Distributed Machine Learning Systems

Prifysgol Abertawe
Swansea University

Christopher Hopkins

Department of Computer Science
Swansea University

This dissertation is submitted for the degree of
Bachelor

October 22, 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 and earlier than doctors. [1] To train these Machine Learning algorithms large datasets are needed. 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 becoming truly gargantuan. Since 2008 Google has been processing more 20PB of data a day using their MapReduce algorithm. [2] While services like the Internet Archive as of 2020 contain over 70PB in its database. We now have labeled datasets such as AViD have 467k videos and 887 action classes, which is in the order of terabytes. [3] Whilst the data grows the as does the Machine Learning models in order to to obtain ever more accurate results. 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 a 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. This is a limited resource which can produce bottlenecks when too many machines are added. With more independent systems working in conjunction and training that costs extensive time and expenses, resiliency has also become and important factor in Distributed Machine Learning. When a Machine fails to work a Distributed Machine Learning System must have contingency plans in place to make sure the minimum amount of parameter data is lost.

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 goes from operating on a single machine possessing all the data and needing to do all the computation, to a worker and parameter server paradigm. In which the parameter server contains the

model and the workers perform operations on it using test data segmented between them. [6]

There are two main variations with respect to the operation of the workers in parameter server model: 1) The parameter server has to wait for the last worker to be finished before it can calculate the new global parameters. much like the MapReduce algorithm. [2] 2) The workers operate asynchronously constantly updating the parameter server, the parameter server calculating new global parameters periodically. [7] Whilst this method is the most common method of machine learning with many benefits, there are 3 key drawbacks:

- The model sacrifices efficiency in either time or computation. Either it must wait for all workers to be done each round, or redundant computations must be made. [8]
- when the parameter server is calculating the new global parameters the workers are idle or otherwise computing on stale data. [9]
- Each time the parameter server calculates a new global parameters, these parameters must be broadcast to each worker simultaneously, consuming vast network bandwidth. [6]

As has already been addressed frequently models can get so large that they can no longer feasibly be held within one worker. Therefore there are also variations with respect to how much of the model each worker operates on: 1) The model is segmented and split between worker. This is known as *Model Parallelism* 2) The data is split between the workers which have their own full local models, but are synced with each other at periodic intervals. This is known as *Data Parallelism* [10] Though model parallelism shows promise it has its own set of drawbacks:

- Often in model parallelism nodes do not communicate with each other, this makes it performing algorithms such as Stochastic Gradient Decent difficult as clusters of nodes are isolated.
- some model parameters take more algorithmic iterations to converge than others, so that they converges at the same rate, this means that some nodes may be idle, not spreading the load equally or efficiently. [11]
- Because some parameters converge at different rates, a scheduler is needed. However this in turn require more computational overhead and communication and reduces iteration throughput. [12]

1.2 Aims

My solution to address these issues raised above 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 from the STRADS and Distbelief machine learning frameworks, [11, 12] in addition to identifying issues with existing systems and addressing them. These are the benefits my system will have over both existing data parallel systems and model parallel systems:

- 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 on comparison to a standard parameter server, where each worker hold the whole model.

2 Related Works

Hello there

3 Project Plan and Time Management

example of reference is [13]

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

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

[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] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, "More effective distributed ml via a stale synchronous parallel parameter server," in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 1223–1231, Curran Associates, Inc., 2013.

[8] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project adam: Building an efficient and scalable deep learning training system," in *11th USENIX Symposium on*

Operating Systems Design and Implementation (OSDI 14), (Broomfield, CO), pp. 571–582, USENIX Association, Oct. 2014.

[9] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Comput. Surv.*, vol. 53, Mar. 2020.

[10] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu, "Petuum: A new platform for distributed machine learning on big data," *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 49–67, 2015.

[11] 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.

[12] 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.

[13] 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