Allreduce for Parallel Learning

John Langford, Microsoft Resarch, NYC

May 8, 2017

Applying for a fellowship in 1997

Interviewer: So, what do you want to do?

John: I'd like to solve AI.

I: How?

J: I want to use parallel learning algorithms to create fantastic learning machines!

Applying for a fellowship in 1997

Interviewer: So, what do you want to do?

John: I'd like to solve AI.

I: How?

J: I want to use parallel learning algorithms to create fantastic learning machines!

I: You fool! The only thing parallel machines are good for is computational windtunnels!

Applying for a fellowship in 1997

Interviewer: So, what do you want to do?

John: I'd like to solve AI.

I: How?

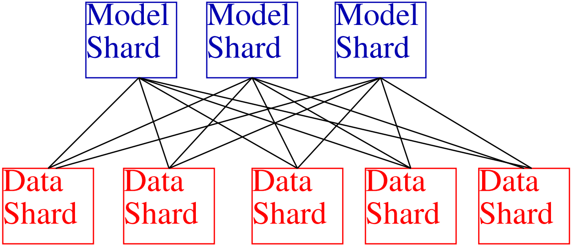
J: I want to use parallel learning algorithms to create fantastic learning machines!

I: You fool! The only thing parallel machines are good for is computational windtunnels!

The worst part: he had a point.

Why is it hard?

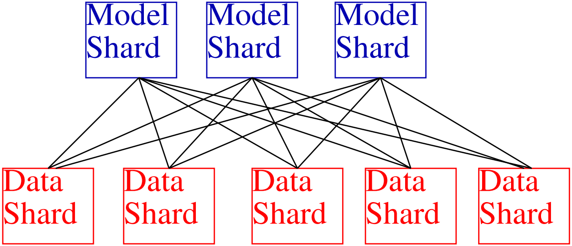
Everyone's first instinct: Try using parameter servers.



1. Hsu, Karampatziakis, Langford, and Smola, "Parallel Online Learning", https://arxiv.org/abs/1103.4204
2. Dean et al, "Large scale Deep Distributed Networks", NIPS 2012.

Why is it hard?

Everyone's first instinct: Try using parameter servers.



1. Hsu, Karampatziakis, Langford, and Smola, "Parallel Online Learning", https://arxiv.org/abs/1103.4204
2. Dean et al, "Large scale Deep Distributed Networks", NIPS 2012.

Big problems in practice:

1. Overwhelmingly inefficient. Best case: marginally faster with x100 electricity.
2. Nondeterministic. Minor bugs incredibly difficult to track.

Terascale Linear Learning ACDL11

Given 2.1 Terafeatures of data, how can you learn a good linear predictor $f_w(x) = \sum_i w_i x_i$?

Terascale Linear Learning ACDL11

Given 2.1 Terafeatures of data, how can you learn a good linear predictor $f_w(x) = \sum_i w_i x_i$?

17B Examples
16M parameters
1K nodes
How long does it take?

Terascale Linear Learning ACDL11

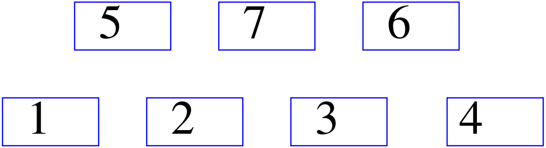
Given 2.1 Terafeatures of data, how can you learn a good linear predictor $f_w(x) = \sum_i w_i x_i$?

17B Examples
16M parameters
1K nodes
How long does it take?

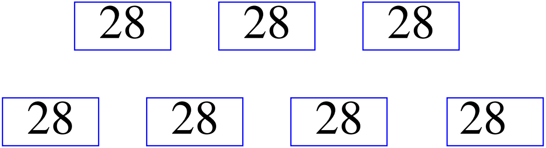
70 minutes = 500M features/second: faster than the IO bandwidth of a single machine⇒ faster than all possible single machine linear learning algorithms.

MPI-style AllReduce

Allreduce initial state

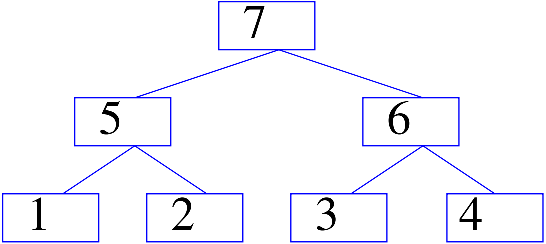


Allreduce final state

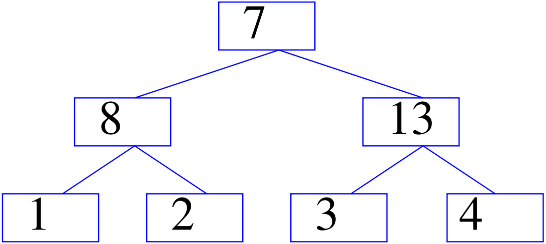


MPI-style AllReduce

Create Binary Tree

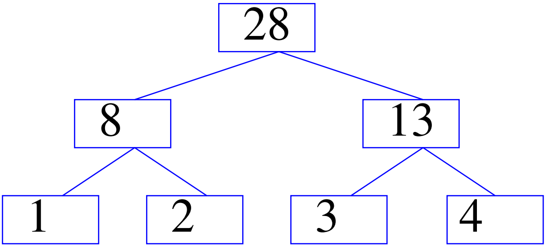


Reducing, step 1

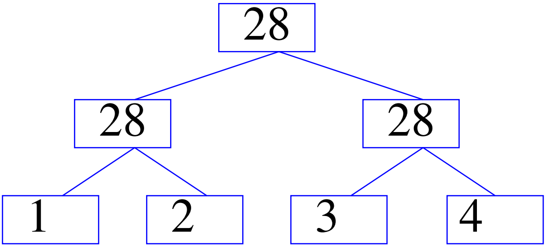


MPI-style AllReduce

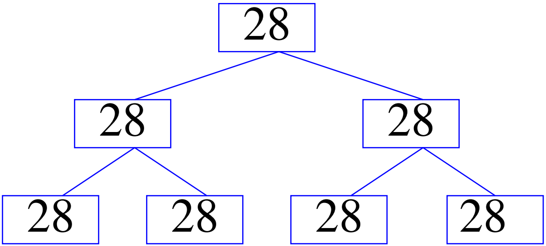
Reducing, step 2



Broadcast, step 1

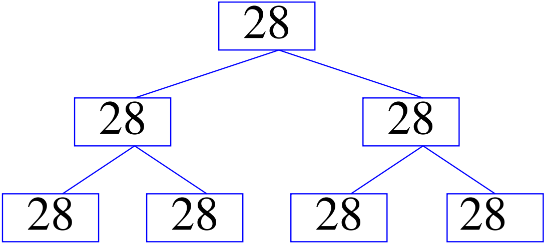


Allreduce final state



AllReduce = Reduce+Broadcast

Allreduce final state



AllReduce = Reduce+Broadcast

Properties:

1. Easily pipelined so no latency concerns.
2. Bandwidth $\leq 6n$.
3. No need to rewrite code!

An Example Algorithm: Weight averaging

$n = \text{AllReduce}(1)$

While (pass number $<$ max)

1. While (examples left)
   1. Do online update.
2. AllReduce(weights)
3. For each weight $w \leftarrow w/n$

An Example Algorithm: Weight averaging

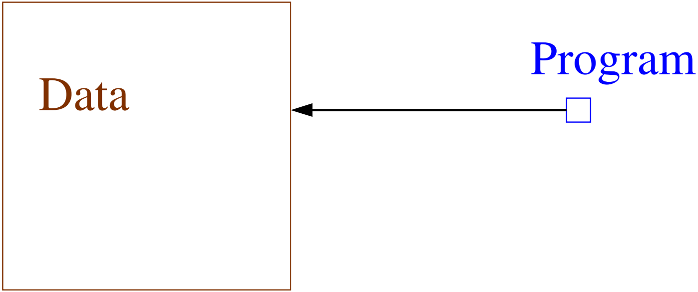
$n =$ AllReduce(1)
While (pass number $<$ max)

1. While (examples left)
   1. Do online update.
2. AllReduce(weights)
3. For each weight $w \leftarrow w/n$

Other algorithms implemented:

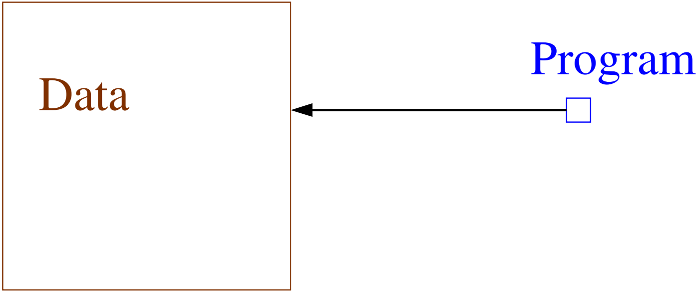
1. Nonuniform averaging for online learning
2. Conjugate Gradient
3. LBFGS

Hadoop AllReduce



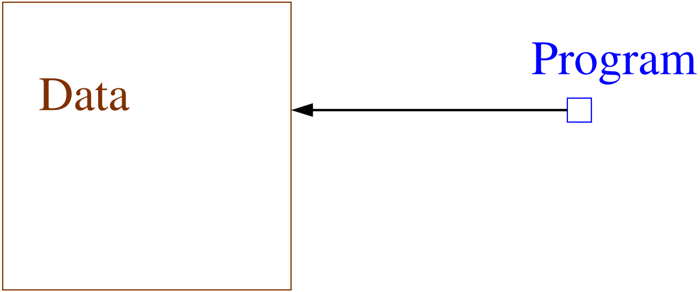
1. "Map" job moves program to data.

Hadoop AllReduce

1. 

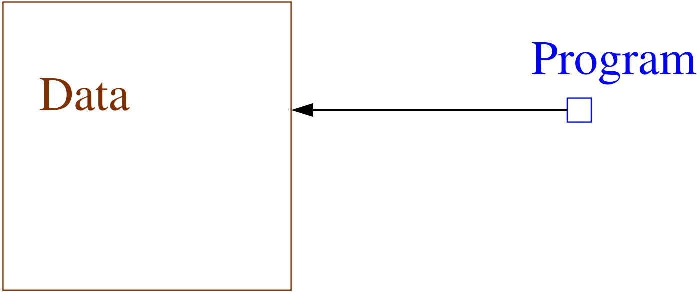
"Map" job moves program to data.
2. Delayed initialization: Most failures are disk failures. First read (and cache) all data, before initializing allreduce. Failures autorestart on different node with identical data.

Hadoop AllReduce

1. 

   "Map" job moves program to data.
2. Delayed initialization: Most failures are disk failures. First read (and cache) all data, before initializing allreduce. Failures autorestart on different node with identical data.
3. Speculative execution: In a busy cluster, one node is often slow. Hadoop can speculatively start additional mappers. We use the first to finish reading all data once.

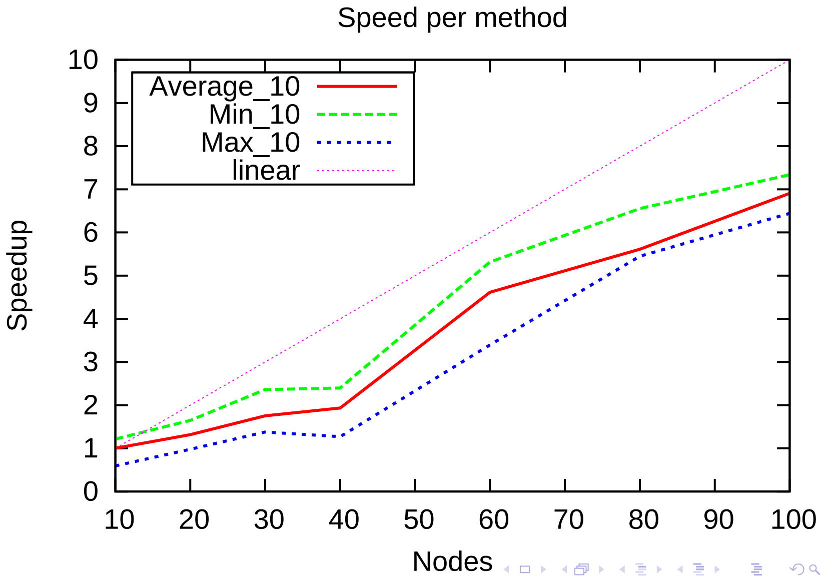
Hadoop AllReduce

1. 

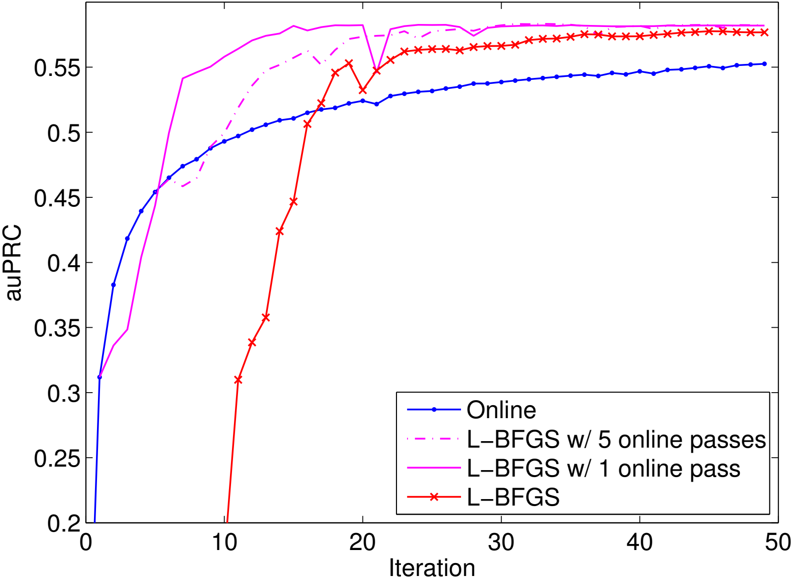
"Map" job moves program to data.
2. Delayed initialization: Most failures are disk failures. First read (and cache) all data, before initializing allreduce. Failures autorestart on different node with identical data.
3. Speculative execution: In a busy cluster, one node is often slow. Hadoop can speculatively start additional mappers. We use the first to finish reading all data once.

The net effect: Reliable execution out to perhaps $10K$ node-hours.

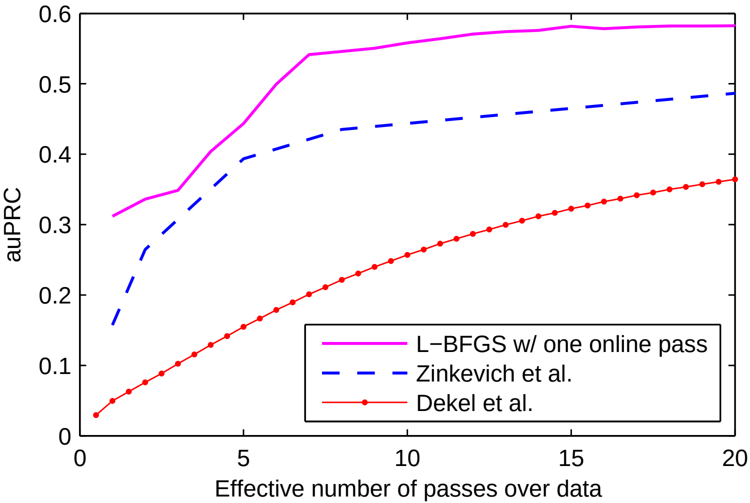
Robustness & Speedup



Splice Site Recognition



Splice Site Recognition



What about parallel deep learning?

Needs to work with a GPU.

GPU Allreduce optimization 1: Minibatch gradient descent

GPUs have much more computation than communication.

GPU Allreduce optimization 1: Minibatch gradient descent

GPUs have much more computation than communication.

Give every GPU n examples and compute average gradient on them. Synchronize Gradient

GPU Allreduce optimization 1: Half-Batch

Do communication asynchronous to computation.

GPU Allreduce optimization 1: Half-Batch

Do communication asynchronous to computation.

1 minibatch communicates while the other computes on older parameters.

GPU Allreduce optimization 1: 1-bit SGD

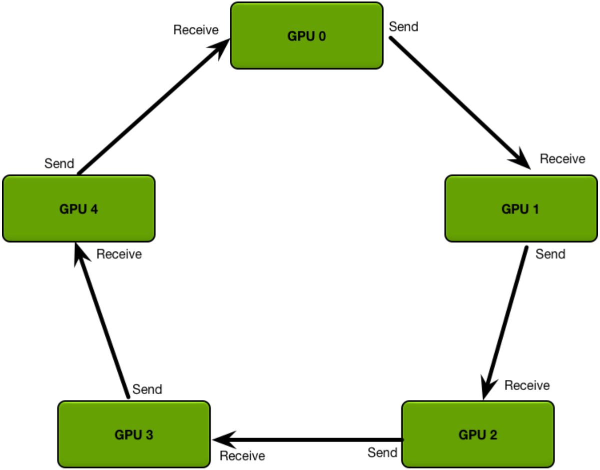
Discretize the gradient to 1 bit before communicating off GPU.

GPU Allreduce optimization 1: 1-bit SGD

Discretize the gradient to 1 bit before communicating off GPU.

Keep and accumulate discretization errors on GPU.

GPU Allreduce Optimization 2: Ring Allreduce



Every node masters a subset and messages travel in a ring.

1. Downside: latency increase?
2. Upside: perfectly efficient synchronization

Bibliography

L-BFGS J. Nocedal, Updating Quasi-Newton Matrices with Limited Storage, Mathematics of Computation 35:773–782, 1980.

grad sum C. Teo, Q. Le, A. Smola, V. Vishwanathan, A Scalable Modular Convex Solver for Regularized Risk Minimization, KDD 2007.

Rings 1 Pitch Patarasuk and Xin Yuan, Bandwidth Optimal all-reduce algorithms for clusters of workstations, JPDC 2009.

avg. G. Mann et al. Efficient large-scale distributed training of conditional maximum entropy models, NIPS 2009.

ov. avg M. Zinkevich, M. Weimar, A. Smola, and L. Li, Parallelized Stochastic Gradient Descent, NIPS 2010.

P. online D. Hsu, N. Karampatziakis, J. Langford, and A. Smola, Parallel Online Learning, in SUML 2010.

D. Mini O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, Optimal Distributed Online Predictions Using Minibatch, JMLR 2012.

Bibliography II

Tera Alekh Agarwal, Olivier Chapelle, Miroslav Dudik, John Langford A Reliable Effective Terascale Linear Learning System, Arxiv 2011/JMLR 2014.

DistBelief Dean et al, Large Scale Distributed Deep Networks, NIPS 2012.

One-bit Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu, 1-Bit Stochastic Gradient Descent and its Application to Data-Parallel Distributed Training of Speech DNNs, Interspeech 2014.

Rings 2 Amodei et al, Deep Speech 2: End-to-End Speech Recognition in English and Mandarin, ICML 2016.