Efficient training of graph classification models - Check-In

Hao-Yu Shih, Winfred Li, Noah Cohen Kalafut

November 30th, 2021

1 Work so Far

- Experimentation with graph analytics in GraphFrames (spark backend)
- Graph classification on brain network data [3] using DGL [1]
  - We implemented simple graph classification models to test for parallelization while controlling model complexity. Over 100 epochs, using inter-batch sampling we got the timings in table 1. The diminishing returns here are self-evident and likely caused by excessive reshuffling of batches and weight syncing. Delayed shuffling or skipped synchronizations could be an interesting area to explore.

Nodes	**1**	**2**	**3**	**4**
Time	7.53	3.76	2.73	2.36

Table 1: Distributed convolutional model on *Peking_1* [3] over 100 epochs.

- Graph classification with DGL
  - We implemented the GraphSAGE model [2] in a distributed fashion in order to get a sense of the scaling behavior of DGL. The results are shown in figure 1. Based on this, we can see that the epoch time does not scale proportionally with the number of nodes, and we believe this is due to increased communication overhead. Because the DGL trainer communicates between partitions by default, one future direction we are exploring is to find a way to train partitions independently as well as interleave independent training with communication. However, we have not been able to figure out a way to do this yet.

2 Challenges we've Run Into

- Reading in varying data formats efficiently

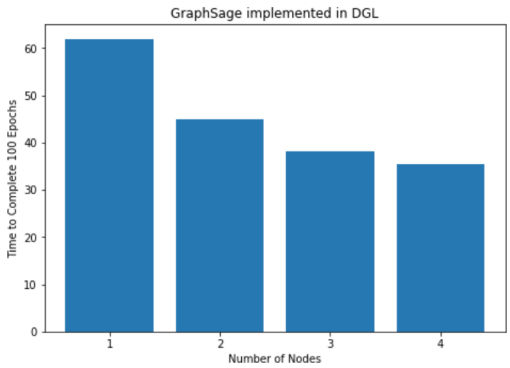

Figure 1: GraphSAGE Scaling

  - There are a wide variety of formats for graph storage. What is most efficient depends on the graph processing framework.
  - Inter or intra-batch parallelization dictates how our graphs are distributed for training. The latter is generally faster on startup but may be overshadowed by the communication overhead.
- Nodes Load balancing
  - We use Metis to partition the graph. However, Metis could only balance the number of nodes in each partition. Now we are trying using DGL's API to balance between in-degrees of nodes of different node types.
- Separating Training and Communication
  - As mentioned above in regards to GraphSAGE, we would like to be able to train partitions independently, but it is not clear how. One thing we've tried is to manually remove edges between partitions but this has not worked due to possible inconsistencies with metadata files.
- Processing graphs of varying size
  - If a dataset has graphs that greatly vary in size (such as *Peking_1* from [3], which ranges from

7 to 134 nodes), balancing the training workload within a batch is difficult, and lends itself to stragglers – especially with large batch sizes.

3 Future Timeline

- Dec 7: Finalize Framework
- Dec 10: Finish Framework Testing
- Dec 14: Presentation Due
- Dec 20: Report Due

4 Need Help With

- Nothing as of this moment

5 Miscellaneous Comments

Graph classification is an interesting subset of graph analysis to explore, as it introduces more complex ways to partition within the model when compared to a traditional classification task. For example, graph partitioning could be mixed with traditional inter-batch parallelization. Graphs can also potentially be compressed in a more straight-forward way, potentially leading to reduction of network overheads.

References

[1] `https://docs.dgl.ai/index.html`.

[2] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[3] S. Pan, J. Wu, X. Zhu, G. Long, and C. Zhang. Task sensitive feature exploration and learning for multitask graph classification. *IEEE Transactions on Cybernetics*, 47(3):744–758, 2017.