SwitchML: Scaling Distributed Machine Learning with In-Network Aggregation

Amedeo Sapio [*1] Marco Canini [*1] Chen-Yu Ho [1] Jacob Nelson [2] Panos Kalnis [1] Changhoon Kim [3]
Arvind Krishnamurthy [4] Masoud Moshref [3] Dan R. K. Ports [2] Peter Richtárik [1]

Abstract

SwitchML is a system for distributed machine learning that accelerates data-parallel training using P4 switches. SwitchML uses in-network aggregation to reduce data transfer during synchronization phases of training. By co-designing the software networking end-host stack and the switch pipeline, SwitchML greatly reduces the volume of exchanged data and the latency of all-to-all communication, speeding up training by up to 300%.

Distributed machine learning has become common practice, given the increasing complexity of DNN models and the sheer size of real-world datasets. While GPUs and other accelerators have massively increased per-node compute power, networks have not improved at the same pace. As a result, in large deployments, distributed ML training is increasingly a network-bound workload [2]. To prevent the network from becoming a bottleneck, we propose SwitchML, a rack-scale system for distributed ML training that uses programmable switches [1] to reduce both latency and the amount of data transferred.

The network bottleneck for training workloads arises from the need to periodically distribute model updates in order to synchronize workers. This communication is usually done with collective operations (i.e., *all-reduce*) or enabled by parameter servers. Both require many CPU cycles to aggregate updates and are oblivious to the physical network topology. SwitchML co-designs the switch processing logic, end-host protocols, and ML frameworks to offload aggregation directly to switches. This reduces the amount of data transmitted during synchronization phases, speeding up training time.

SwitchML must solve three main problems: limited computational power, maintaining synchronization, and dealing with failures. We address these challenges by appropriately dividing the functionality between hosts and switches, resulting in an efficient and reliable streaming aggregation protocol.

In SwitchML (Figure 1), the switches do aggregation computations, while end-hosts manage reliability and do more complex computations. Workers stream updates to the switch's aggregation buffer pool; they determine when buffers can be used, reused, or require failure handling. Each slot in the pool aggregates a vector of k integers, delivered simultaneously in one update packet. The aggregation function is addition, which is commutative and associative, so the result doesn't depend on packet arrival order. We tolerate packet loss using a switch scoreboarding mechanism and a retransmission mechanism driven solely by end-hosts, which together ensure that workers operate in lock-step, maintaining aggregation throughput.

[*]Equal contribution [1]KAUST [2]Microsoft Research [3]Barefoot Networks [4]University of Washington.

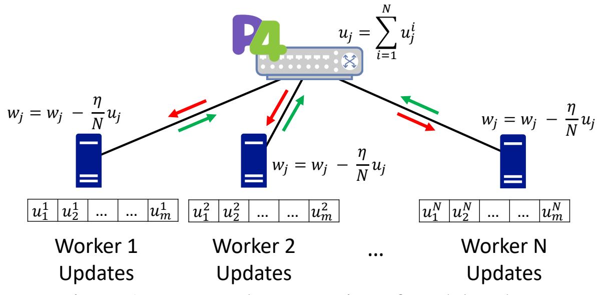
$u_j = \sum_{i=1}^{N} u_j^i$
$w_j = w_j - \frac{\eta}{N} u_j$
$w_j = w_j - \frac{\eta}{N} u_j$
$w_j = w_j - \frac{\eta}{N} u_j$
u_1^1 u_2^1 ... ... u_m^1
u_1^2 u_2^2 ... ... u_m^2
u_1^N u_2^N ... ... u_m^N
Worker 1 Updates
Worker 2 Updates
...
Worker N Updates

Figure 1. In-network aggregation of model updates.

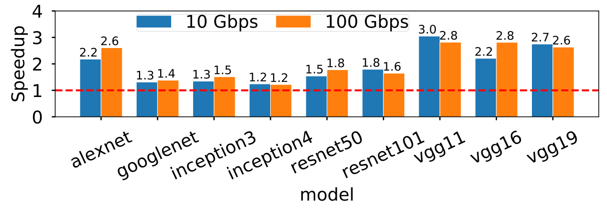

Figure 2. Training speedup on a 10 (blue) and 100 (orange) Gbps network with 8 workers. Results are normalized to NCCL (fastest TensorFlow baseline). SwitchML speeds up training by up to 300%.

Since switches don't support floating-point operations, we convert floating-point values to 32-bit fixed-point values, processed by the switch as integers. We quantize updates by multiplying by a scaling factor, chosen by profiling to avoid truncation and overflow. Both theoretical and experimental analysis show that this quantization neither significantly affects the training time nor ultimate model accuracy.

Our approach is not tied to a single ML framework; it comprises a P4 switch program and an end-host C++ library providing a familiar all-reduce API. We have integrated SwitchML with TensorFlow using Horovod and Caffe2 using Gloo.

We analyze the performance of SwitchML using standard benchmarks on popular CNNs in TensorFlow, trained over ImageNet (except for AlexNet, which uses synthetic data). We trained on 8 machines using Horovod, each with 1 NVidia P100 16 GB GPU, dual Xeon E5-2630 v4's, 128 GB RAM, and dual Intel 82599ES 10Gbps and Mellanox Connect-X 5 100Gbps NICs. We use a 64 x 100 Gbps switch with Barefoot Networks' Tofino chip [1]. Figure 2 shows the training performance speedup compared to TensorFlow using the NCCL library. Overall, SwitchML's speedups range between 20%-300%. As expected, different models benefit from in-network aggregation differently, depending on how network-bound they are.

Our talk will discuss SwitchML's design and implementation, and describe future challenges for scaling it up and deploying it in a multi-tenant environment.

REFERENCES

[1] Barefoot Networks. Tofino. https://barefootnetworks.com/products/brief-tofino/.

[2] L. Luo, J. Nelson, L. Ceze, A. Phanishayee, and A. Krishnamurthy. PHub: Rack-Scale Parameter Server for Distributed Deep Neural Network Training. In *SoCC*, 2018.