Advanced Topics in Distributed Systems

Critical review by Zainab Abbas

This report contains critical review of the papers presented in the Advanced Topics in Distributed Systems course.

Selected Papers:

- **Paper 1:** Mirhoseini, A. et al, 2017. *Device placement optimization with reinforcement learning*. ICML'17.
- **Paper 2:** Nazi, A. et al., 2019. *GAP : Generalizable Approximate Graph Partitioning Framework*.
- **Paper 3:** Isabelle Stanton and Gabriel Kliot. 2012. *Streaming graph partitioning for large distributed graphs*. ACM SIGKDD.

Justification:

Optimising the processing of huge volumes of data in a distributed setting can be achieved by partitioning the data or the model. Data partitioning involves partitioning of data across various compute nodes (workers) and then deploying data processing applications, such as, deep learning models and graph processing applications, over these partitions. On the other hand, for model partitioning the model operations are placed across the computer nodes (workers). The papers chosen consist of two papers that focus on model partitioning. In particular, partitioning deep learning models over a cluster of heterogeneous machines. The other paper is about data partitioning with streaming graphs used as the input data representation. All approaches used in the papers aim to optimise the performance of data processing applications.

Review for Paper 1

1. Significance:

This work proposes a method to optimize the training time of neural networks. Inorder to meet the growing computation requirements of neural networks, heterogeneous distributed environments containing CPUs and GPUs are being used. Before the device placement method proposed in this work, human experts' knowledge and heuristics were being used to distribute the neural network operations across the devices. This method proposed a machine learning based solution to optimize the placement of TensorFlow computational graphs.

2. Contribution:

- The work proposes a novel method based on reinforcement learning approach to distribute the TensorFlow computational graph operations across the devices in a heterogeneous distributed setup.
- The proposed solution finds placements that outperforms baseline solutions for Image classification on Inception-V3 and for neural machine translation and language modeling on LSTMs.

3. Solutions:

The work uses a sequence-to-sequence model to find placements of operation in the neural network graphs. The model consists of an encoder RNN and a decoder RNN. The operations of the input graph are first fed to the encoder RNN in sequence. Each input operation is embedded first, the embedding contains the operation type, input shape and adjacency list based on dependencies of the operations. At each timestep, the decoder RNN outputs the device for the operation at the encoder.

The reward signal to the model is based on the execution time of the proposed placement.

4. Experimental Quality:

The paper contains experiments on three well known deep learning networks. The experimental setup and metrics measured in the work are clearly presented and are justified based on the overall goal of the paper in terms of improvement of the training time for deep learning models.

The baseline approaches are also well explained and are justified for the comparison. Overall, the experimental section is very clear and beautifully written. The placement diagrams make the work very visual and easy to understand.

Review for Paper 2

1. Significance:

The work proposes a deep learning based graph partitioning framework. The aim of the proposed method is to reduce the graph cuts and balance the load across the machines in a distributed setup. To the best of our knowledge it is one of the first works to use graph embeddings and deep learning for graph partitioning. The embedding learning is further general enough to partition unseen graphs. The model is tested on various neural network graphs and random graphs.

2. Contribution:

- The work proposes a novel deep learning and graph embedding based graph partitioning framework.
- The proposed method produces a generalized model during training, which can be used to partition similar unseen graphs during inference.

3. Solutions:

The framework used in this work uses Graph Neural Networks to learn the embeddings of the underlying graph and then create a partitioning network with an objective to minimize cuts and balance the load. The graph embeddings are learned using the GraphSAGE method with a loss function based on minimizing cuts. The embeddings are created using degree information of the vertices, the adjacency matrix and node features.

Once the embeddings are learned, they are used to train the partitioning module that generates the probabilities for placing each node to the partitions.

4. Experimental Quality:

The experimental section in my view is not very strong. The claims of creating a generalized model for partitioning unseen graphs is not well justified specially for random graphs, which are synthetically generated. The authors should have tested the model with real-world power law based graphs to show a strong contribution in the graph partitioning research area.

The paper lacks the performance results. Once the partitioning is done, what happens next is important. For example, if a neural network graph is partitioned, did the training time of the application improve based on the resulting partitions ? How good is the partitioning time compared to the baselines ?

Review for Paper 3

1. Significance:

The paper introduces streaming graph partitioning methods for distributed graph processing. It is a novel work to introduce online partitioning compared to classical offline graph partitioning, e.g., *Metis*. The partitioning methods proposed in this work are aimed at minimizing the graph cuts and balance the load across the compute nodes in a distributed graph processing setup. The results show significant improvement compared to offline methods, in terms of cuts, load balance and the graph application run time. The application tested in this work is PageRank.

2. Contribution:

- The paper proposes novel online graph partitioning methods that do not require pre-processing over the complete input graph.
- The proposed method improves the graph application, e.g., PageRank, runtime compared to baseline.

3. Solutions:

The work proposed various heuristics for streaming graph partitioning. The proposed heuristics receive a stream of vertices as input and place the vertices in the given set of partitions with an aim to balance the load and keep the number of edge-cuts minimum.

The heuristics keep some state in memory, such as, previous assignments of vertices and load of the partitions, to make decisions for partitioning of incoming streams of vertices.

4. Experimental Quality:

The paper contains detailed experiments on graph partitioning that compare the proposed methods with the state-of-the-art *Metis,* in terms of cut and load balance for various real-world graph datasets. The experiments also include results for graph streams in various order, and the impact of partition number on the partitioning quality. Finally, a graph application, PageRank, is tested to demonstrate the effect of partitioning on the overall runtime of the application.

The experimental setup and metrics measured in the paper are well explained and justify the results.