여 합치는 구조를 가지고 있다. 즉, SIGMA는 zero 값에 대한 MAC 연산을 배제하고 non-zero 값만을 선택하여 선택적으로 합칠 수 있는 구조를 제시하여 희소 행렬 연산에 따른 비효율성을 낮추고 있다. 반면, SIGMA와 같은 희소 행렬 연산을 위한 가속기 구조에서는 non-zero값을 PE에 전달하는 데이터 전송 부분과 이를 연산하는 부분에서 성능 저하 요소가 발견될 수 있다. 본 논문에서는 이를 loading latency와 streaming latency로 구분하였다.

Loading latency는 내부 SRAM에서 PE의 registers에 저장하기 위한 데이터를 전달하기 위해 필요한 시간을 의미한다. 즉, SIGMA에서는 non-zero 데이터만을 저장하는 bitmap 압축 방식의 입력 데이터를 디코딩 및 그룹화하여 PE로 데이터를 전달하게 된다. 이 때 그룹 내의 구성 요소의 개수가 적을 경우 많은 수의 데이터 전송이 필요하다. 결과적으로 군집된 형태를 가지는 행렬 데이터의 경우 효율적으로 loading이 이루어지지만, 구성요소 사이의 거리가 멀 수록 loading latency는 증가하게 된다.

Streaming latency는 SRAM에서 PE로 행렬 데이터를 broadcast하는 과정에서 필요한 시간을 의미한다. SIGMA에서는 PE의 registers에 저장할 stationary 데이터는 non-zero 값 만을 선별하여 저장하지만, 행렬 데이터를 streaming할 경우에는 조밀 행렬 형태를 가지는 행렬 데이터를 broadcasting하게 된다. 따라서, 전달하는 데이터가 많은 zero 값을 가지게 될 경우 효율이 떨어지게 된다.

3. 실험 및 분석

본 논문에서는 GCN 커널의 성능을 분석하기 위해서 GCN의 aggregation 부분을 TPU와 SIGMA에서 실행시켰을 때 발생하는 loading latency, streaming latency와 PE utilization, memory access를 분석하였다. 실험은 cycle-accurate 가속기 시뮬레이터인 STONNE[8]을 사용하였으며 가속기의 configuration은 표2와 같다. GCN 커널은 표1에서 나온 Cora와 Coauthor CS 데이터셋을 사용하였으며 희소성이 50%인 행렬과 함께 비교하여 희소 행렬 연산에 따른 performance overhead를 측정하였다.

Datasets	Cora	Coauthor Physics	Coauthor CS	CitationFull DBLP	CitationFull PubMed
Adj_Sparsity	0.1439%	0.0417%	0.0487%	0.0337%	0.0228%
Input_Sparsity	1.268%	0.392%	0.875%	0.318%	10.022%

[표 1, Graph Dataset의 Adjacency, Input Matrix Sparsity]

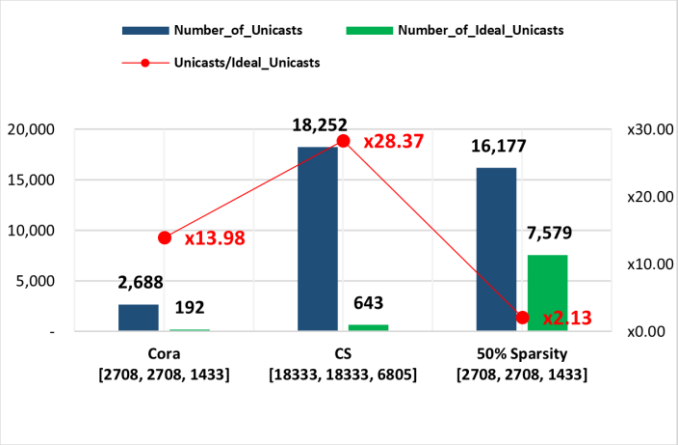
Architecture	TPU	SIGMA
Dataflow	Weight_Stationary	MK_STA, KN_STR
Multiplier	256	256
Reduction Bandwidth	256	256
Distribution Bandwidth	32	256
Sparsity_Support	X	O

[표 2, Hardware Configuration]

3.1 Loading latency

그림 2는 SIGMA에서 GCN 커널의 aggregation 부분의 adjacency 행렬 로딩에 필요한 latency를 측정한 결과를 나타낸 것이다. 데이터셋 아래의 숫자는 행렬의 dimension을 나타낸다. Ideal unicast는 SRAM과 PE간의 데이터 전송에서 zero값이 포함되지 않는 이상적인 값을 나타낸다. 실험에 사용한 그래프의 adjacency 행렬은 매우 많은 수의 zero값을 포함하고 있으며 구성 요소 사이의 거리는 매우 먼 특성을 가지고 있다. 그러므로, 실제 unicast가 발생하는 횟수는 ideal한 횟수 대비 매우 많은 수가 필요하다는 것을 알 수 있다. 즉, 비효율적인 unicast가 많이 발생할 수록 SRAM과 PE 사이의 대역폭을 효과적으로 사용하지 못하는 문제가 발생한다. 때문에 이상적인 unicast의 횟수 대비 14~29배

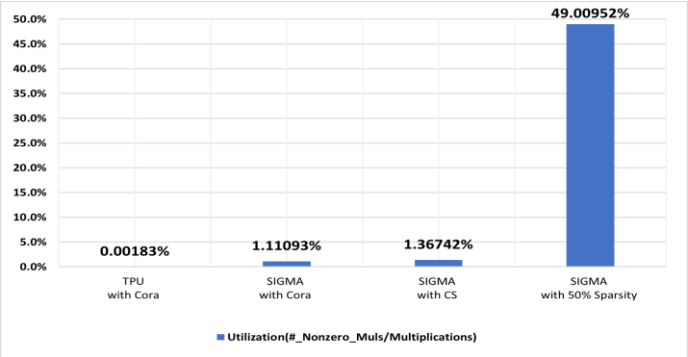
정도의 unicast가 발생하며 이는 adjacency 행렬의 크기와 희소성이 커질 수록 overhead가 커지는 경향을 보인다.



[그림 2, Unicast Overhead in SIGMA(Loading Latency)]

3.2 Streaming latency 및 PE utilization

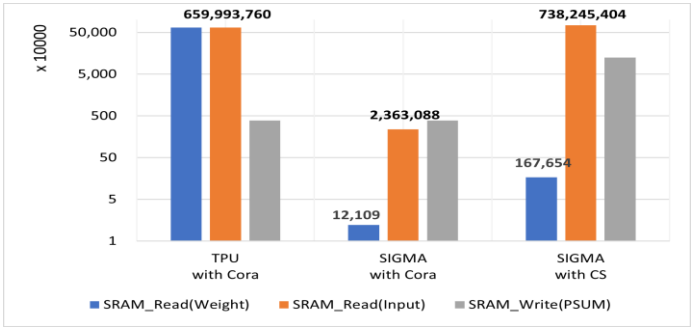
그림3은 TPU와 SIGMA에서 PE의 utilization을 보여준다. TPU와는 달리 SIGMA는 행렬의 희소성을 고려한 구조를 가지고 있으며 높은 희소성을 가지는 행렬을 stationary 행렬로 사용하여 미리 PE에 로딩을 하고, 다른 행렬은 입력 행렬로 사용하는 방식을 가지고 있다. 그렇지만 표1에서는 보는 바와 같이 GCN에 사용되는 행렬은 모두 대체적으로 높은 희소성을 가지고 있으므로 PE의 utilization이 전체적으로 낮게 나타나게 된다. 본 실험에서 TPU의 PE utilization은 매우 낮은 수치를 보였으며, SIGMA도 2%가 안되는 PE utilization을 보였다. 이는 50%의 희소성을 가지는 입력 행렬이 50% 정도의 PE utilization을 보여주는 것에 비하면 매우 낮은 수치라고 말할 수 있다. 즉, 기존의 가속기 구조가 매우 높은 희소성을 보여주는 GCN의 데이터 구조에는 적합하지 않다는 것을 보여주고 있다.



[그림 3, Processing Element Utilization]

3.3 SRAM access

그림4는 TPU와 SIGMA에서 adjacency 행렬을 loading할 때와 입력 행렬을 streaming할 때의 SRAM에서 데이터를 fetch하는 횟수를 나타낸 것이다. TPU에서는 희소성이 높은 행렬에 대한 압축 형태를 지원하지 않기 때문에 매우 많은 수의 SRAM Read가 발생한다. SIGMA에서는 bitmap 방식의 압축 데이터 형식을 지원하므로 상대적으로는 적은 양의 SRAM read가 발생하지만, 입력 행렬을 streaming할 때에는 zero 데이터도 전송이 필요하므로 많은 양의 SRAM Read가 발생하는 것을 볼 수 있다.



[그림 4, Number of SRAM Reads/Writes]

4. 결론

본 논문에서는 GCN을 기존의 가속기 구조에서 실행했을 때의 성능 저하 요소를 분석하였다. 기존의 가속기는 군집 형태를 가지는 행렬 연산에 있어서는 높은 효율을 보여주고 있지만, GCN에서 사용하는 희소 행렬 연산에는 낮은 효율을 보여주는 것을 볼 수 있다. 이는 기존의 가속기가 비교적 낮은 희소성을 가지는 행렬 연산을 고려하여 설계가 되었기 때문이다. 그러므로, 높은 희소성을 가지는 GNN 커널의 연산을 효과적으로 수행하기 위해서는 불필요한 zero값을 이동시키고 연산하는 데에 필요한 오버헤드를 최소로 할 수 있는 효과적인 하드웨어 구조가 필요함을 본 논문에서 밝혔다.

[1] Scarselli, Franco, et al. "The graph neural network model." IEEE transactions on neural networks 20.1, 2008.
[2] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907,2016.
[3] Qin, Eric, et al. "Sigma: A sparse and irregular gemm accelerator with flexible interconnects for DNN training." 2020 HPC). IEEE, pp. 58-70, 2020.
[4] Jouppi, Norman P., et al. "In-datacenter performance analysis of a tensor processing unit." Proceedings of the 44th annual international symposium on computer architecture. 2017.
[5] Kung, Hsiang-Tsung. "Why systolic architectures?." Computer 15.01, pp. 37-46, 1982.
[6] Sze, Vivienne, et al. "Efficient processing of deep neural networks: A tutorial and survey." Proceedings of the IEEE 105.12, pp. 2295-2329, 2017.
[7] Nassimi, David, and Sartaj Sahni. "A self-routing Benes network and parallel permutation algorithms." IEEE Transactions on computers 30.05, pp. 332-340, 1981.
[8] Muñoz-Martínez, Francisco, et al. "STONNE: Enabling Cycle-Level Microarchitectural Simulation for DNN Inference Accelerators." 2021 IISWC, IEEE, 2021.
[9] Grossman, Max, et al. "A survey of sparse matrix-vector multiplication performance on large matrices." arXiv preprint arXiv:1608.00636.2016.