VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

Multi-GPU with Horovod

Georg Zitzlsberger georg.zitzlsberger@vsb.cz

26-03-2021

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research, Development and Education

MINISTRY OF EDUCATION, YOUTH AND SPORTS

Agenda

Parallelism

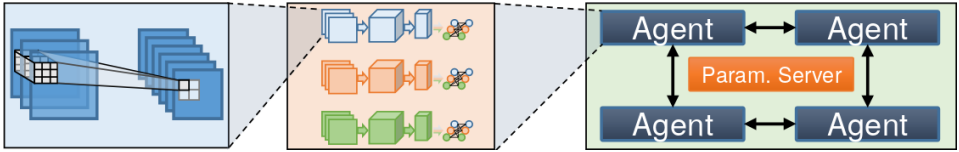
Distributed Training

Horovod

Parallelism

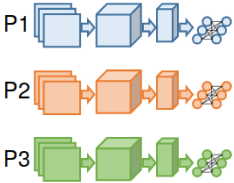
Parallelism can be found in:

- Low level operations in the network
- Parallelized networks
- Distributed training



(Image: Ben-Nun, et al.)

Differnce Data vs. Model Parallelism



- Network layers assigned to different workers
- Every worker trains with the same data
- Activations are exchanged (requires large I/O bandwidth)
- **Enables bigger models**

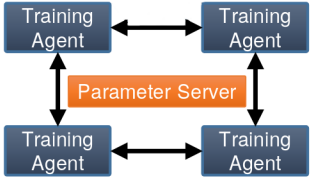
- All workers see the same network
- Every worker trians with different data
- Gradints (weights) are exchanged (averaging to common model)
- Side effect: "sharp" minima
- **Enables faster training**

(Images: Ben-Nun, et al.)

Distributed Training

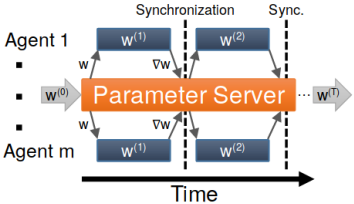
What if one node (GPU) is not enough?

- Model Consistency
- Parameter Distribution and Communication:
  Centralization and Compression (e.g. FP16)
- Training Distribution:
  Model Consolidation and Optimization Algorithms

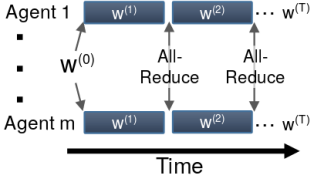


(Image: Ben-Nun, et al.)

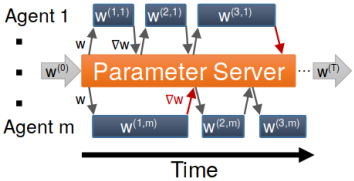
Distributed Training: Model Consistency



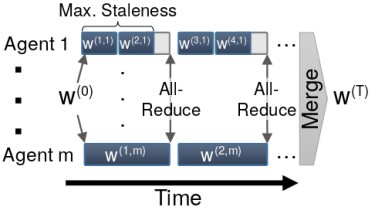
(a) Synchronous, Parameter Server



(b) Synchronous, Decentralized



(c) Asynchronous, Parameter Server



(d) Stale-Synchronous, Decentralized

(Image: Ben-Nun, et al.)

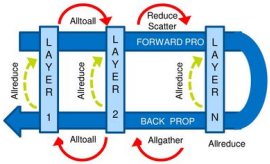
Distributed Training: Libraries

Different backends:

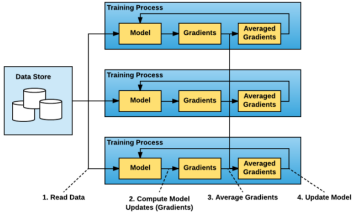
- Message Passing Interface (MPI)
- NVIDIA Collective Communications Library (NCCL)
- Intel oneAPI Collective Communications Library (oneCCL) ▸ Intel oneCCL
  ~~Intel Machine Learning Scaling Library~~ ▸ Intel MLSL
  (both use MPI)

Strategies vary among frameworks:

- ▸ Horovod for Tensorflow/Keras, PyTorch and MXNet (NCCL + MPI, or Gloo)
- Tensorflow has different "strategies" (e.g. `MultiWorkerMirroredStrategy`)
- PyTorch supports MPI, NCCL and Gloo (default)

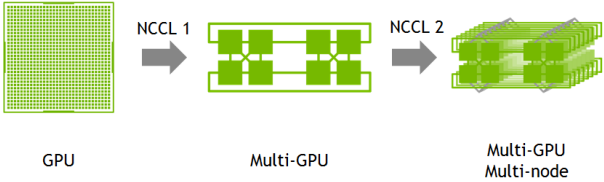


(Image: Intel (MLSL))



(Image: Uber)

NVIDIA Collective Communications Library

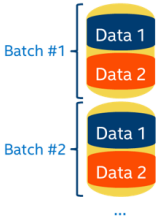


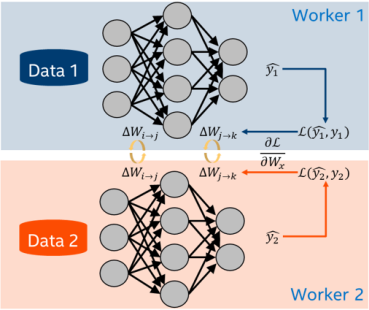
(Image: NVIDIA)

NVIDIA Collective Communications Library (NCCL):

- Similarity to MPI collectives
- Pre NCCL 2.4: ring communication
- NCCL 2.4: double binary tree communication with improved latency
- Supported by major Deep Learning frameworks, e.g. Tensorflow, PyTorch
- Supported by distributed training framework Horovod

Distributed Training: Data Parallelism in Detail





- Batch size limits parallelism
- Scaling batch size requires scaling of learning rate (linearly)

Horovod

- Developed by Uber Engineering
- Part of Michelangelo (Uber's Machine Learning Platform)
- Aimed at and demonstrated for large scale
- Uses MPI based collective communication (synchronous & decentralized)
- Only small code modifications needed
- Supports the most common frameworks:
  - Tensorflow (1.x & 2.0) + Keras
  - Pytorch
  - MXNet

How to Use Horovod

- Horovod comes with a wrapper horovodrun, e.g.:
  `$ horovodrun -np 4 -H server1:2,server2:2 python train.py`
- Different back-ends are possible: MPI, Gloo, NCCL, oneCCL, etc.
- Intel MPI or OpenMPI can be used:
  `$ mpirun -n 4 -ppn 2 -hosts server1,server2 python train.py`

Horovod for KERAS/Tensorflow 2.0

- Add the following:
  - `hvd.init()`:
    Initializes Horovod (and MPI underneath)
  - `hvd.callbacks.BroadcastGlobalVariablesCallback(0)`:
    Initialize model to start with same copies
  - `hvd.DistributedOptimizer(...)`:
    Wrapper around standard optimizer (SGD, Adam, etc.) to enable distributed weight/gradient updates
- Full documentation can be found here

Horovod for KERAS/Tensorflow 2.0

What needs attention:

- If `tf.data.Dataset` is used, consider `shard(num_shards, index)`, e.g.: `my_dataset.shard(hvd.size(), hvd.rank())`
- If training steps are used, instead of number of epochs, adjust the steps, e.g.: `training_steps /= hvd.size()` (assuming perfectly balanced training data)
- If training data size is large, avoid loading it at every worker and divide accross workers
  The same script is executed on all workers!
- Scale the learning rate linearly with the number of workers, e.g.:
  `lr *= hvd.size()`
  See Alex Krizhevsky's paper:
  Strictly speaking it should be `lr *= sqrt(hvd.size())`

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

IT4Innovations National Supercomputing Center

VŠB – Technical University of Ostrava
Studentská 6231/1B
708 00 Ostrava-Poruba, Czech Republic
www.it4i.cz

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research, Development and Education

MINISTRY OF EDUCATION, YOUTH AND SPORTS