**MPI-ML: A High-Performance Sparse Communication Layer for Machine Learning**

Cedric Renggli  
ETH Zurich  
Dan Alistarh  
IST Austria  
Torsten Hoefler  
ETH Zurich

### Communication in Machine Learning Workloads

Data-parallel Machine Learning  
- **Data shared among multiple compute nodes**  
- **Set of parameters (model) is consistent across machines**  
- **Communication and Synchronization are often bottlenecks**  

**Example:**

- **Challenge:** Reduce communication and synchronization overheads  
- **Idea:** Exploit robustness to asynchrony and quantization noise

**MPI-ML: An Overview**

- **Main Idea**
  - Efficient support for sparse communication  
  - MPI-like operation semantics on top of sparse streams  
  - Allow non-blocking operations and quantized communication

- **Efficient Sparse Collectives: the Static Case**
  - **Setting:** In this case, the size of the resulting data remains below the threshold $d$, where this sparse representation is still efficient.
  - **Case 1:** Latency-Dominated  
    - Use recursive doubling in the 4th round, nodes that are distance $2^d$ apart exchange all data.  
    - Runtime: $\alpha g P + \frac{d}{2} + \beta P$
  - **Case 2:** Bandwidth-Dominated  
    - Split the dimension into $P$ partitions, and then perform a sparse AllGather:
      - Runtime: $\alpha g P + \frac{d}{2} + \beta P$

- **Efficient Sparse Collectives: General Solution**
  - **The Dynamic Case**  
    - The result becomes dense during addition  
    - We use the solution from the Bandwidth-dominated case  
    - Communicate data in sparse format, then broadcasts dense
  - **Quantized Communication**  
    - We implement QSGD quantization [Alistarh et al.]
    - Allows sent values to be rounded to arbitrary precision  
    - Provable convergence, and provides additional compression
  - **Non-blocking collectives**  
    - Threads can proceed even if the collective has not completed  
    - Allows us to overlap communication and computation

**Performance**

**Target Architectures:**
- **Supercomputing:** CSCS Piz Daint (3rd worldwide, 1st in Europe)
- **Cloud computing:** Amazon EC2 and research cluster

**Target Applications:**
- **Large-scale distributed optimization** (SGD and SCF for regression)
- **Training large neural networks** (CNNs and RNNs)

**Many settings have naturally-sparse communication!**
- **Linear regression via Stochastic Gradient Descent:**
  - **Stochastic gradient:**
    - $\hat{w} (x) = a_i (a^T x_i - b_i)$
  - If samples $a_i$ are sparse, the update is sparse!  
  - Occurs in many real-world datasets.
- **Top-k SGD [Dryden et al., Aji & Heafield]**
  - **Idea:** Only communicate some top percentage of the gradient
  - **Save the rest locally**
  - **Accuracy on CIFAR-10:**

*Figure 4: Data density vs. reduction time for various algorithms, dimensions $S = 1000$ and $P = 5$ nodes.*

*Figure 5: MNIST convergence for various algorithms.*

*Table 1: Speedup (SSAR) using MPI-ML. The bar represents average time in minutes for a full dataset pass, and its communication part of bracket. Speedup ratios show MPI-ML above and with communication timing in brackets.*

*Table 2: System performance using MPI-ML. The bar represents average time in minutes for a full dataset pass, and its communication part of bracket. Speedup ratios show MPI-ML above and with communication timing in brackets.*