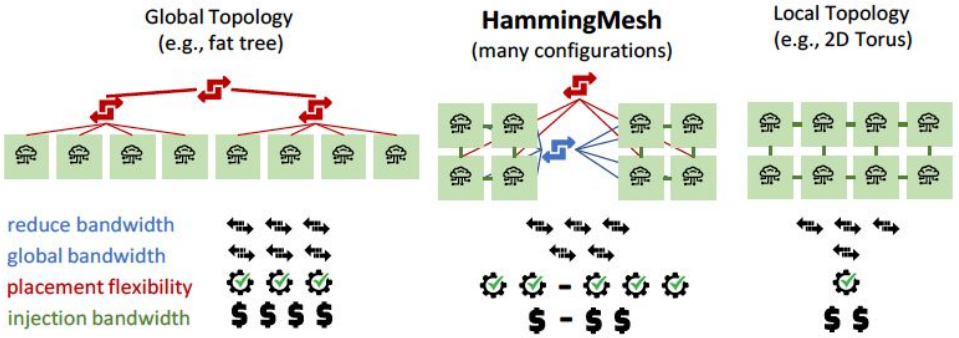
HammingMesh: A Network Topology for Large-Scale Deep Learning

Motivation

- Data-driven programming or software 2.0, is limited by the capability of machines to perform the compute and data-intensive training jobs
- Limited by data movement
- Memory and network bandwidth are expensive
- Today’s HPC networks, optimized for full global (bisection) bandwidth, are inefficient for deep learning workloads

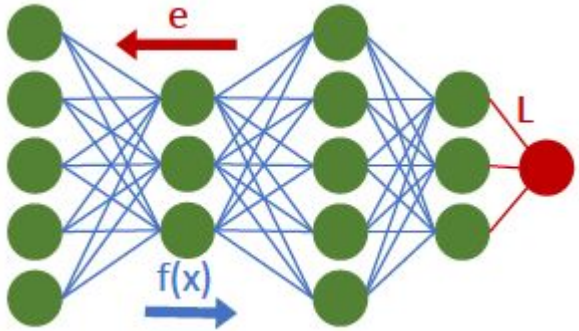
What is a HammingMesh



- A flexible topology that adjusts the ratio of local and global bandwidth for deep learning workloads
- Combines ideas from torus and global-bandwidth topologies (e.g., fat tree) to enable a flexibility-cost trade off
- HammingMesh topology
  - uses technology-optimized local (e.g., PCB board) and global (optical, switched) connectivity
  - utilizes limited packet forwarding capabilities in the network endpoints to achieve lower cost and higher flexibility
  - enables full-bandwidth embedding of virtual topologies with deep learning traffic characteristics
  - supports flexible job allocation even with failed nodes
  - enables flexible configuration of oversubscription factors to adjust global bandwidth

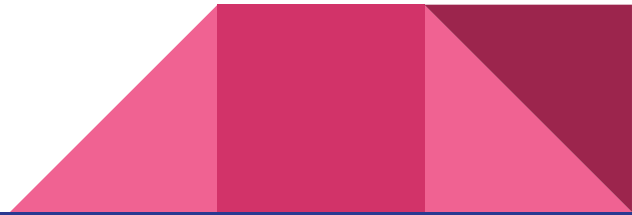
Communication in Distributed Deep Learning

- Deep learning training with Stochastic Gradient Descent
  - The forward pass evaluates the network function f(x) on a set of M examples
  - The backward pass of SGD computes the average loss L and propagates the errors e backward through the network to adapt the parameters P
  - This training process proceeds through multiple (computationally identical) iterations until the model achieves the desired accuracy
- Parallelism and data distribution can fundamentally be arranged along three axes: data parallelism, pipeline parallelism, and operator parallelism.



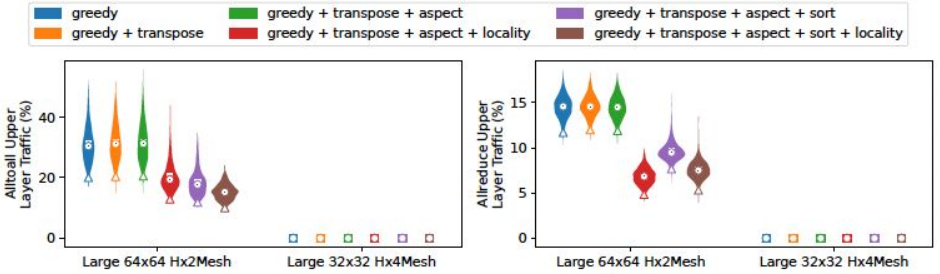
HammingMesh Structure

- Combines local short copper cables with global long fiber cables
- Local groups are formed by local inexpensive high-bandwidth 2D mesh
  - uses short metal traces on PCB boards
- Combines sparsely connected boards in a dimension-wise fully-connected topology
- Boards are connected to a 2D Hamming graph
- Accelerator ports are arranged in planes with 4 directions each



Using HammingMesh in Practice

- Allocating Jobs on HammingMesh
  - 1) Identify all available indexes in each row, resulting in y sets of at most x indexes
  - 2) Set S (“selected”) to the first row with at least v indexes
  - 3) Add another row whose intersection with all rows in S has at least v indexes to S
  - 4) Repeat the last step until S contains u rows, fail if no such set exists
- Experimental Workloads
  - simulate how well a representative job mix that completely fills a full global bandwidth topology



Random Failures in HammingMesh

- Allocation algorithm achieves a median utilization of working boards higher than 70%
- 40 failed boards achieve median utilization of 62% of the small Hx4Mesh
- All meshes with fewer boards are more affected by random failures
- Allocating the jobs in random arrival order decreases he utilization by at most 10% on large networks

Thank You