Parallel SGD using MPI

CSE 633 Parallel Algorithms
Instructor: Professor Russ Miller

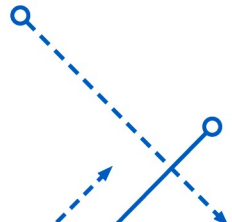
Jiarui Liu

May 6th 2021

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

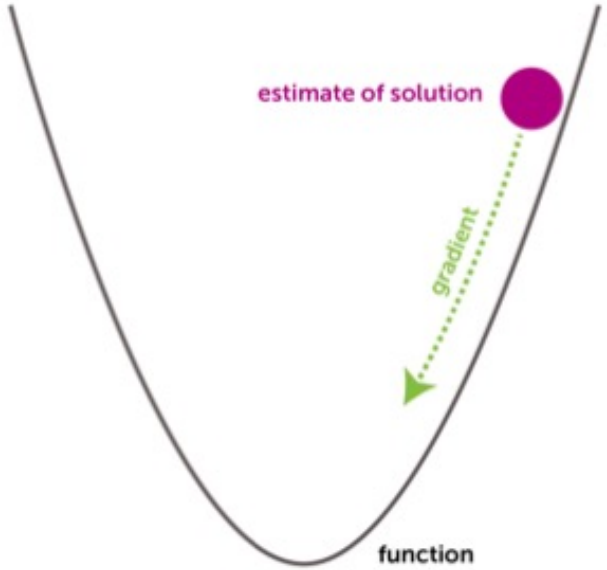
Introduction

- Motivation
- Word2Vec implementation
- Parallel implementation of SGD on neural network
- Experiment Results
- Conclusion



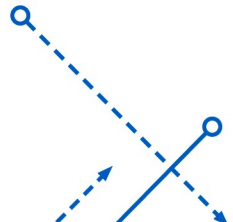
Motivation

- Gradient Descent is a simple and popular algorithm used to minimize loss functions.
  Slow to run on large datasets – even be considered computationally wasteful
- Stochastic-Gradient Descent (SGD) is a variation of Gradient Descent.
  Computes the gradient and updates weight matrix on small batches of training data, rather than the entire training set itself. Still Slow.
- Parallel SGD to improve training speed
  Want a fast and stable solution for parallelizing training over multiple independent nodes to achieve higher speedup.



Word2Vec Implementation

- Implementation of Word2Vec algorithm using the skip-gram architecture.
- Skip-gram: pass in a word and try to predict the words surrounding it in the text.
- Used the text8 dataset – a file of cleaned up Wikipedia articles

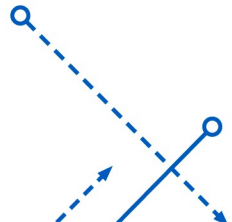


Word2Vec Output

```
Nearest to no: heist, tr, comme, agile, quality, abuse, neapolitan, subrack,
Nearest to six: zero, carmine, on, radii, ln, manon, and, raum,
Nearest to would: often, detours, environments, watershed, connotations, euphoric, redesign, gab,
Nearest to these: from, in, of, the, to, ceramic, and, a,
Nearest to also: the, and, of, in, what, a, fishkeeping, to,
Nearest to have: in, of, a, the, be, three, that, infamous,
Nearest to other: vojvodina, codas, partnering, westland, arithmetics, manga, berio, porky,
Nearest to marriage: eugenicists, rubidium, exponentials, fenwick, specifically, stitch, topos, ephraim,
Nearest to freedom: balances, glise, skirmishes, kfor, masaryk, indoor, pillaged, height,
Nearest to powers: boathouse, singing, franjo, berries, synoptic, games, rumour, stardom,
Nearest to nobel: nationally, aerosols, photographs, sinha, freud, arose, szombathely, chip,
Nearest to running: almenr, illustrators, quintilis, employs, py, stressing, zeus, patrilineal,
```

A Different Neural Network

- Unfortunately, I couldn't implement Word2Vec on CCR due to the reason that I cannot install some of the needed libraries and packages since I don't have the authority in my CCR account.
- So I create another Neural network that performs a classification task based on coordinate inputs.
- The model contains two dense layers, with ReLU and softmax activation function.
- 40,000 samples are used, tagged in 4 classes.



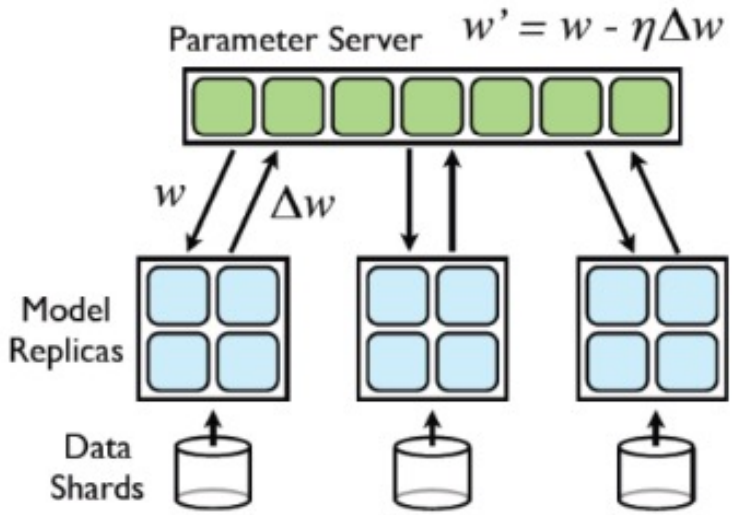
Parallel SGD with MPI

Asynchronous SGD

Asynchronous SGD:

- all workers have a copy of the model, a model replica asks the Parameter Sever for an updated copy of its model parameters
- at every subset:
- workers get parameters from the server
- workers get data from its own data loader, or randomly selected dataset
- workers calculate forward and gradient
- once the calculation is done, gradient is sent to the server,

the server updates the parameters



Divide the data into a number of subsets and run a copy of the model on each of the subsets.

Results

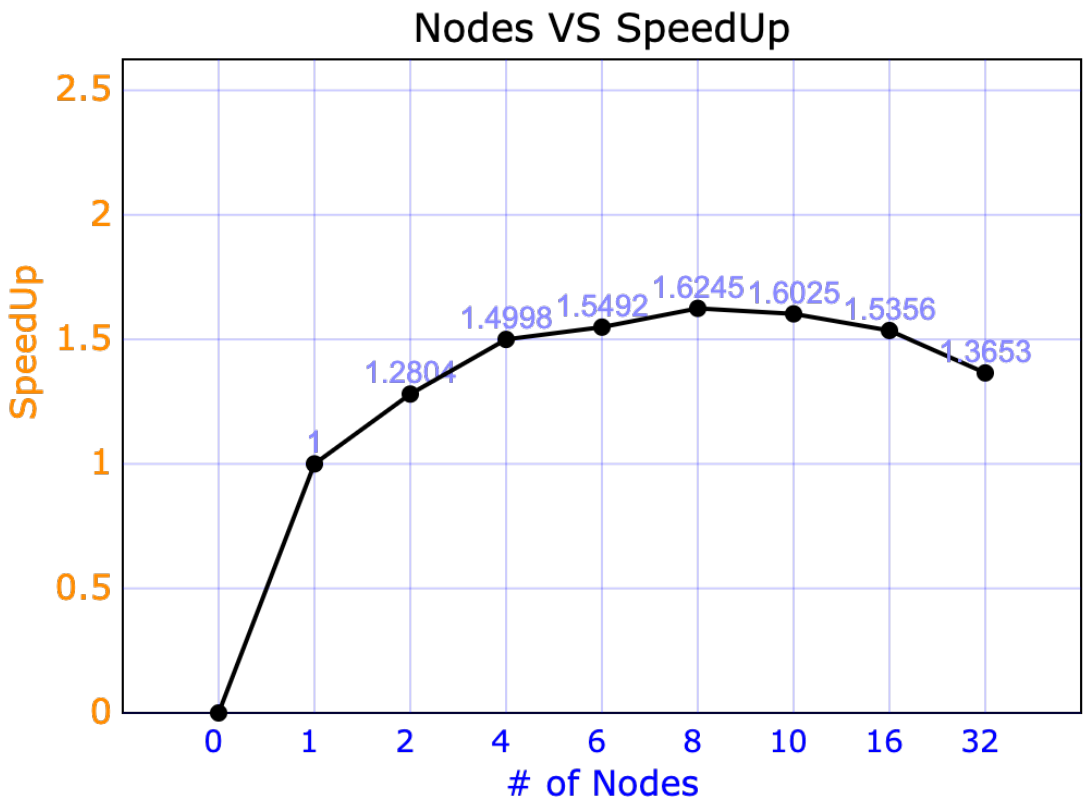
- Ran the experiment in 100 epochs.
- Parallelized SGD on different number of nodes: 1, 2, 4, 6, 8, 10, 16, 32, with 1 processor per node.
- Recorded the time consumption to the training procedure.
- Recorded the training accuracy and loss when using different number of nodes.

Results - SpeedUp

- Number of Nodes V.S. SpeedUp

Number of Nodes	1	2	4	6	8	10	16	32
Execution time in seconds	795.53	621.33	530.41	513.5	489.72	496.44	518.07	582.66
SpeedUp	1	1.2804	1.4998	1.5492	1.6245	1.6025	1.5356	1.3653

Results - SpeedUp



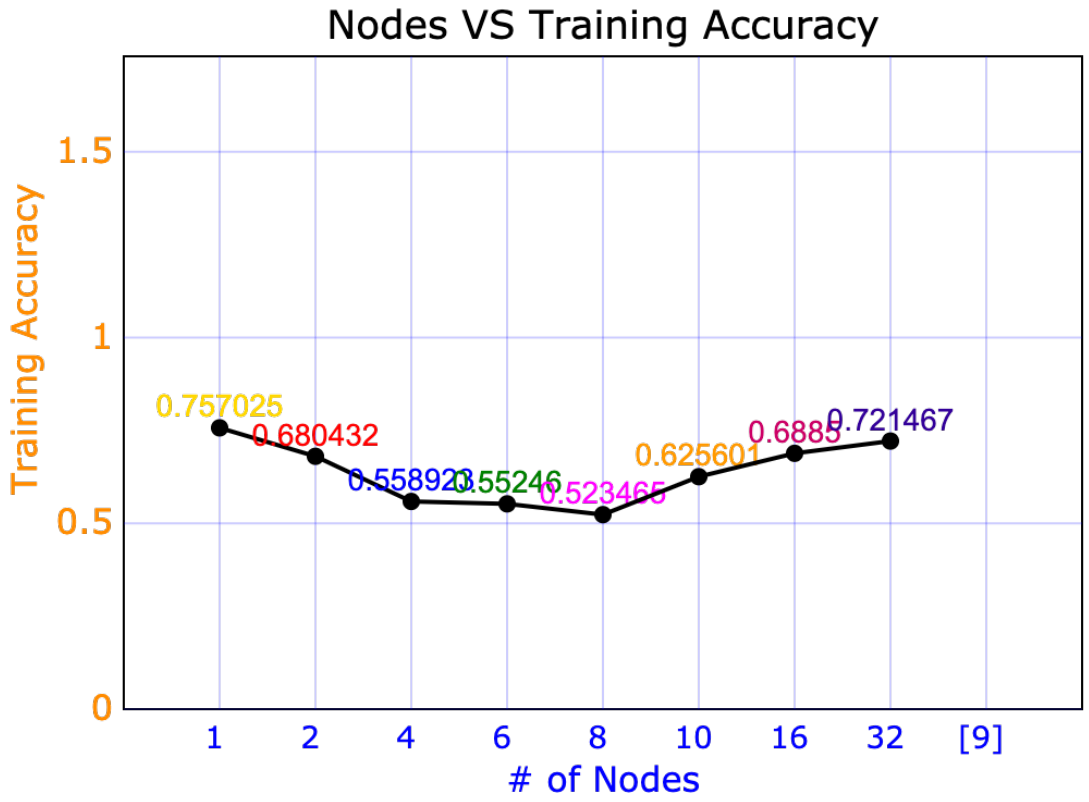
- SpeedUp doesn't looks as good as I expected.
  - Possible reason: Only the SGD procedure is parallelized, a big part of computation is still serial.
- SpeedUp tops at about 8 Nodes partitions, then begins to decrease.
  - Possible reason: Partition becomes too small, communication overhead becomes severe.

Results – Training Accuracy

- Number of Nodes V.S. Training Accuracy

Number of Nodes	1	2	4	6	8	10	16	32
Training Accuracy	0.757025	0.680432	0.558923	0.55246	0.523465	0.625601	0.6885	0.721467
Loss	0.631085	0.764362	0.943527	0.957342	1.03074	0.864385	0.756486	0.692658

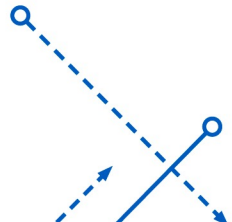
Results – Training Accuracy



- Highest accuracy is achieved with the serial training code.
  - Possible reason: in the serial case, the gradients are calculated on the entire training set, so it may mitigates overfitting and increase performance.

Conclusion

- Parallel SGD can help with Machine learning speedup, but only by itself, the improvement is not quite significant. If work with model and data parallelization, I assume that one can achieve better performance.
- There is tradeoff between computation cost and communication cost. For the cases when there are thousands or more epoch in the training, or the training dataset is not large enough, I am afraid the parallelization won't help much.



Reference

- niu et al. *Hodwild! SGD - A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. 2011*
- *Ji et al. HogBatch SGD - Parallelizing Word2Vec in Shared and Distributed* Memory. 2016