GR-362

SCALE-Sim Extension to Support GNN Inputs And CNN Backpropagation

Introduction

Long training time on large-scale machine learning models now is a main obstacle that limits the development of the AI community. For example, the OpenAI GPT-4 takes 4-7 months [6] to complete the training. Based on the AI model statistic from OpenAI, the training complexity of AI models doubled every 3.4 months [7]. However, the system performance improvement rate from Moore's Law only doubles every 18 months. Thus, we can forecast a more significant gap in the near future, and we should explore application-specific system architectures to support upcoming AI models better.

SCALE-Sim [2-4] is an open-source simulator of CNN Accelerator, which includes a systolic array and three SRAMs, and is similar to the Google TPU architecture. However, SCALE-Sim only supports forward propagation for CNN inference. Therefore, to support the simulation of parallel training for large-scale AI models, we must extend SCALE-Sim with backpropagation implementation and integrate another network simulator (BookSim [8]). Furthermore, GNN has become popular nowadays. To expand this open-source simulation tool's applicability, we will further extend the SCALE-Sim to support GNN training via GraphSAGE [1] to generate node embedding.

Motivation

AI is everywhere. However, the system permanence growth rate cannot follow the development of new AI models. So, we have to explore AI-specific system architectures to narrow down this gap. In this work, we will develop a simulation tool by integrating open-source SCALE-Sim and Booksim to evaluate the training performance of AI-specific systems.

Innovations

1) Extend the SCALE-Sim to support the CNN backpropagation.
2) Integrate the SCALE-Sim with BookSim to support parallel CNN training
3) Extend the SCALE-Sim to support GNN training.

Methodologies

CNN Training Extension:

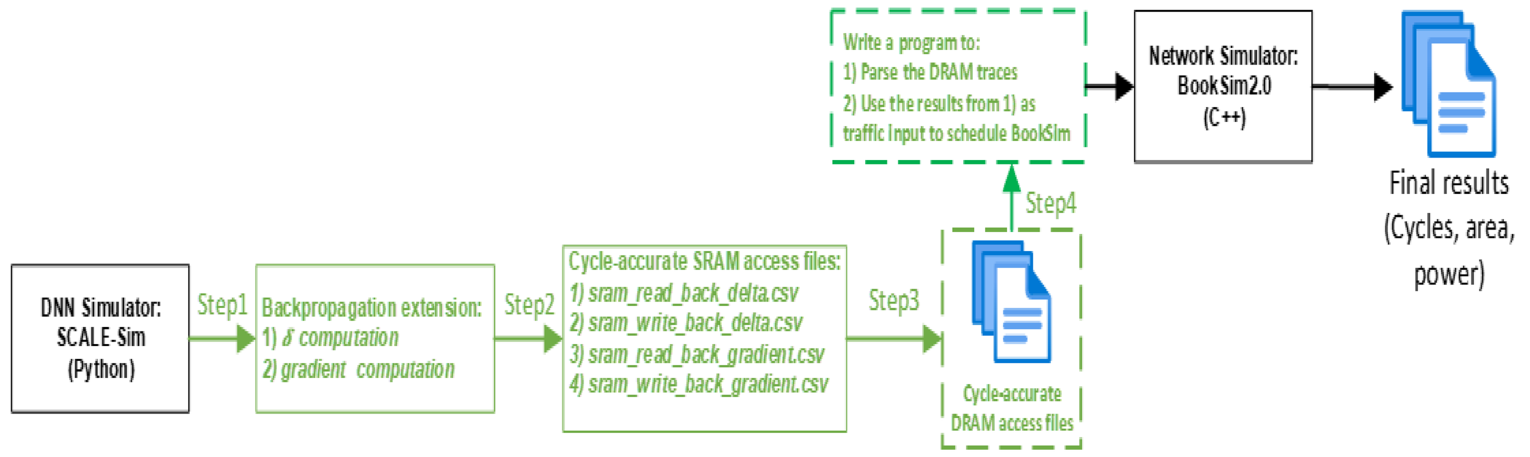


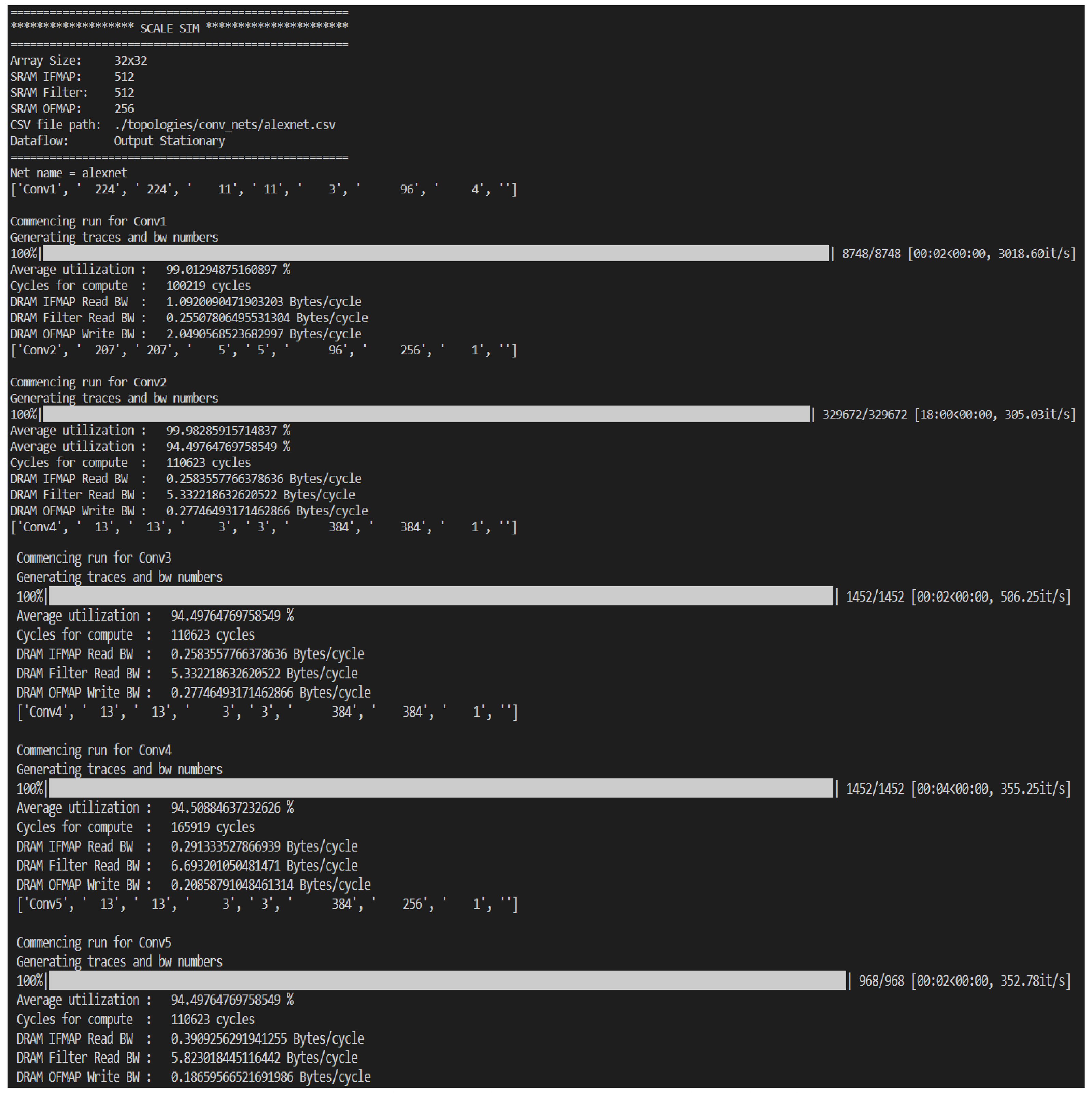
Figure 1: The Proposed Simulation Toolchain for CNN Parallel Training Systems

GNN Input Extension:

The Systolic array-based GNN accelerator is a simulator that eases the computation process used in GNN models by accepting graphs as input. The input to the GNN simulator can be unstructured as a graph in non-Euclidean space, consisting of nodes and edges, each with a set of associated features. The graph input is typically represented as a pair of matrices: an adjacency matrix and a feature matrix. Graph embedding techniques such as GraphSAGE are then used to learn low-dimensional vector representations, or embeddings, of nodes in a graph by sampling and aggregating information from the neighborhood of each node. Once the embeddings are learned, they can be used for various downstream tasks, such as node classification, link prediction, and graph clustering. To use GraphSAGE embeddings as inputs to the SCALE-Sim, the weights of the CNN need to be specified and trained in addition to the pooled GraphSAGE embeddings. The weights of a CNN [5] are the learnable parameters that are used to transform the input tensor into a higher-level representation that captures important features of the input data. These weights are typically initialized randomly and then trained using an optimization algorithm to minimize a loss function that measures the difference between the predicted outputs of the CNN and the ground-truth outputs. The weights of the CNN would be trained to map the input tensor of pooled embeddings to an output tensor that represents the desired output.

Results

Currently, we successfully ran the default SCALE-Sim on KSU HPC. The systematic evaluation of CNN accelerators using SCALE-Sim revealed that scalability is crucial for system efficiency. The comparative graph shows that as the systolic array's size increases, the convolutions' cycle counts decrease. The evaluation also highlighted potential bottlenecks in the accelerator's design, such as limited memory bandwidth and computational resources, and optimized the architecture for improved performance and efficiency. This study emphasizes the need for scalable and adaptive hardware designs that efficiently handle a wide range of CNN/GNN models and use simulation frameworks for their systematic evaluation and optimization.



```
====================================================
***************** SCALE SIM ********************
====================================================
Array Size:     32x32
SRAM IFMAP:     512
SRAM Filter:    512
SRAM OFMAP:     256
CSV file path:  ./topologies/conv_nets/alexnet.csv
Dataflow:       Output Stationary
====================================================
Net name = alexnet
['Conv1', '  224', ' 224', '    11', ' 11', '    3', '      96', '     4', '']

Commencing run for Conv1
Generating traces and bw numbers
100%|████████████████████████████████| 8748/8748 [00:02<00:00, 3018.60it/s]
Average utilization :   99.01294875160897 %
Cycles for compute  :   100219 cycles
DRAM IFMAP Read BW  :   1.0920090471903203 Bytes/cycle
DRAM Filter Read BW :   0.25507806495531304 Bytes/cycle
DRAM OFMAP Write BW :   2.0490568523682997 Bytes/cycle
['Conv2', '  207', ' 207', '    5', ' 5', '      96', '     256', '    1', '']

Commencing run for Conv2
Generating traces and bw numbers
100%|████████████████████████████████| 329672/329672 [18:00<00:00, 305.03it/s]
Average utilization :   99.98285915714837 %
Average utilization :   94.49764769758549 %
Cycles for compute  :   110623 cycles
DRAM IFMAP Read BW  :   0.2583557766378636 Bytes/cycle
DRAM Filter Read BW :   5.332218632620522 Bytes/cycle
DRAM OFMAP Write BW :   0.27746493171462866 Bytes/cycle
['Conv4', '  13', '  13', '    3', ' 3', '      384', '    384', '    1', '']

Commencing run for Conv3
Generating traces and bw numbers
100%|████████████████████████████████| 1452/1452 [00:02<00:00, 506.25it/s]
Average utilization :   94.49764769758549 %
Cycles for compute  :   110623 cycles
DRAM IFMAP Read BW  :   0.2583557766378636 Bytes/cycle
DRAM Filter Read BW :   5.332218632620522 Bytes/cycle
DRAM OFMAP Write BW :   0.27746493171462866 Bytes/cycle
['Conv4', '  13', '  13', '    3', ' 3', '      384', '    384', '    1', '']

Commencing run for Conv4
Generating traces and bw numbers
100%|████████████████████████████████| 1452/1452 [00:04<00:00, 355.25it/s]
Average utilization :   94.50884637232626 %
Cycles for compute  :   165919 cycles
DRAM IFMAP Read BW  :   0.291333527866939 Bytes/cycle
DRAM Filter Read BW :   6.693201050481471 Bytes/cycle
DRAM OFMAP Write BW :   0.20858791048461314 Bytes/cycle
['Conv5', '  13', '  13', '    3', ' 3', '      384', '    256', '    1', '']

Commencing run for Conv5
Generating traces and bw numbers
100%|████████████████████████████████| 968/968 [00:02<00:00, 352.78it/s]
Average utilization :   94.49764769758549 %
Cycles for compute  :   110623 cycles
DRAM IFMAP Read BW  :   0.3909256291941255 Bytes/cycle
DRAM Filter Read BW :   5.823018445116442 Bytes/cycle
DRAM OFMAP Write BW :   0.18659566521691986 Bytes/cycle
```

Figure 2: Output of SCALE-Sim with systolic array dimension of 32*32

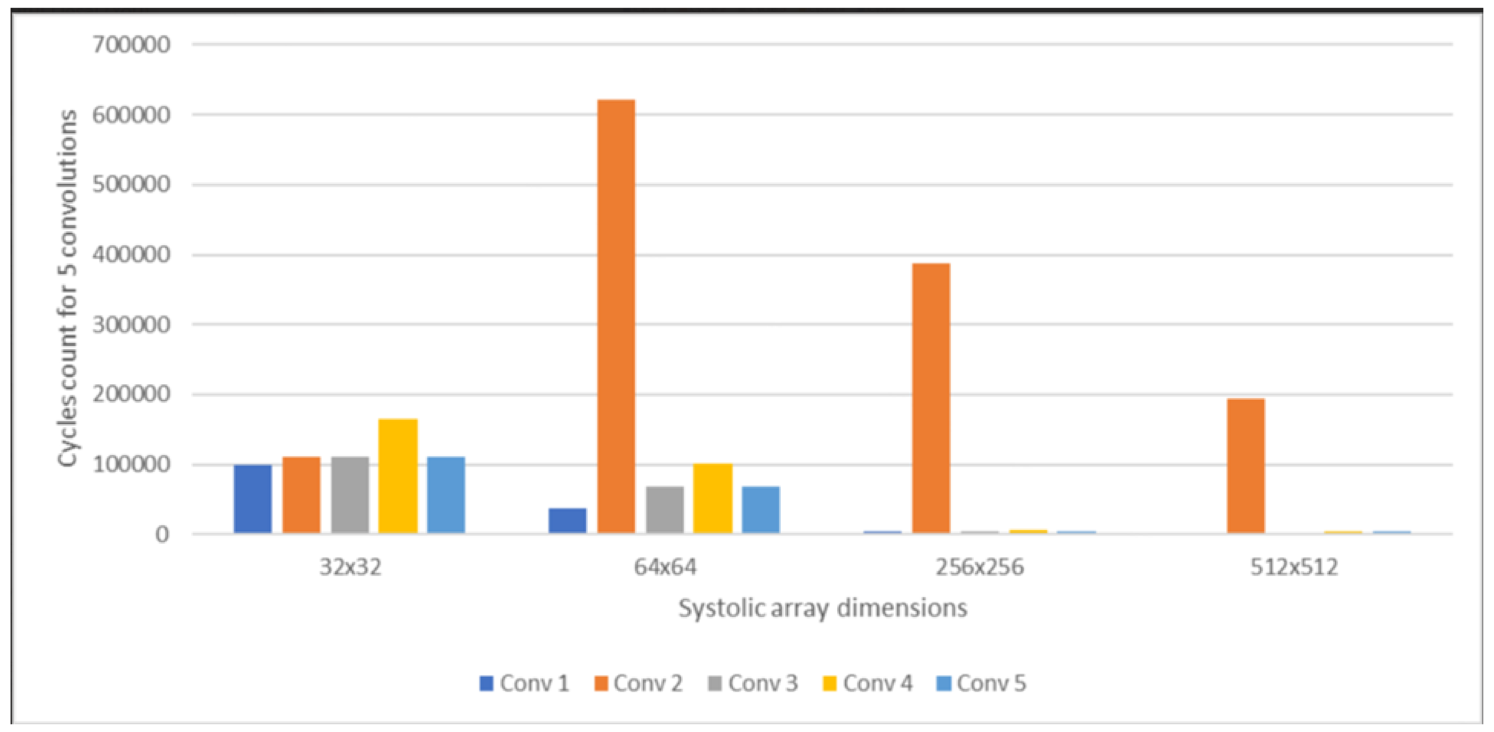


Figure 3: Connection between the cycle count for convolution-layers and different systolic array dimensions

Conclusions

In this work, we explore the code structure of SCALE-sim to extend to support parallel CNN/GNN training. From the CNN side, we explore how to extend the backpropagation in SCALE-Sim and integrate it with an open-source network simulator (Booksim). From the GNN side, we explore the GraphSAGE embeddings methodology to ensure GNN inputs could fit into this parallel training simulator. In other words, we are building an open-source simulation tool to evaluate CNN/GNN parallel training systems via integrating SCALE-Sim and BookSim. The SCALE-Sim has been successfully run in the KSU HPC. Extreme heterogeneous architecture is a strong candidate for the next-generation supercomputer, and the machine learning accelerator should be a critical component for this future architecture. We are using the KSU HPC to explore the architectures of the next-generation Zettascale HPC.

Contact Information

Rutul Desai - rdesai18@students.kennesaw.edu
Akshaya Pasupuleti - ppasupul@students.kennesaw.edu
Prof. Bobin Deng - bdeng2@kennesaw.edu
Prof. Guangchi Liu - gliu8@kennesaw.edu

References

[1] Hamilton, W.L., Ying, R. and Leskovec, J. (2018) Inductive representation learning on large graphs, arXiv.org. Available at: https://arxiv.org/abs/1706.02216 (Accessed: April 7, 2023).

[2] Samajdar, A. et al. (2019) Scale-SIM: Systolic CNN accelerator simulator, arXiv.org. Available at: https://arxiv.org/abs/1811.02883 (Accessed: April 7, 2023).

[3] Samajdar, A., Zhu, Y. and Whatmough, P. (2019) ARM-software/scale-SIM, GitHub. Available at: https://github.com/ARM-software/SCALE-Sim (Accessed: April 7, 2023).

[4] Samajdar, A. et al. (2018) A Systematic Methodology for Characterizing Scalability of DNN Accelerators using SCALE-Sim. Available at: https://cpb-us-w2.wpmucdn.com/sites.gatech.edu/dist/c/332/files/2020/03/scalesim_ispass2020.pdf (Accessed: April 7, 2023).

[5] Bazaga, A. et al. (2019) A convolutional neural network for the automatic diagnosis of collagen VI related muscular dystrophies, arXiv.org. Available at: https://arxiv.org/abs/1901.11074 (Accessed: April 7, 2023).

[6] Stephen McAleese, Retrospective on 'GPT-4 Predictions' After the Release of GPT-4, Available at: https://www.lesswrong.com/posts/iQx2eeHKLwgBYdWPZ/retrospective-on-gpt-4-predictions-after-the-release-of-gpt

[7] Amodei, D., Hernandez, D., Sastry, G., Clark, J., Brockman, G., & Sutskever, I. (2018, May). *AI and Compute*. Open AI. https://openai.com/blog/ai-and-compute/

[8] Nan Jiang et al., "A detailed and flexible cycle-accurate Network-on-Chip simulator," 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Austin, TX, USA, 2013, pp. 86-96, doi: 10.1109/ISPASS.2013.6557149.

KENNESAW STATE UNIVERSITY
COLLEGE OF COMPUTING AND SOFTWARE ENGINEERING

Authors: Rutul Desai, Akshaya Pasupuleti
Advisors: Prof. Bobin Deng, Prof. Guangchi Liu