**Md Majid Jahangir | SBU ID - 112077145**
**AMS 560 - Big Data Systems, Algorithms and Networks**

## Assignment 6 (Paper Reading)
## Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds

**Motivation :** Nowadays, data generated by an organization or a system is vast in units of petabytes and is also geographically generated and stored across the globe. Applying machine learning (ML) algorithms on such a geographically distributed vast data is a huge challenge as it is very infeasible to move this globally generated data from across the globe to a centralised data center in order to apply ML algorithms on them. The infeasibility arises from the fact that moving such vast data over wide-area networks (WANs) is extremely slow and is also prone to constraints of privacy and data protection laws. This motivated the need for an ML framework system that is geographically distributed and spanning multiple data centers. Therefore, this paper [1] proposed such a geo-distributed ML system called Gaia that tries to decouple communication within a data center from communication between data centers.

**Prior Solutions:** The paper "Towards geo-distributed machine learning" [2] discussed the problem of running ML algorithm on a geographically distributed data and devised a communication-efficient algorithm for logistic regression models; however the solution would force ML researcher to change the underlying algorithm which is quite challenging in terms of algorithm specific. On the contrary, Gaia leverages design of communication-efficient mechanism at system level without any need of modifications to ML algorithms.
Other prior distributed ML systems or frameworks such as Apache Mahout [3], Apache Spark MLlib [4], Tensorflow [5], etc also proposes several synchronization models and optimizations enabling large-scale ML applications on a large number of machines; however their work assumes network communication happening within a data center and does not consider scarce WAN bandwidth and hence suffer from performance degradation when they are deployed across multiple data centers on WANs.
Another solution which uses the same non-uniform convergence mechanism in ML algorithms to improve convergence speed and minimizing communication of updated parameters over network as same as Gaia is another prior work [6]. This prior work uses various filters as same as Gaia to minimise communication between workers and parameter servers. However, this also does not differentiate communication over LANs over WANs and also does not propose a synchronization model across multiple data centers.
Last but not the least, another famous similar solution released almost at the same time as that of Gaia is Google's Federated Learning and is focused on the same issue and tries to reduce communication over WAN. It is also based on parameter server model as is Gaia and updates model's parameter's value in a stale manner. However, it differs from Gaia in the sense that Gaia is a peer to peer solution where each datacenter has a parameter server, whereas Federated Learning has a single master parameter server across data centers.

**Key Ideas of Proposed Solution:** The key ideas of the proposed solution is centric towards the goal of the paper i.e. to minimize the communication over WANs and that the solution is general and flexible enough to be applicable to a wide variety of ML algorithms without any modification to the underlying algorithms. The proposed solution Gaia is built on the designs of parameter-server architecture that provides ML worker machines with a distributed global shared memory abstraction for ML model parameters that are collectively trained until convergence to fit the input data. The key idea of the solution is to maintain an approximately-correct copy of the global ML model inside of each data center, and to dynamically eliminate any unnecessary communication between data centers. This is enabled by decoupling the synchronization model (communication/consistency) inside of a data center from the synchronization model between multiple data centers. Such a decoupling allows Gaia to run the conventional synchronization model ensuring maximised utilization of the LAN bandwidth inside of a single data center; and also to employ a new synchronization model called Approximate Synchronous Parallel (ASP) for across data centers, to make efficient use of scarce WAN bandwidth. The solution guarantees ML algorithm convergence by ensuring that

each of the ML model copy across different data centers is approximately correct based on a model defined by ASP. Detailed explanation on these key ideas has been provided under the key questions section.

**Performance:** The paper uses 3 different platforms for performance evaluation namely Amazon EC2 - 22 machines spread across 11 EC2 regions, Emulation EC2 - since experimenting on Amazon EC2 was expensive, EC2 was emulated in a local cluster with 22 machines with subdued LAN and CPU speed matching with Amazon EC2 speed metrics; and Emulation Full Speed - emulating Amazon EC2 with 22 machines in their local cluster at their full LAN and CPU speed which is more than Amazon EC2.
Also, Gaia was evaluated on 3 popular ML applications namely, Matrix Factorization - technique used in recommender systems; Topic Modeling - unsupervised method for discovering hidden semantic topics in unstructured documents; and Image Classification - classifying images into categories.
The 2 performance metrics used are execution time until algorithm convergence - the convergence criteria is that the value of objective function of the algorithm changes by less than 2% over 10 iterations; and the second metric is the cost of algorithm convergence cost includes both cost model based on Amazon EC2 including server time and cost of data transfer on WANs. These metrics are evaluated and compared against Baseline, 2 state of the art parameter-server systems known as IteStore and GeePS across multiple data centers; and LAN, IterStore and GeePS within a single data center.
It was observed that their Emulation EC2 platform matched the execution time of Amazon EC2 deployment and on Amazon EC2 Gaia provides a speedup of 2 times over Baseline and Gaia's performance is very similar to performance of LAN for Matrix Factorization (MF) ML application. For Topic Modelling (TM) application, Gaia provides a speedup of 2 time and is within 1.25 times of ideal speed of LAN; whereas for Image Classification (IC) Gaia provides a speedup of 5.6 times over Baseline and within 1.32 times LAN speed. For Emulation Full Speed platform Gaia improves performance by 3.8 times for MF, 3.7 times for ™ and 6 times for IC over Baseline.
Cost metric was divided into 3 parts - cost of machine waiting on network, cost of machine time spent on computation and cost of data transfer across different data centers. It was observed Gaia was cheaper than the cost of each respective Baseline by 2.6 to 59 times and majority of cost saving came from reduction of data transfer on WANs and reduction of machine waiting for networks.

**What is difference between ASP and the other ML synchronization models? [Key Question]:** One of the ML synchronization model is Stale Synchronous Parallel (SSP), which bounds how stale or old a parameter can be, whereas ASP bounds how inaccurate a parameter can be in the model, in comparison to the most up-to-date value. Due to this ASP has higher flexibility in performing updates as the server can delay synchronization definitely as long as the aggregated update is insignificant. SSP still send all updates unlike Gaia
Another model called Bulk Synchronous Parallel (BSP) synchronizes all updates after each worker goes through its shard of data and hence all worker needs to see the most up to date model before proceeding to next iteration. Whereas in ASP, all workers need not see the most up to date and it ensures that global model copy in each data center is approximately correct.
Another model called Total Asynchronous Parallel (TAP) removes the synchronization between workers completely and all the workers will run based on results of best effort communication i.e. all updates are received or sent. On the contrary, ASP selectively sends significant updates only based on some significance threshold and not all updates are sent and the aggregated updates are sent only when they become significant. Also, TAP does not guarantee convergence whereas ASP guarantees convergence.

**How does ASP work? [Key Question]:** Approximate Synchronous Parallel is a synchronization model used by Gaia for communication across multiple data centers. It is based on a key observation of ML researchers that vast majority of updates to the global ML model

parameters from each worker machine are insignificant. With ASP, these insignificant updates to the same parameter within a data center are aggregated until the aggregated updates are significant enough to be communicated to other data centers. ASP allows the ML researcher to specify the function and and threshold to determine the significance of updates for each ML algorithm and also provides default configuration for unmodified ML algorithms. ASP ensures all the significant updates are synchronized across all model copies and dynamically adapts communication to the available WAN bandwidth between multiple data centers using a special selective barrier and mirror clock to ensure convergence of algorithm even during a sudden fall in available WAN bandwidth.

The 3 components of ASP are -

1. **Significance filter** - User has to provide 2 inputs to ASP - a significance function and a significance threshold. A parameter server will aggregate updates from local worker machines and will share the aggregated updates to other data centers whenever the aggregated data becomes significant. In order to facilitate convergence to optimal point, ASP will automatically reduce the significance threshold over the period of time.
2. **ASP selective barrier** - Whenever a parameter server receives significant updates at higher rate than what WAN bandwidth can support, then instead of sending updates, it will first send a short control message to other data centers. The receiver of this selective barrier message will block its local workers from reading the parameters until it receives significant updates from sender of barrier.
3. **Mirror Clock -** This component provides the final safety net implementing SSP across multiple data centers. Whenever a parameter server receives all the updates from local machine workers at the end of a clock, it will report its clock to the servers that are in charge of same parameters, in multiple data centers. Whenever a server detects its clock is ahead of the slowest server, that server will block allowing the slowest mirror server to catch up.

**How are significant and insignificant updates defined in Gaia? [Key Question]:** The significance filter component of ASP is used to define whether an update will be significant or not. It is specified by users by providing a significance function and a significance threshold. The significance function returns the significance of each update and an update is designed as significant if only its significance is larger than threshold. Also to ensure that algorithm converge to optimal point, ASP automatically reduces the significance threshold over time (for example if the original threshold is ‘v’ then the threshold at iteration numbered ‘t’ of ML algorithm will be ‘v/sq.root(t)). Whenever a local server receives a parameter update from a worker, it invokes the significance filter to determine the significance of the update. If the update is significant then the filter sends a MIRROR UPDATE request to mirror client and resets the accumulated update for this parameter

**Strength :**

1. It is one of the first geo-distributed machine learning framework or system which allows machine learning algorithms to be applied on geographically distributed data across multiple data centers. This solves many of the challenges that are observed in this era with petabytes of data stored across the globe in multiple data center such as communicating over WAN with low bandwidth and high cost of WAN bandwidth.
2. Unlike other related solutions that existed prior to this paper, Gaia does not require ML researchers to modify the underlying ML algorithms which can be challenging and algorithm specific. Gaia is more of design of communication efficient mechanism at the system level.
3. Unlike other conventional large scale distributed ML framework, Gaia decouples communication model over LAN and WAN and does not assume that communication always happen within a data center. As a result of this, it has significant higher

performance metrics than its counterpart which just assumes communication within a data center and it can differentiate communication over LAN and WAN.
4. Communication overhead between multiple data centers is proportional to the number of data centers and the farther the data centers are, the expensive it gets to transfer parameter updates, however Gaia with usage of overlay networks and hubs mitigates this issue by aggregating the updates at hubs and thus saving communication on WANs.
5. Communication models employed by Gaia makes it very cost effective.
6. Usage of ASP synchronization model helps Gaia in achieving optimum convergence faster than its counterpart solution in a distributed data environment.

**Weakness :**

1. The paper focuses on the challenge of having a scarce WAN bandwidth but does not focus on WAN latency. Since, one of the performance metric of the paper was to investigate the time taken to converge for a ML algorithm, the paper does not study the effect of WAN latency which may degrade the performance results. Bandwidth and latency are two key metrics for analysing the performance of any system and not considering the effect of latency is a key weakness of this paper.
2. In a centralised data environment, where all data from multiple data centers are transferred to a centralised server before applying ML algorithms, such an environment outperforms Gaia in applications such as Topic Modelling and Image Classification and this happens whenever there is a performance gap between Gaia and a LAN platform setup.
3. Gaia is not designed for heterogeneous WAN based infrastructure which is quite prominent in this period. Nowadays, in a single data center or across multiple data centers we come across heterogeneous environment and this is huge limitation for Gaia to adapt to such heterogeneous environments.

**Follow Up Works :**

1. Decentralized Distributed Deep Learning in Heterogeneous WAN Environments [9] - In this cloud computing era, there is a growing interest to analyze data and provide insightful user experiences through cloud-based deep learning (DL) products such as Amazon Sagemaker, etc. However, much of this learning currently takes place on centralized high-performance cloud infrastructures such as GPU clusters. However, due to concerns of efficiency, cost, and privacy, such DL may be better carried out across a combination of edge servers, private clouds, and public clouds. Therefore, distributed deep learning in WANs is used for training models using geo-distributed machines with heterogeneous computing and network capabilities. Distributed deep learning training in heterogeneous WANs can be challenging because of scarce WAN bandwidth and also, in a WAN-based heterogeneous network comprising cloud servers and edges servers, with large gaps in their computing resources and network capacity, the training can slow down significantly or even fail to converge. The paper proposes a decentralized distributed deep learning framework for such heterogeneous WAN-based infrastructures. The framework dynamically and automatically adjusts 1) the frequency of parameter sharing, 2) the size of parameters shared depending on individual network bandwidth and data processing power, and 3) introduces a new scaling factor to control the degree of contribution to parameter updates by considering the amount of data trained during unit time in each device. This paper extends Gaia in the sense that Gaia was never designed for heterogeneous WAN-based infrastructures.

**References :**

1. Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds - Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R. Ganger, and Phillip B.

Gibbons, Carnegie Mellon University; Onur Mutlu, ETH Zurich and Carnegie Mellon University
2. Towards geo-distributed machine learning: I. Cano, M. Weimer, D. Mahajan, C. Curino, and G. M. Fumarola
3. “Apache Mahout.” http://mahout.apache.org/
4. “Apache Spark MLlib.” http://spark.apache.org/ mllib/
5. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Largescale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. http://tensorflow.org/
6. M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B. Su, “Scaling distributed machine learning with the parameter server,” in OSDI, 2014
7. http://muratbuffalo.blogspot.com/2017/09/paper-review-gaia-geo-distributed.html
8. https://slideplayer.com/slide/12567242/
9. Decentralized Distributed Deep Learning in Heterogeneous WAN Environments: Rankyung Hong, Abhishek Chandra - https://dl.acm.org/citation.cfm?id=3275447