GA

ML

CI

BI

# ACCELERATING DEEP NEURAL NETWORK TRAINING WITH MODEL-AWARE NETWORK SCHEDULING

**Allocation:** Exploratory/40 Knh
**PI:** Roy H Campbell[1]
**Co-PIs:** Sayed Hadi Hashemi[1], William Gropp[1]

[1]University of Illinois at Urbana-Champaign

## EXECUTIVE SUMMARY

Deep Neural Networks (DNNs) form the crux of advanced solutions in a variety of domains such as computer vision and natural language processing. The increasing complexity of these problem domains necessitates large-scale, distributed training of the associated DNNs. Today, performance and scalability of distributed DNN training are bottlenecked by iterative computations among nodes. In this work, we took a more in-depth look at communications in DNN workloads.

- We proposed a performance model for DNN workloads and used the model to explain the lesser-known performance bottlenecks in these workloads.
- Our benchmark results showed that while high-performance networks such as Gemini substantially reduce the communication time, they may decrease overall performance by reducing the overlap compared to commodity networks.
- Lastly, we developed Caramel, a model-aware approach to take advantage of faster networks while achieving the highest performance.

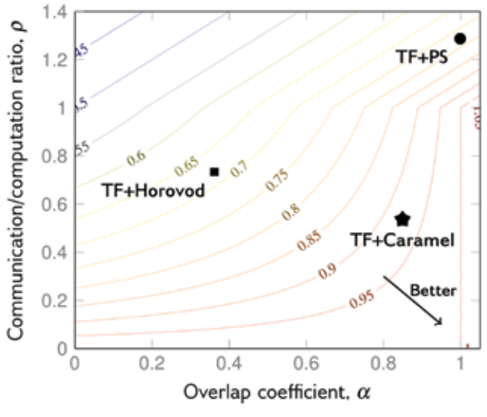


Figure 1: Overlap coefficient and communication/computation ratio for different communication methods with GPU utilization contours in the background (using Inception-v3 with eight workers). Our system (Caramel) versus TensorFlow Default (PS) versus TensorFlow and MPI (Horovod).

## RESEARCH CHALLENGE

In frameworks such as TensorFlow [1] or PyTorch, the computation and communication involved in training are represented using dataflow graphs, which are directed acyclic graphs (DAGs). The state of the DNN is represented by a vector of parameters. Each iteration involves the computation of parameter updates, followed by their exchange among the participating nodes.

Currently, performance and scalability of distributed DNN training is bottlenecked by this parameter aggregation [2,3]. At a high level, there are two causes of these network inefficiencies during aggregation. First, the most commonly used pattern of parameter aggregation relies on a centralized server, Parameter Server (PS), and an all-to-one/one-to-all communication pattern, causing incast/outcast at the PS and limiting the goodput (sustained performance) of the system. Second, the scheduling of operations is such that training computation blocks on the network-intensive aggregation leaves GPUs idle. The increasing sizes of training data sets and DNN models and, therefore, increasing the number of machines involved in training, is likely to reduce the overall inefficiency.

In this work, our goal was to minimize network bottlenecks in distributed DNN training to reduce the iteration time and increase GPU utilization. Toward this goal, we designed and built Caramel. The intuition behind Caramel is that while some related incast/outcast problems may be found in data analytics and graph-processing systems, DNN training offers a particular opportunity for network optimization since it has a more predictable environment with fixed parameter sizes and nearly identical iterations. Caramel applies this idea by: (1) analyzing a variety of aggregation patterns and choosing the appropriate aggregation pattern for a given environment and model, which we refer to as "network-aware optimization"; and (2) leveraging an understanding of the operations within model training to improve communication/computation overlap, which we refer to as "model-aware optimization."

## METHODS & CODES

To achieve performance improvement through network-aware optimization, we analyzed all-to-one and decentralized data aggregation techniques such as the bucket [4] and halving–doubling (HD) [5] algorithms. The bucket algorithm and HD offer better network load distribution across workers compared with PS and naive all-to-all communication. However, these techniques incur the cost of multiple stages per transfer and require synchronization across workers during aggregation. With Caramel, we showed that the decentralized aggregation patterns such as the bucket algorithm and HD can work well in DNN applications when coupled with model optimizations that enable better synchronization among workers.

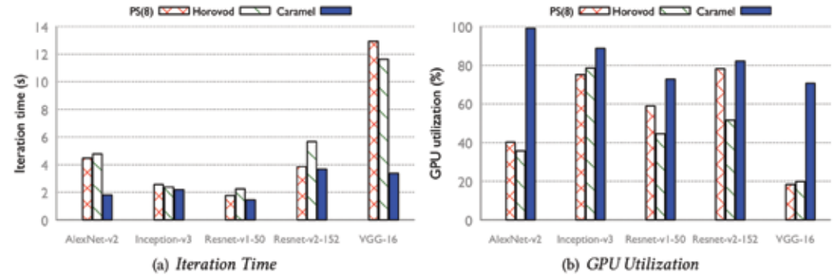


Figure 2: Performance comparison of our system (Caramel) versus TensorFlow Default (PS) versus TensorFlow and MPI (Horovod).

To understand the opportunity for network acceleration through model-aware optimization, we investigated dataflow models associated with 16 DNNs and identified common model characteristics that enable efficient network transfers. We found that the same model can result in network transfers activated (i.e., parameters being ready for aggregation) in different orders across multiple workers. This can prove detrimental to decentralized aggregation where all workers should activate the same parameter before the transfer is initiated. To solve this problem, we enforced ordering in network transfers by adding additional dependencies in the DAG to force all workers to activate network transfers in the same order.

All DNNs we analyzed have a large number of small parameters that incur significant overhead during network transfer. To tackle the small-parameter overhead, we implemented model-aware batching in Caramel, while ensuring that the batched parameters were ready at nearly the same time to avoid waiting. In addition, we identified the opportunity for increasing the window of network transfer during an iteration of DNN training without delaying the training. An iteration has two phases: forward pass and backward pass. Currently, transfers are restricted to the backward pass. We proposed techniques for extending network transfers to the forward pass in Caramel, thereby increasing the overlap of communication and computation.

We also implemented Caramel over TensorFlow and demonstrated that the iteration time can be reduced by up to 3.8 times without changing DNN training functionally, with the communication time reduced by at least two times in all networks.

## RESULTS & IMPACT

We analyzed a variety of DNN models on TensorFlow with the commonly used Parameter Server implementation, TensorFlow with Horovod [6], which relies on MPI all reduce implementation of the bucket algorithm and halving–doubling, and our method. We conducted our experiments on both the Blue Waters network as well as a public cloud (Azure) with commodity networks.

In Fig. 1, we observe that TensorFlow with Parameter Server offers high overlap. However, the communication cost of PS is very high. The Horovod implementation, on the other hand, reduces the communication cost with an efficient aggregation pattern. However, these decentralized patterns suffer from a poor overlap of communication and computation. As a result, PS with a higher communication cost achieves better GPU utilization in this instance. Ideally, we want to be in the bottom right region of this plot to achieve high utilization.

In Fig. 2, we report the iteration time and GPU utilization with 16 workers for Caramel, PS with eight servers co-located with the workers, and Horovod for five representative networks. We observe that Caramel can improve the iteration time by up to 3.84 times (in VGG-16) and GPU utilization by up to 2.46 times (in AlexNet-v2). We also find that performance gains are not equal across networks. The speedup is higher when communication time is much higher than computation time. In compute-intensive networks such as Inception-v3 and Resnet-v2-152, network optimization results in minimal improvement in iteration time and GPU utilization compared with PS.

## WHY BLUE WATERS

Blue Waters' excellent platform makes it easy to conduct large-scale exploration to find potential performance opportunities. Furthermore, the vibrant community of Blue Waters users and staff helped us to get up to speed faster using past experiences on other systems.

## PUBLICATIONS & DATA SETS

Hashemi, S.H., S.A. Jyothi, and R. Campbell, On The Importance of Execution Ordering in Graph-Based Distributed Machine Learning Systems. *SysML* (2018).

Hashemi, S.H., S.A. Jyothi, and R. Campbell, Network Efficiency through Model-Awareness in Distributed Machine Learning Systems. *USENIX NSDI* (2018).