FID3008 Advanced Topics in Distributed Systems
Review of Presented Papers

Sina Sheikholeslami

April 2020

1 Introduction: Device Placement for Distributed Deep Learning

Device placement can be defined as the problem of learning to partition a dataflow graph across a set of available devices. In the domain of deep learning, using such strategies and policies enables us to, e.g., partition large models across several processing units (e.g., GPUs) to train them in an efficient, distributed manner. A suitable placement strategy, if found, can both decrease the training time, and increase the resource efficiency, while also giving us the option to train very large models that do not fit on a single GPU.
However, since the dataflow graph of large models can include billions of trainable parameters and many operators, and the run time of a dataflow graph depends on a variety of settings such as the degree of parallelism, the number of nodes, and the network links between devices and nodes, finding a suitable placement strategy becomes a non-trivial task.
The selected papers discuss new approaches to device placement that do not require human intervention. The first two papers, describe Placeto (Addanki et al. 2019) and GDP (Zhou et al. 2019) which use reinforcement learning (RL) to learn placement policies. On the other hand, (Wang, Huang, and Li 2019) introduces Tofu, a system that uses a recursive, dynamic-programming based approach to partition the dataflow graph. All three papers provide interesting ideas as well as directions for further investigation. As all the papers contain several ideas and proposals, in the following I will briefly reflect on the ones that I found more interesting.

2 Placeto

Placeto is a RL-based approach to find *generalizable* device placement policies, meaning that the resulting policy can be applied to any computation graph. The key ideas presented in this work are 1) modeling the device placement task as finding a sequence of iterative placement improvements, and 2) using graph embedding to capture topological properties of the dataflow graph.
As Placeto in based on RL, it uses a Markov Decision Process, where each state corresponds to the current placement of operator groups of the dataflow on a set of devices. The initial state is a random assignment of operator groups to devices. An action, in turn, is assigning one operator group on a device, which leads to a new placement as the state. This process is shown in Figure 1.

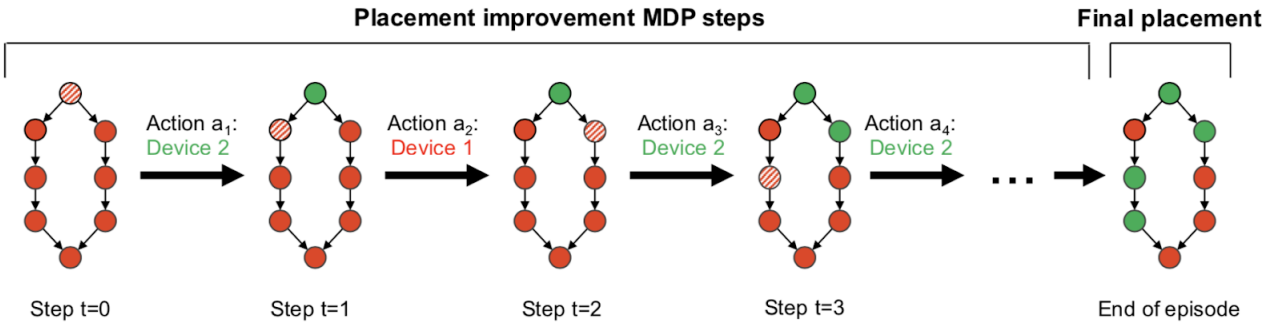

Figure 1: At each intermediate step, Placeto places one operator group on a device. Image taken from (Addanki et al. 2019)

The authors propose two approaches for assigning rewards. In the first approach, a zero reward is given at

each intermediate step, and a reward equal to the negative run time of the final placement is given at the terminal step (the end of an episode). The second approach assigns intermediate rewards equal to the run time improvement of the new state compared to the previous one. Although having intermediate rewards means the run time for each step has to be determined, the authors show that this approach leads to lower variance in run times between episodes.
Placeto uses a graph neural network to capture topological properties of the dataflow and process the raw features associated with each node (operator group), such as its total run time and its current placement. The overall embedding approach consists of three steps: 1) computing per-group attributes, 2) local neighborhood summarization (using message passing), and 3) pooling summaries. The output of the graph neural network is then given as input to the policy network, which in turn outputs placement probabilities leading to a new placement. The end-to-end process is shown in Figure 2.

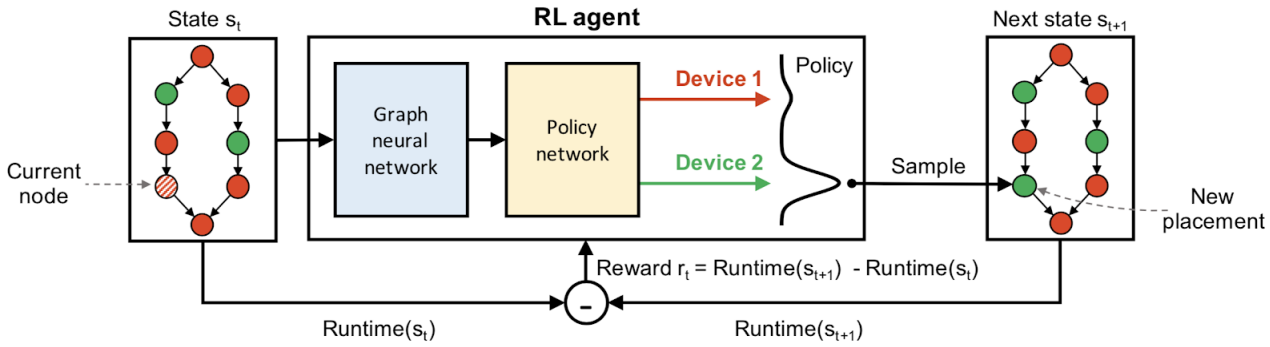

Figure 2: Placeto's end-to-end placement process. Image taken from (Addanki et al. 2019)

The authors' main evaluation of Placeto concerns its performance (run time of the best placement found, and the time taken to find that best placement) and generalizability (the performance of a policy for unseen graphs from the same families). The results in both aspects seem to be promising and experiments are explained in good detail in the paper. However, the generalizability of a learned policy is limited to graphs that are structurally similar with the family of graphs used for training.
A notable shortcoming of Placeto is that the operator groups have to be formed beforehand, and they use a reinforcement learning approach proposed by (Mirhoseini et al. 2017) to do that. Also, all the operators in a single group will be placed on one node, which further makes the final result sensitive to the initial groupings. One interesting area for investigation can be trying different graph embedding approaches.
Overall, this is a well-written paper, and the appendix section contains further implementation and theoretical details to understand their approach.

3 GDP

GDP, short for Generalized Device Placement, is an end-to-end device placement network that can generalize to arbitrary graphs, therefore improving on the generalizability aspect compared to Placeto. Another key contribution of the paper is the fact that GDP does not require an explicit operator grouping stage before the placement, and uses a placement network with recurrent attention mechanism for this. It also provides the possibility to pre-train the policy on a batch of graphs and then fine-tune it for a specific graph structure. To this end, the RL objective in GDP is defined to simultaneously reduce the expected run time of the placements over a set of dataflow graphs, rather than individual training per graph.
The policy network in GDP consists of a graph-embedding network that again encodes the graph structure and features of nodes into a trainable graph representation. For this, they use GraphSAGE (Hamilton, Ying, and Leskovec 2017) These node embeddings are then fed to a Transformer-based placement network that outputs the device placement probabilities. The overall process for GDP is shown in Figure 3.
The paper includes a rich set of experiments, but key details that might be necessary to fully understand the results are missing. Again, they mainly focus on performance and generalizability.

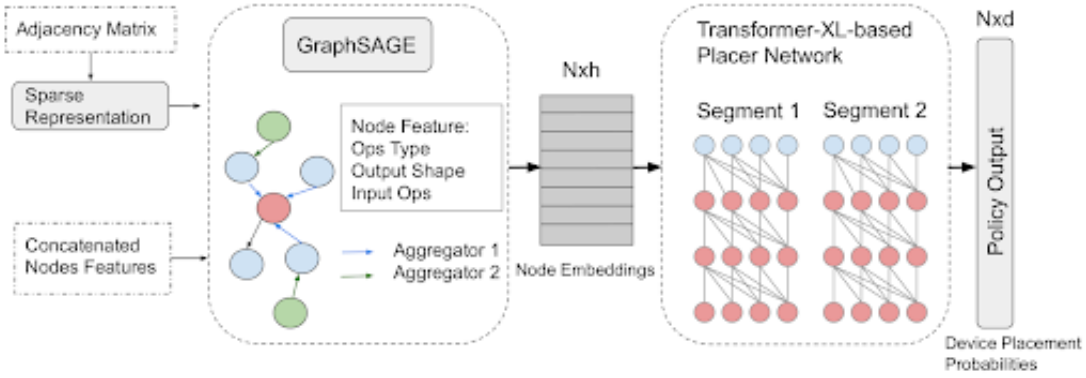

Figure 3: An overview of GDP, where it uses GraphSAGE to model the dependencies between the operations and build a general, end-to-end device placement for a wide set of dataflow graphs. Image taken from (Zhou et al. 2019)

4 Tofu

Tofu (Wang, Huang, and Li 2019) takes a different approach compared to GDP and Placeto: it aims for both fine-grained tensor partitioning as well as partitioning the overall dataflow graph on a set of devices. Fine-grained partitioning in this context refers to partitioning the input/output tensors and parallelizing the execution of individual operators.

To enable partitioning of individual operators, the authors propose Tensor Description Language (TDL), a high-level specification of operators that explains the way an output tensor is derived from its input (rather than "how" it is implemented). These specifications can be statically analyzed using symbolic interval analysis. The actual partitioning of an operator is done using a technique called partition-n-reduce, in which the symbolic intervals resulting from TDL-based analysis are used to partition the operator along two workers.

Tofu attempts to minimize the total communication cost as a proxy to optimizing end-to-end execution time and per-worker memory consumption. For partitioning the overall dataflow graph, Tofu use the dynamic programming algorithm proposed by (Jia et al. 2018). To enhance this algorithm, Tofu uses graph coarsening to shrink the search space and make it linear, and uses recursive search, where in each recursive step it partitions each tensor among two worker groups. An example of the recursive partitioning is shown in Figure 4.

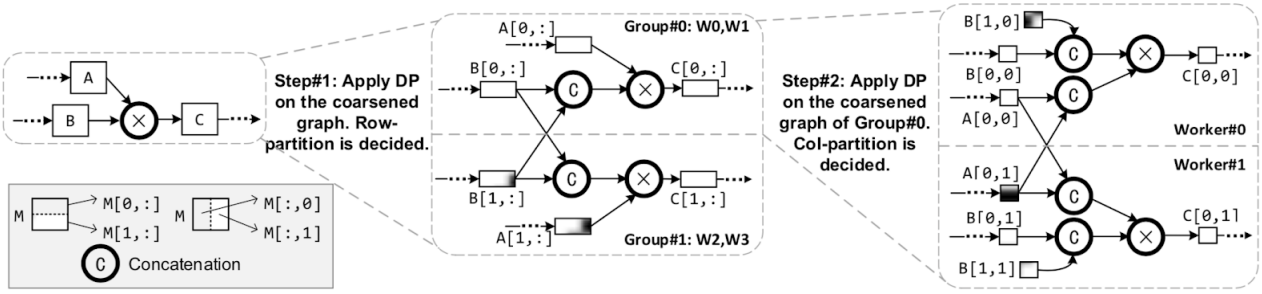

Figure 4: An example of recursive partitioning of a matrix multiplication on four workers. Image taken from (Wang, Huang, and Li 2019)

It should be mentioned that the authors have implemented Tofu to work with MXNet, but they claim the approach is generalizable enough to be applied to other frameworks such as TensorFlow.

Regarding the evaluation and experiments, again this paper includes a rich set of experiments but the descriptions are not that clear. Also, the experiments are performed on a single machine with multiple GPUs rather than multiple machines with GPUs, due to the considerable differences between intra-node and inter-node network bandwidths. Speaking of the quality of the text, the paper is very well written up until to the point where the authors attempt to describe the details of overall dataflow graph partitioning. From there, it becomes hard to follow, and some important details appear to be missing.

I find the idea of symbolic interval analysis simple and interesting, and partition-n-reduce also seems to be promising. However, as the authors themselves indicate, some operators cannot be expressed in TDL, as it does

not support loops or recursions. However, the authors claim that this has been a design decision. Also, there is no point in using Tofu for moderately sized models that can fit in the memory of a single GPU, and the authors claim that in such scenarios the resulting placement strategies are no better than data parallelism.

References

Addanki, Ravichandra et al. (2019). "Placeto: Learning generalizable device placement algorithms for distributed machine learning". In: *arXiv preprint arXiv:1906.08879*.

Hamilton, Will, Zhitao Ying, and Jure Leskovec (2017). "Inductive representation learning on large graphs". In: *Advances in neural information processing systems*, pp. 1024–1034.

Jia, Zhihao et al. (2018). "Exploring hidden dimensions in parallelizing convolutional neural networks". In: *arXiv preprint arXiv:1802.04924*.

Mirhoseini, Azalia et al. (2017). "Device placement optimization with reinforcement learning". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 2430–2439.

Wang, Minjie, Chien-chin Huang, and Jinyang Li (2019). "Supporting very large models using automatic dataflow graph partitioning". In: *Proceedings of the Fourteenth EuroSys Conference 2019*, pp. 1–17.

Zhou, Yanqi et al. (2019). "GDP: Generalized Device Placement for Dataflow Graphs". In: *arXiv preprint arXiv:1910.01578*.