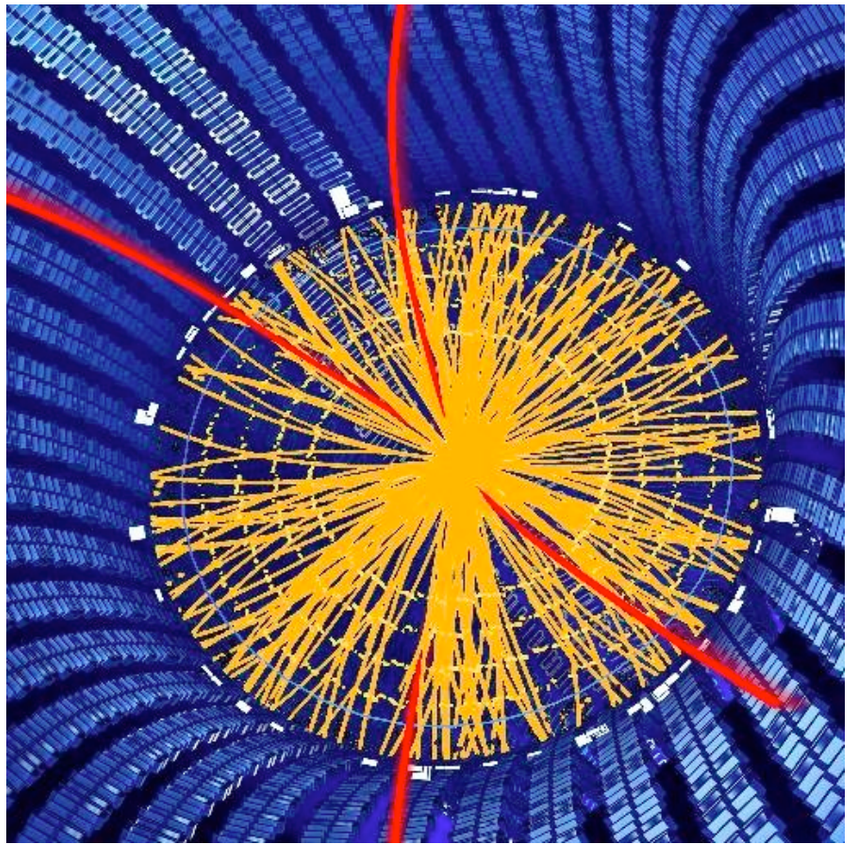
Caltech

Large-Scale Distributed Network Training with MPI



Dustin Anderson
Caltech

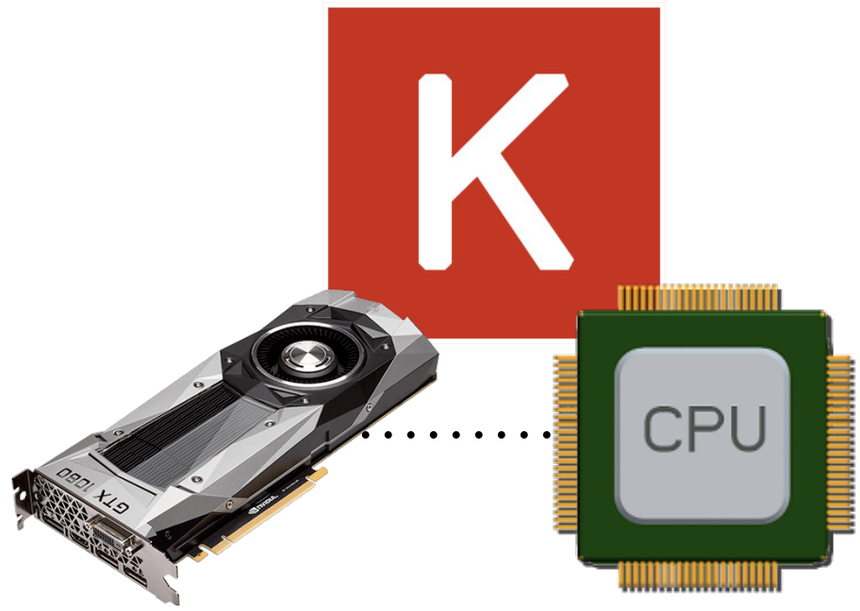
DS@HEP
11 May 2017

Overview

- GPU computation can dramatically speed up neural network training

- Lightweight ML libraries enable fast prototyping of NN models

  ➡ E.g., Keras



- User-friendly support for training with CPU + a single GPU

Overview

- Modern computing resources provide access to many GPUs simultaneously

- Physicists often have access to, e.g., supercomputers with hundreds of nodes

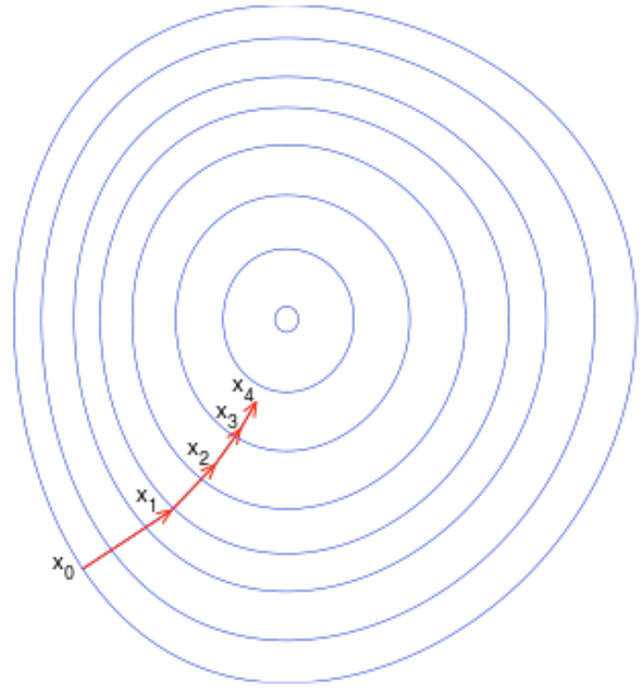
- Would like to maximally exploit these

Neural Network Training

- NN training paradigm: stochastic gradient descent (**SGD**)

- Update rule for model weights $\vec{w}$

$$\vec{w} \to \vec{w} - \eta \nabla Q(\vec{w})$$



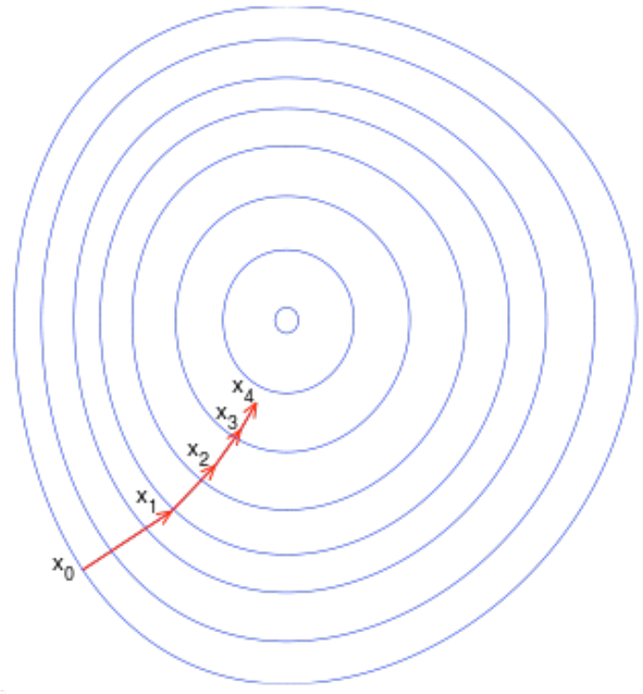
Minimize loss function Q(w)

- Variants: dynamically adjust learning rate η or have separate η for each parameter

  ➡ Adam, RMSProp, Nadam, …

Neural Network Training

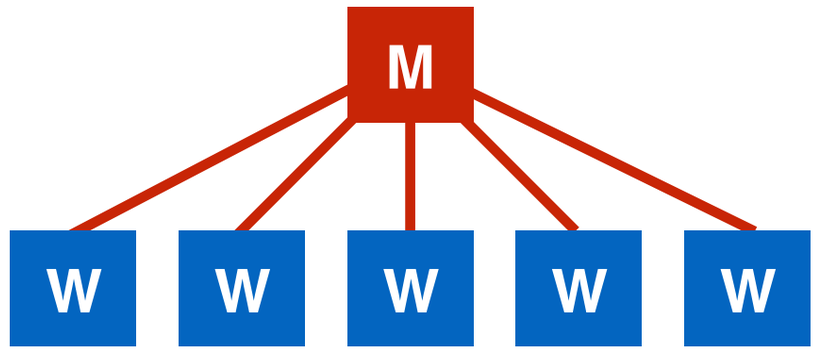
- SGD is inherently sequential
- Two alternating steps:
  1. Compute gradient of loss
  2. Update model weights
- But algorithms exist to parallelize it



Minimize loss function Q(w)

Distributed Training

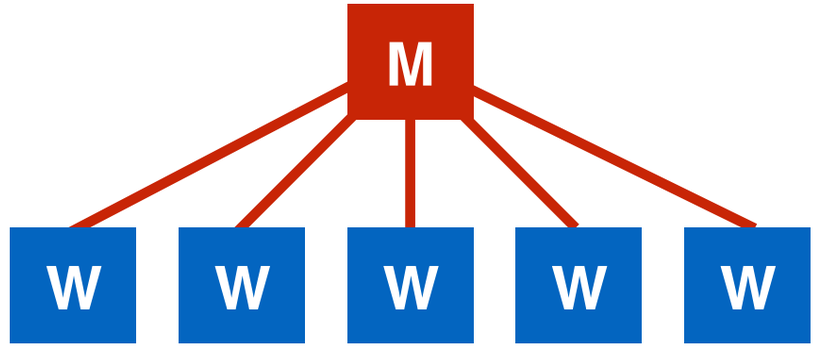
- Paradigm for distributed SGD:
  - ➡ Have N compute nodes
  - ➡ One node acts as “Master”
  - ➡ The others are “Workers”



- Workers can communicate with Master synchronously (all at once) or asynchronously

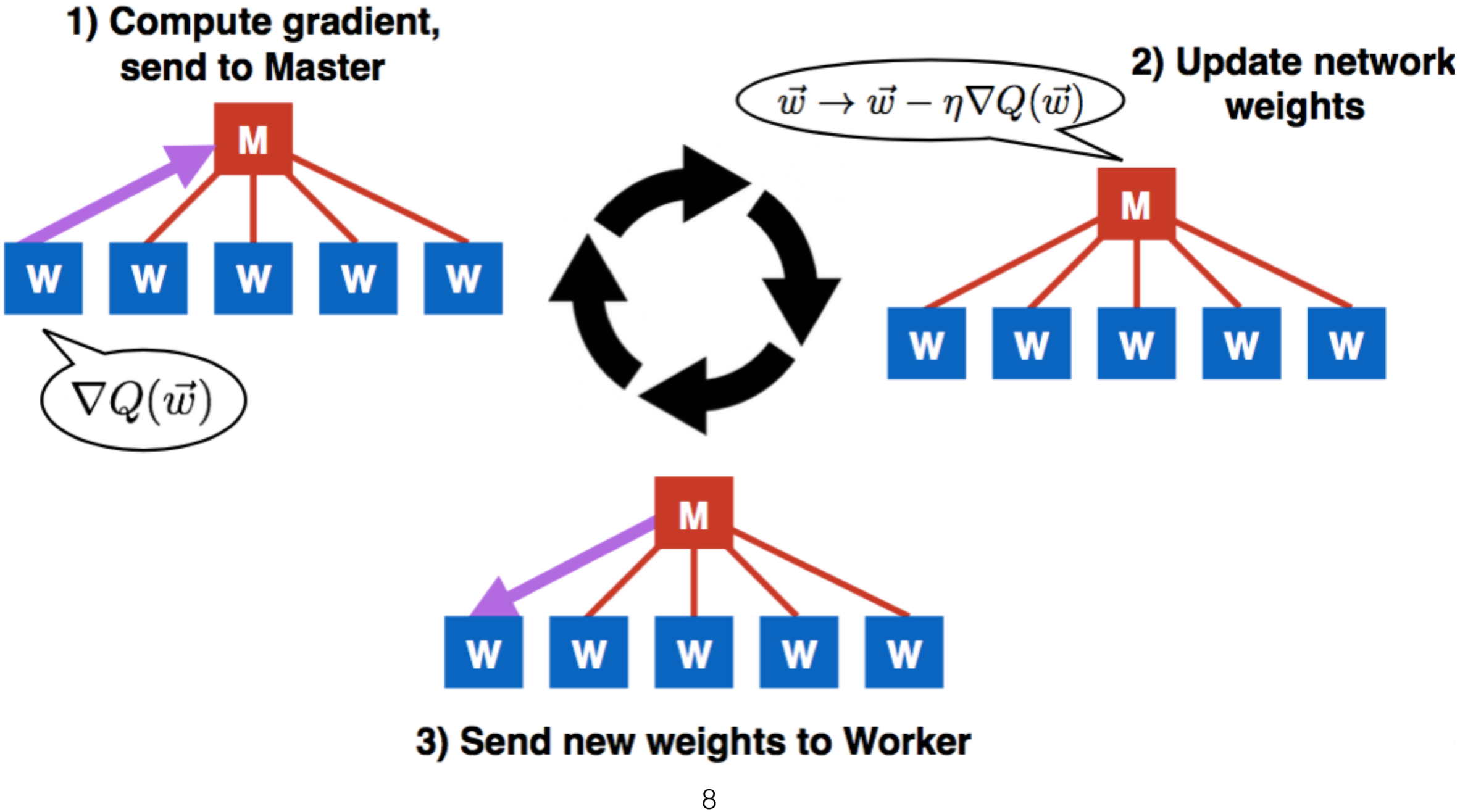
Downpour SGD

- "**Downpour**" is a straightforward algorithm for distributed SGD
- Worker-Master communication is asynchronous
- Master and Workers each have a **local copy of the NN weights** and some training/validation data



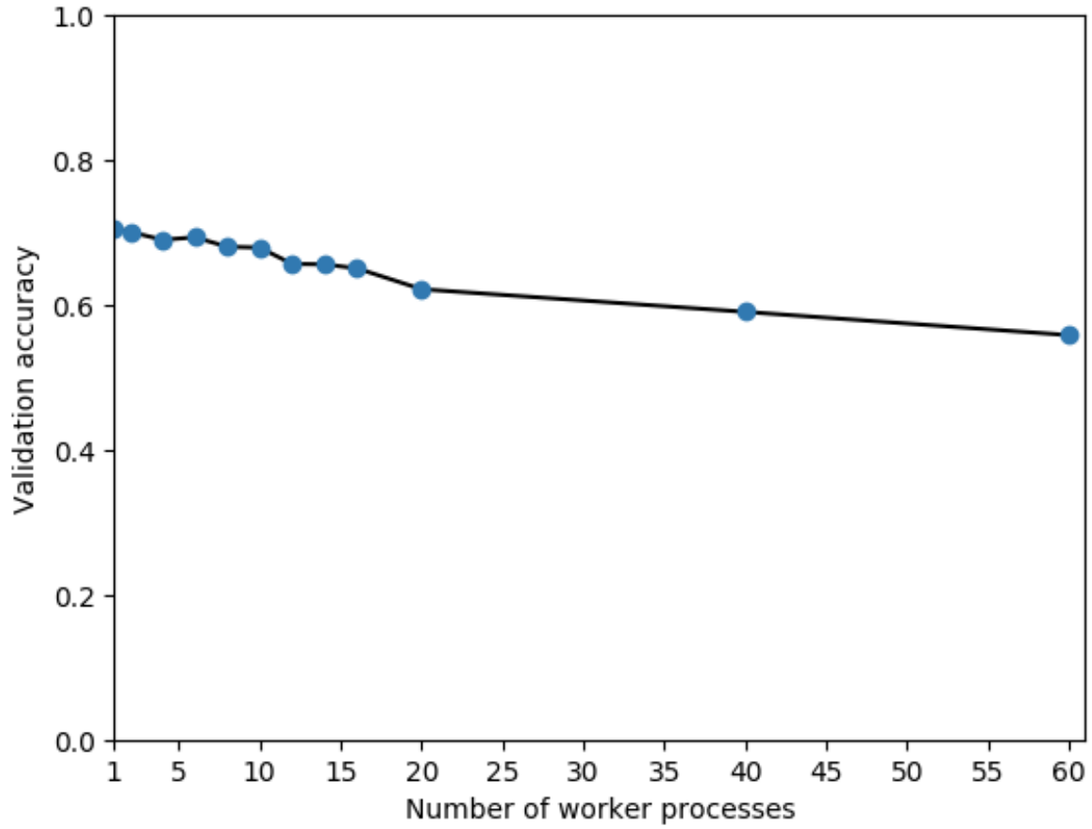
J. Dean, et al. Large scale distributed deep networks. NIPS'12

Downpour SGD



Stale Gradient Problem

- In a distributed setting, nodes often compute gradients using **outdated model parameters**
- SGD updates using old weights are suboptimal (“stale”)
- This issue can be mitigated by suitable choice of SGD momentum [1]

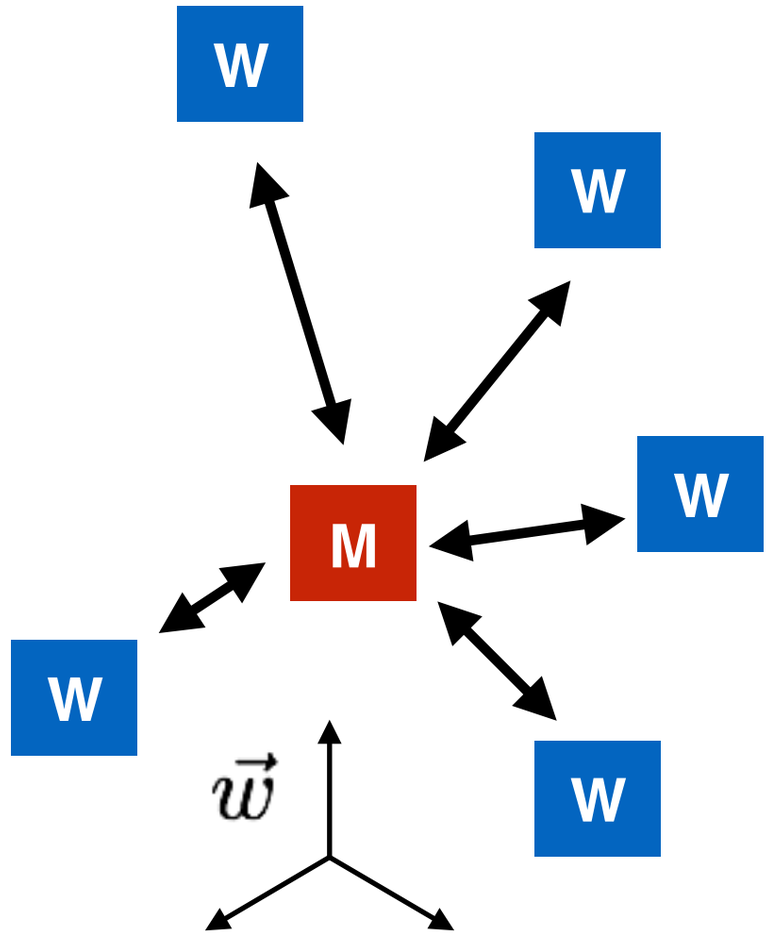


Validation accuracy decreases with larger number of nodes

[1] Omnivore: An Optimizer for Multi-device Deep Learning on CPUs and GPUs https://arxiv.org/pdf/1606.04487.pdf

Elastic Averaging SGD

- A different distributed training algorithm
- Master and Worker model weights are connected via an elastic force
- Workers have individual freedom to explore the parameter space



Zhang et al., Deep learning with elastic averaging SGD. https://arxiv.org/abs/1412.6651

Learning with MPI

- The Message Passing Interface (MPI) is a widespread standard for parallel programming
  - ➡ Used e.g. for job submission at supercomputing sites

- MPI code is portable and agnostic to underlying hardware

- APIs/Libraries for C++, Python (mpi4py), and many others

MPI-Learn Library

- **MPI-Learn** is a python library for MPI-based distributed training of neural networks

https://github.com/duanders/mpi_learn

- Interfaces with Keras

- Goal: provide a lightweight, “plug & play” interface to multi-GPU training

MPI-Learn Library

- Basic workflow:

  A. **Define neural network** as a Keras model

  B. **Define a generator for training data** as a Python generator

  C. **Define training algorithm** and any hyperparameters

  D. **Launch** distributed training!

MPI-Learn Library

- Library features:

  ➡ Downpour (with choice of gradient update algorithm) and Elastic Averaging SGD training

  ➡ Synchronous and asynchronous training

  ➡ Support for Theano and Tensorflow backends to Keras

  ➡ Preliminary support for other Master-Worker hierarchies

https://github.com/duanders/mpi_learn

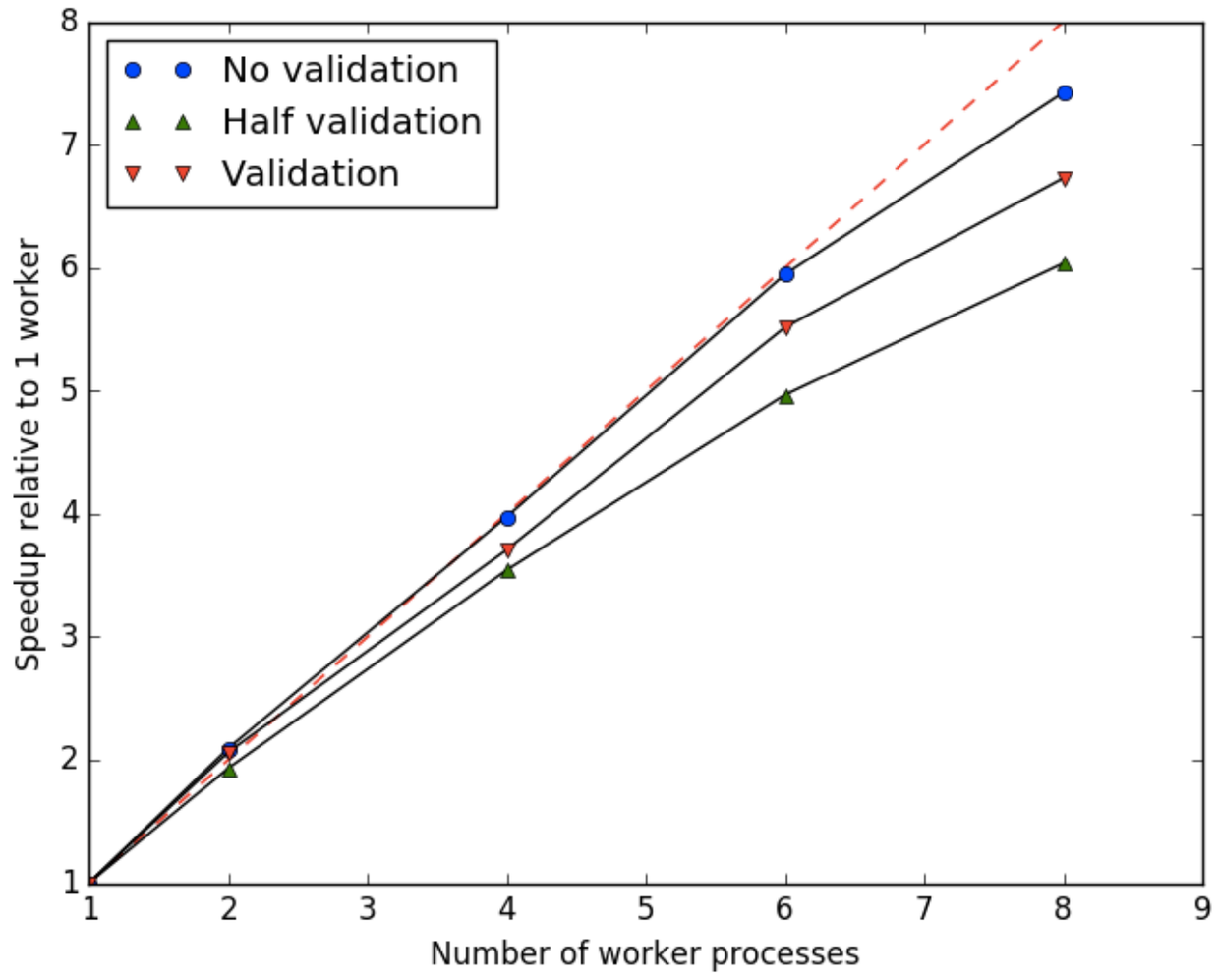
Caltech

Performance Tests

- Trained a benchmark NN on up to 8 GPUs

  ➡ RNN event classification model from J-R Vlimant's talk on Tuesday

- **Training speed-up roughly linear with # of GPUs**



Validation is performed on a single node → constant contribution to training time

Caltech

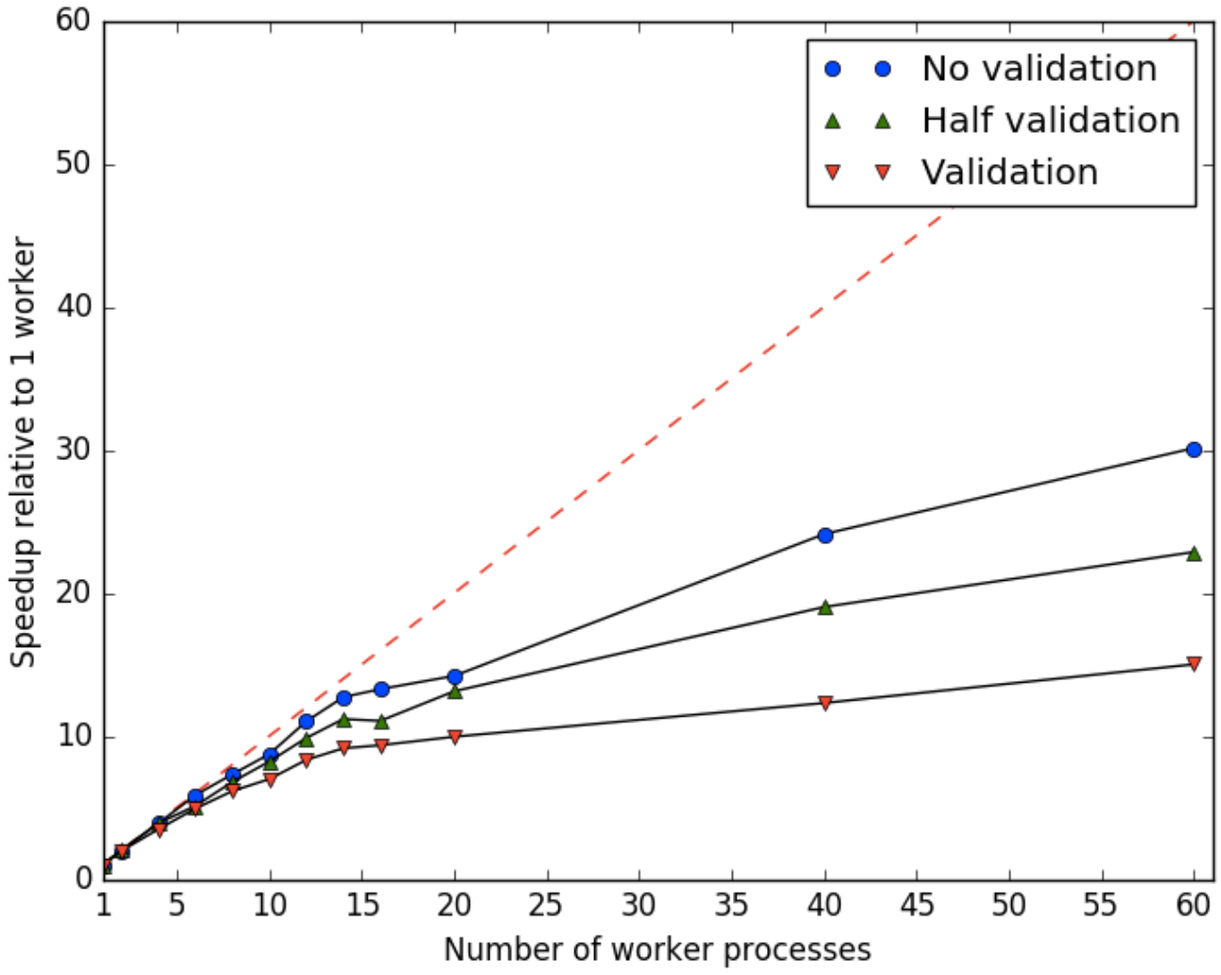
Performance Tests

- Larger-scale test using **ALCF Cooley** cluster

  ➡ Trained with up to 60 GPUs

- Speed-up is linear up to ~15 GPUs

- Speed-up is **30X** when running on 60 GPUs



Conclusion

- Distributed learning becomes increasingly important as DNNs become larger and more widespread

- The MPI-Learn library provides a convenient interface to multi-GPU training of Keras models

- Facilitates quicker prototyping and testing of large deep neural networks