Distributed Machine Learning Systems

Christopher Hopkins

Department of Computer Science

Swansea University

This dissertation is submitted for the degree of

Bachelor

October 23, 2020

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 40,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 100 figures.

Christopher Hopkins
October 2020

Declaration

This is where you write your abstract ...

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Context

Machine Learning algorithms have become ubiquitous in modern life. Powering social media feeds, email spam filters, advertising personalisation and even identifying breast cancer more accurately than doctors. [1] To train these Machine Learning algorithms large datasets are needed. The more data available, the more nuanced and accurate model will be able to be produced

The more nuanced and complex the problem being solved, the more data is necessary. As the scale of problems we are trying to solve dramatically increase, the scale of datasets are too. Services like the Internet Archive as of 2020 contain over 70PB in its database. Labeled datasets such as AViD have 467k videos and 887 action classes, which is in the order of terabytes. [2] Whilst the data grows the as does the Machine Learning models in order to obtain ever more accurate results. Since 2008 Google has been processing more 20PB of data a day using their Machine Learning MapReduce algorithm. [3].More recently the cutting edge GTP-3 Natural Language Processing model contains 175 Billion parameters. [4] And efforts are being made to create models with trillions of parameters. [5]

Deriving meaning from these vast quantities of data to obtaining nuanced insights from them is a difficult task. Not only because deeper insights into data require larger Machine Learning models. But because more data is needed to populate the parameters of these models. Both of these factors contribute to the need to distribute the computation of the model across multiple nodes otherwise known as Distributed Machine Learning. Distributed Machine Learning is often a pre-requisite for training models, now datasets and models are becoming so large. [6] Scalability is another challenge a Distributed Machine Learning system must face. As more machines are added to a system the communication overhead increases. Communication bandwidth is a limited resource which can produce bottlenecks when the network becomes saturated. With more independent systems working in conjunction and training that costs extensive time and expenses, resiliency has also become a important factor in Distributed Machine Learning. When a Machine fails to work a Distributed Machine Learning System must have contingency plans. These plans can ensure the minimum amount of model data is lost, and that the machine learning can continue swiftly after a major error.

The popular current solution is to have multiple machines compute the model together, communicating the improvements that they've made to each other. The model operates on a worker and parameter server paradigm. The server holds the global model including all the parameters. Workers then use the model in conjunction with the training data split between them to update the global model in the parameter server. [6]

However this model is not perfect and has 3 considerable drawbacks:

- Workers either spend too much time being idle or perform redundant computation, even when the system is under full load. [3]
- Every worker must communicate with a single parameter server, this limits scalability as eventually the network bandwidth becomes saturated severely impacting performance. [6]
- Many parameter server models require the whole model to be replicated within each node. [7] This means that very large models simply cannot run in machines with low system memory.

1.2 Aims

My solution to address these issues raised above and a number of others is to introduce a new model for Distributed Machine Learning: *RingTMP*. RingTMP (Ring Topological Model Parallel) is a Ring Topological Model Parallel Distributed Machine Learning framework focusing on optimising Distributed Stochastic Gradient Descent. This is a novel design drawing in inspiration much research but particularly from the STRADS and Distbelief machine learning frameworks, [8, 9] in addition to identifying issues with existing systems and addressing them. These are the benefits my system will have over existing parameter server architecture:

- Workers will have less idle time, because the work will be distributed more proportionally between nodes. And computing resources will be used more efficiently.
- There will less communication across nodes as the parameter server so not need to send their local weights to the global parameter server each iteration.

- There will also be a potential for larger communication bandwidth, as with RingTMP ring topology only adjacent nodes need to communicate. While with a parameter Server model every worker needs to communicate with the Parameter server at the end of each iteration, which can lead to bottlenecking.

- The nodes require no scheduling as in some model parallel systems. Instead scheduling is managed in a decentralised manner via communication with adjacent nodes. This also makes the system more resilient, in the case of the scheduler crashing.

My aims more specifically for this project are to:

- Create a prototype RingTMP framework and run it multiple machines simultaneously.

- Create a parameter server model framework using the same tools and run that over multiple machines simultaneously.

- Show that RingTMP reduces the time workers are idle in comparison to the parameter server model.

- Exhibit that per iteration RingTMP makes more progress than the parameter server.

- Display that RingTMP is more scalable than a generic parameter server.

- Show that this solution can be as resilient as a parameter server model distributed neural network.

- Demonstrate that RingTMP can hold larger models in comparison to a standard parameter server, where each worker hold the whole model.

2 Background

One of the first pieces of research into Distributed Machine learning was 'Distributed Inference for Latent Dirichlet Allocation' in 2008 [10] One of the first instances of Distributed Machine Learning was used to categorise New York Times articles using Latent Dirichlet Allocation (LDA), which identifies the affiliations words have to certain topics. While the paper focused on parallelising the algorithm and running them over multiple artificially isolated cores the results showed that distributed machine learning could have scalability and didn't impact the rate of convergence of the model. This was followed by a paper by Jia et al. [11] which produced much faster results than its predecessors by using memcache layer in every machine, every machine would message every other machine with updates of its local parameters to create an approximate global state, it was mentioned in passing that arranging the nodes in a star topology and caching the values that passed through it could make the system more scalable. Then in 2014 came the parameter server as it is known today [6]. They improved upon the shared memcache layer by turning it into its own standalone server.

3 Project Plan and Time Management

example of reference is [12]

References

[1] S. M. McKinney, M. Sieniek, V. Godbole, J. Godwin, N. Antropova, H. Ashrafian, T. Back, M. Chesus, G. S. Corrado, A. Darzi, M. Etemadi, F. Garcia-Vicente, F. J. Gilbert, M. Halling-Brown, D. Hassabis, S. Jansen, A. Karthikesalingam, C. J. Kelly, D. King, J. R. Ledsam, D. Melnick, H. Mostofi, L. Peng, J. J. Reicher, B. Romera-Paredes, R. Sidebottom, M. Suleyman, D. Tse, K. C. Young, J. De Fauw, and S. Shetty, "International evaluation of an ai system for breast cancer screening," *Nature*, vol. 577, pp. 89–94, Jan 2020.

[2] A. Piergiovanni and M. S. Ryoo, "Avid dataset: Anonymized videos from diverse countries," 2020.

[3] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, p. 107–113, Jan. 2008.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[5] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "Zero: Memory optimizations toward training trillion parameter models," 2020.

[6] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, (Broomfield, CO), pp. 583–598, USENIX Association, Oct. 2014.

[7] Z. Jia, M. Zaharia, and A. Aiken, "Beyond data and model parallelism for deep neural networks," 2018.

[8] J. K. Kim, Q. Ho, S. Lee, X. Zheng, W. Dai, G. A. Gibson, and E. P. Xing, "Strads: A distributed framework for scheduled model parallel machine learning," in *Proceedings of the Eleventh European Conference on Computer Systems*, EuroSys '16, (New York, NY, USA), Association for Computing Machinery, 2016.

[9] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. aurelio Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1223–1231, Curran Associates, Inc., 2012.

[10] D. Newman, P. Smyth, M. Welling, and A. U. Asuncion, "Distributed inference for latent dirichlet allocation," in *Advances in neural information processing systems*, pp. 1081–1088, 2008.

[11] A. Smola and S. Narayanamurthy, "An architecture for parallel topic models," *Proc. VLDB Endow.*, vol. 3, p. 703–710, Sept. 2010.

[12] S. K. Patel, V. Rathod, and S. Parikh, "Joomla, drupal and wordpress-a statistical comparison of open source cms," in *Trendz in Information Sciences and Computing (TISC), 2011 3rd International Conference on*, pp. 182–187, IEEE, 2011.

Appendices

A one

B two