Towards Geo-Distributed Training of ML Models in a Multi-Cloud Environment

Chetan Phalak
TCS Research, India
chetan1.phalak@tcs.com

Dheeraj Chahal
TCS Research, India
d.chahal@tcs.com

Manju Ramesh
TCS Research, India
manju.ramesh1@tcs.com

Rekha Singhal
TCS Research, India
rekha.singhal@tcs.com

ABSTRACT

Geo-distributed (GD) training is a machine-learning technique that uses geographically distributed data for model training. Like Federated Learning, geo-distributed machine learning can provide data privacy and also benefit from the cloud infrastructure provided by many vendors in multiple geographies. However, GD training suffers from multiple challenges such as performance degradation due to cross-geography low network bandwidth and high cost of deployment. Additionally, all major cloud vendors such as Amazon AWS, Microsoft Azure, and Google Cloud Platform provide services in several geographies. Hence, finding a high-performance as well as cost-effective cloud service provider and service for GD training is a challenge. In this paper, we present our evaluation of the performance and cost associated with training models in multi-cloud and multi-geography. We evaluate multiple deployment architectures using computing and storage services from multiple cloud vendors. The use of serverless instances in conjunction with virtual machines for model training is evaluated in this study. Additionally, we build and evaluate cost models for estimating the cost of distributed training of models in a multi-cloud environment. Our study shows that the judicious selection of cloud services and architecture might result in cost and performance gains.

CCS CONCEPTS

• **Computing methodologies** → **Model verification and validation**; • **Computer systems organization** → **Cloud computing**.

KEYWORDS

Geo-distributed training, multi-cloud, cost model

ACM Reference Format:
Chetan Phalak, Dheeraj Chahal, Manju Ramesh, and Rekha Singhal. 2024. Towards Geo-Distributed Training of ML Models in a Multi-Cloud Environment. In *Companion of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24 Companion), May 7–11, 2024, London, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3629527.3651422

ICPE '24 Companion, May 7–11, 2024, London, United Kingdom

ACM ISBN 979-8-4007-0445-1/24/05…$15.00
https://doi.org/10.1145/3629527.3651422

1 INTRODUCTION

Many large enterprises have their data servers located across the globe to store their customer data. The conventional way of training models in such scenarios involves collecting data at one data center by transmitting over a wide area network (WAN). However, the prevailing government regulations such as the General Data Protection Regulation (GDPR) enforce user data protection by restricting enterprises from owning data rights [33]. Federated learning [15] (FL) has emerged as a popular solution to address the problem of data islands while preserving data privacy.

GD training based on FL involves training models using data residing in multiple geographies. The need for GD training originates from the distribution and partitioning of data in different regions of the globe to preserve data privacy, government regulations, compliance laws, etc. One of the popular approaches for distributed learning is FedAvg [22] which involves local model training by each client. All participating clients share the gradients with the server which are aggregated and communicated back to each client. Local clients use aggregated gradients to update their models. The model training is a compute-intensive process and requires dedicated resources such as Virtual Machines (VMs) or bare-metal machines via IaaS offering on cloud. However, GD training imposes multiple challenges such as

- Cost escalations due to frequent model sharing across geographies. Additionally, performance degradation and fluctuations are expected as communication overwhelms the low bandwidth of WAN between participating locations [12].
- Capacity planning is a challenge due to heterogeneity in data sizes distributed across multiple locations. An optimal distributed training architecture and placement of resources such as client VMs, and storage services in various geographies is necessary for performance and cost advantages.

These challenges can be mitigated by judiciously choosing cloud resources and services available from cloud vendors. All popular cloud vendors have unique features such as configurations and cost models for the corresponding services provided by them [26] resulting in diverse performance and cost for a given workload.

All cloud service providers share cost models for their services. However, multi-cloud deployment of an application results in complex cost models. A robust cost for a multi-cloud deployment can result in estimating the expenses for multiple deployment scenarios.

As discussed above, aggregators perform computations sporadically when gradients are available from all the clients. Serverless instances are good options for running aggregators due to *pay-as-you-go* cost model. Major Cloud Service Providers (CSPs) provide serverless platforms known as Function-as-a-Service (FaaS) such as AWS Lambda [3], Microsoft Azure functions [23], and Google functions [10]. FaaS billing is based on *pay-as-you-go* such that you pay

only for the actual uses of the resources, unlike IaaS where users are charged for running instances even if they are idle. However, there are a few limitations of serverless instances such as (a) *peer-to-peer* (P2P) communication is not possible between serverless instances and (b) Serverless instances do not have persistent storage. These challenges can be addressed in FaaS using cloud storage services such as AWS S3 [4], and Google Cloud Platform (GCP) storage [9].

Further improvements in performance and cost are possible by using a hierarchical design of aggregators [5, 19, 29]. An intermediate layer is added to aggregate the gradients of the workers from a region or geography before performing a global aggregation.

An in-depth understanding of the performance and cost of GD training architecture in a multi-cloud environment is essential to build an efficient system. Also, apriori knowledge of the system performance and cost of deployment would be advantageous for making judicious decisions. In this work, we analyze GD training architectures using IaaS, FaaS, and storage services from AWS, Microsoft Azure, and Google Cloud Platform (GCP). We also propose a and cost model for estimating the performance and cost of GD training in a multi-cloud and multi-geography environment. Succinctly, our contribution is as follows

- We evaluate multiple Geo-distributed training architectures using IaaS and FaaS services from multiple-cloud vendors.
- We study a hierarchical architecture for aggregating the gradients using a serverless platform. An investigation of the impact on performance and cost due to the placement of the aggregator in a particular geography is presented.
- We present a model for estimating the cost of training using multiple cloud services in a multi-cloud environment.

The rest of the paper is structured as follows. We discuss related work in Section 2. Our architecture and its evaluation are discussed in Section 3. We discuss our cost model in Section 4. The experimental setup and analysis are presented in Sections 5 and 6 respectively followed by the conclusion in Section 7.

2 RELATED WORK

Geo-distributed training is being explored by researchers in both academia and industry [1] [12]. Several distributed training frameworks employing data parallelism, like Horovod [30] and HOGWILD [27], have been created.. An efficient communication library for distributed training of deep learning (DL) models in a public cloud cluster is presented in [31]. Most of the large-scale distributed training frameworks are designed for data distributed within a data center or a region. In very recent work, a framework called Multi-FedLS is proposed for reducing execution time and managing resources on a cloud for Federated Learning applications [6]. The framework provides insights into VM instance selection in multi-cloud but does not consider FaaS serverless platforms.

Previous research has investigated gradient aggregation techniques employing parameter servers [16] and all-reduce methods [18]. A GD training framework called Cloudless-Training based on a parameter server-based approach is presented in [32]. Another framework known as sky computing is designed to accelerate GD computing in a federated learning [35]. Distributed GraphLab [20] is an extension of GraphLab [21] and directly targets asynchronous, dynamic, graph-parallel computation in the shared-memory setting, which leads to network congestion reduction in distributed ML training. However, the mentioned frameworks focus on performance improvement in terms of latency and resource utilization in multiple geographies but do not consider multi-cloud environments. The acceleration of communication in a LAN and WAN environment for geo-distributed learning is presented in [1]. However, the work is focused on geo-distributed training using a single cloud data center in different geographies.

A framework called COSTA [7] is designed for cost monitoring and managing the workload migration to the public cloud. The placement of parameter servers in a wide-area network for geo-distributed machine learning is discussed in [17]. Finding the optimal data storage service in a multi-cloud environment using optimization algorithms is presented in [28]. To the best of our knowledge, there is no prior study on performance comparison cost estimation for ML training using IaaS as well as FaaS cloud services in a multi-cloud and multiple geography environment.

Nebula-I [34] is a framework for collaboratively training DL models over remote heterogeneous clusters specifically GPU and NPU connected via low-bandwidth. Nebula-I successfully helps to launch training tasks over cloud but training the general model is still a challenge. A geo-distributed ML system called Gaia [13] decouples the intra and inter-communication between data centers located in various geographies enabling different communication and consistency models for each. The advantages of multi-cloud deployment for AI workflows have been studied in [24–26]. However, it was primarily for inference workload which had different characteristics as compared to the long-running model training process.

Although few frameworks have been proposed for GD training, very little is known about the performance and cost implications in a serverless and multi-cloud environment. We believe that this work is a pioneering effort to study the performance and cost trade-offs in training models using IaaS, FaaS, and storage services from multiple cloud vendors.

3 OUR ARCHITECTURE

In this section, we discuss the proposed architecture. As shown in Fig. 1, GD training involves following steps

- (Step 1) Workers spread over multiple geographies or regions fetch training data from storage and train the model on a mini-batch from the data.
- (Step 2) On completion of the mini-batch, by all participating workers share gradients or local models with the aggregator or parameter server.
- (Step 3) The aggregator collects the data and sends back the aggregated gradients or model to all the workers.
- (Step 4) Workers update their models and continue with the next mini-batch of the data.

Training in Gradient Descent (GD) can be classified into *centralized* and *de-centralized* architectures. In a centralized architecture, all the workers from different geographies send their model or gradients to one master located in one of the geographies (Fig. 2).

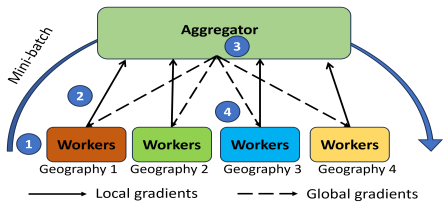


Figure 1: GD training architecture

However, in a decentralized architecture, local updates are aggregated in each geography and then also shared with the master for global updates (Fig.2).

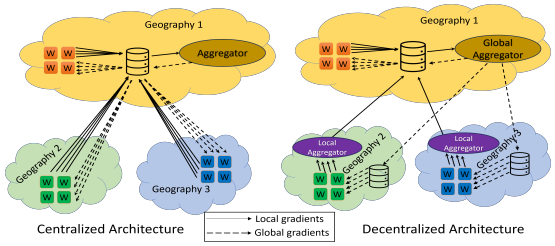


Figure 2: (a) Centralized architecture for GD training (b) Decentralized architecture for GD training

We use LambdaML [14] framework developed for training small ML/DL models in a distributed environment using a cloud serverless platform and storage services specifically in AWS. We have modified the original LambdaML framework to run workers on VM for training large models as well as aggregators using serverless instances in conjunction with storage services. Our modified implementation of LamdaML supports a multi-cloud environment allowing worker VMs, aggregator, and storage services to run on AWS, Azure, and Google cloud platforms. We continue using serverless instances for aggregation for cost savings and high scalability.

In our evaluation, we use an iterative training procedure called FedAvg [22]. Each participating worker trains the model locally and communicates its gradients to the serverless aggregator via cloud storage service. The server aggregates the gradients received from all the clients and synchronizes with all the workers in various geographies.

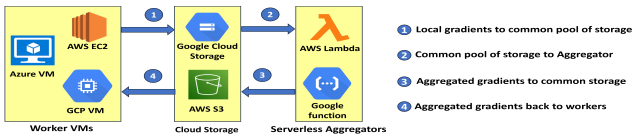


Figure 3: Our architecture for multi-cloud GD training

Our multi-cloud GD training deployment architecture, illustrated in Fig. 3, involves initializing VM instances across various geographies and clouds (AWS, Azure, and GCP) based on data storage service locations and their respective CSPs. Each VM instance acts as a worker, retrieving training data from associated cloud block storage services like AWS S3, Azure storage, or GCP storage. Workers share gradients with an aggregator via cloud storage service (GCP storage or AWS S3) after completing one mini-batch of training data. A serverless instance (GCP function or AWS Lambda) handles gradient aggregation due to restrictions on persistent storage and P2P communication in FaaS. The aggregated gradients are then saved back into the storage service, and all workers update their local models accordingly. This process repeats until convergence.

4 COST MODEL

In order to estimate the total cost of our architecture in a multi-cloud environment for GD training, we developed cost model. The total cost C of multi-cloud geo-distributed training includes the following components - cost for compute ($C_{Compute}$) incurred by workers and aggregator, cost for storing the gradients in storage service ($C_{Storage}$) and the cost for data transfer ($C_{DataTransfer}$). All the rates are cloud vendor-specific and vary with each region.

$$C = C_{Compute} + C_{Storage} + C_{DataTransfer} \tag{1}$$

4.1 Compute Cost

Compute cost ($C_{Compute}$) includes cost due to workers VMs (C_{IaaS}) and aggregator or FaaS (C_{FaaS})

$$C_{Compute} = C_{IaaS} + C_{FaaS} \tag{2}$$

Compute cost for worker VMs (C_{IaaS}): The worker VMs are billed for the entire duration for which it is used. The cost depends on the type of machine, region, and the cloud provider. The cost for IaaS or VMs for n workers is given by

$$C_{IaaS} = \sum_{i=1}^{n} T_{vm\_i} * R_{vm\_i} \tag{3}$$

where T_{vm_i} is the time taken for processing and R_{vm_i} is the cost rate for the i^{th} VM. Different cloud service providers offer different configurations of virtual machines and the cost rates vary for each cloud region. *Compute cost for Aggregator (C_{FaaS}):* Each invocation of the aggregator FaaS function is billed for the memory configured for the execution duration as follows:

$$C_{FaaS} = T_{FaaS} * M_{GB} * R_{mem} \tag{4}$$

where T_{FaaS} is the execution time of the function, M_{GB} is the memory configured for the function and R_{mem} is the billing rate (vendor and geography specific) per memory-time consumed.

4.2 Storage Cost

Storage service is billed for the size of the data stored and the number of operations performed on the storage objects. These operations include writes, reads, deletes, lists, etc. Each operation has a unique billing rate.

Let m be the model or gradient size to be stored in the storage service. Each worker writes a gradient file in the storage, and the aggregator writes the final updated model to storage. Hence the number of writes is $n + 1$, where n is the number of workers. As mentioned in section 3, the aggregator gets triggered when a worker

writes local gradients to the storage. At each of these invocations of the aggregator, it lists all files present in storage to check whether all workers have written their gradients or not. Hence there are n list operations in each batch. The aggregator function reads the gradient files from storage created by each of the workers. Similarly, each of the workers reads the final aggregated gradients from the storage resulting in $n * 2$ read operations. Each worker sends multiple read requests to the storage service till the aggregated gradients are saved by the aggregator and available for download. Although read requests failed till the aggregated gradients were available, these l read requests made to storage incur charges. Hence, there are $(n * 2 + l)$ read requests billed. The aggregator deletes the n model files created by the workers and only the updated model remains in the storage. Delete operations are free from all cloud providers. The total cost for storage per iteration is given as,

$$C_{Storage} = m * R_{st} + ((n * 2 + l) * R_{reads}) + ((n + 1) * R_{writes}) + (n * R_{lists}) \quad (5)$$

where R_{st} is the cost rate per GB per month. R_{reads}, R_{writes}, and R_{lists} are the rates of reads, writes, and lists respectively. Here Rate is for 1000 requests and *Rst* is the cost per GB per month.

4.3 Data transfer Cost

In a multi-cloud and multi-geography scenario, data is transferred across clouds and geographies of various CSPs. Generally, all incoming traffic or ingress is free for all cloud providers. However, all data transfers outside the cloud or geography are billed. The data transfer is billed for the total size of the data. Based on the cloud provider and geographies, either inter-geography or outward traffic rates are applied for the data transfers.

Inter-cloud transfer: When traffic leaves a particular cloud to other cloud providers or outside the internet, it is billed as per internet egress or outward transfer rates of the source cloud geography. Cost of Data transfer ($C_{DataTransfer}$) during an iteration of multi-cloud geo-distributed training consists of outward transfer from workers (C_{DT_IaaS}), aggregator (C_{DT_FaaS}) and storage ($C_{DT\_Storage}$).

$$C_{DataTransfer} = C_{DT\_IaaS} + C_{DT\_FaaS} + C_{DT\_Storage} \quad (6)$$

Transfer from workers: Each worker writes the gradients of size m to the storage. The cost of data transfer for n workers is

$$C_{DT\_IaaS} = \sum_{i=1}^{n} m * R_{vm\_i\_st} \quad (7)$$

where $R_{vm\_i\_st}$ is the cost rate of billing for transfer from the cloud region of i^{th} VM to the storage cloud region.

Transfer from FaaS: Aggregator writes the updated model of size m to the storage

$$C_{DT\_FaaS} = m * R_{agg\_st} \quad (8)$$

where R_{agg_st} is the cost rate of billing for transfer from the cloud region of the aggregator function to the storage cloud region.

Transfer from Storage: Aggregator reads the gradients from storage, one gradient file (size m) per worker. Total n gradient files are transferred to the aggregator FaaS function. Additionally, each of the n workers fetches the updated gradients of size m from storage.

$$C_{DT\_Storage} = (n * m * R_{st\_agg}) + (\sum_{i=1}^{n} m * R_{st\_vm\_i}) \quad (9)$$

where R_{st_agg} is the cost rate of billing for transfer from the cloud region of storage to the aggregator cloud region. $R_{st\_vm\_i}$ is the cost rate of billing for transfer from the storage cloud region to the cloud region of i^{th} VM.

5 EXPERIMENTAL SETUP

We conducted a comprehensive study on the GD training architecture, focusing on two use cases: our in-house recommender system (NISER) [11] using a Graph Neural Network Algorithm on the diginetica [8] dataset and sentiment analysis employing LSTM on the IMDB dataset. The recommender system is trained on 720K sessions and 43K product items, generating 17MB gradients, while sentiment analysis uses 50K movie reviews with 49MB gradients.

Models were trained in a distributed environment across Mumbai, London, Oregon, and Sydney. Workers ran on VMs from Amazon EC2, GCP, and Azure with consistent configurations (2 cores, 8GB memory). FaaS instances from AWS Lambda and GCP function deployed gradient aggregators configured with 1GB memory. Training data was fetched from AWS S3 storage, and gradients were stored on S3 and GCP storage. Completion time and cost for one mini-batch were recorded in all experiments, with cost calculated using CSP billing services.

6 EXPERIMENTAL ANALYSIS

In this section, we perform an analysis of experiments conducted by placing worker VMs and aggregators in various geographies and clouds.

6.1 Data transfer bandwidths

We study the bandwidth available when data is transferred from the worker VMs to a storage service. We consider 4 regions namely London, Mumbai, Oregon, and Sydney resulting in a total of 16 combinations of source and destination. VMs are from AWS, Azure and GCP. Storage services are from AWS and GCP. Hence for each source-destination chosen, there are 6 combinations of VM-storage service. Right side graph of Fig. 4 gives the bandwidth during an inter-geographies transfer, while left side graph shows the bandwidth when the source and destination are in the same geography.

- As expected, intra-region transfers have higher bandwidth as compared to inter-region bandwidths
- For inter-region, the transfer to AWS S3 from any of the three cloud worker VMs (AWS, Azure, GCP) has higher bandwidths than the transfer to GCP cloud storage (Fig. 4).
- For transfers within the same region, transfer to AWS S3 is better than transferring to GCP cloud storage in the majority of cases. The only exceptions are - GCP worker VM to GCP cloud storage transfer in London and Oregon regions (Fig. 4).

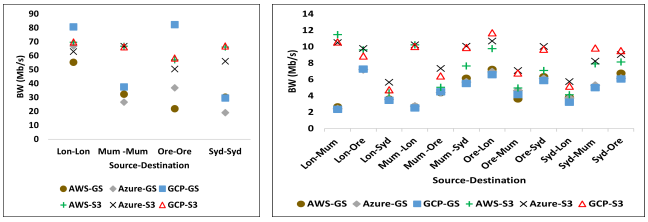


Figure 4: Use case NISER: Upload bandwidth for gradient transfer from worker VM (AWS, Azure, GCP) in source region to Storage service (AWS S3, GCP Cloud storage) in a same and different geography.

6.2 Effect of cloud vendor and aggregator on mini-batch time distribution

As previously mentioned, the total time for one mini-batch includes worker training time, gradient uploading, aggregation time, and aggregated gradient downloading. In this experiment, we investigate how aggregator placement and cloud vendor choice affect each aspect of the total mini-batch time. Specifically, we conduct experiments with all workers in GCP across four geographies: London, Sydney, Mumbai, and Oregon. We measure the mini-batch execution time using Google function and Google storage in each of these locations (5a). The same experiment is then repeated with VM workers in the AWS cloud across the same geographies, utilizing an AWS Lambda instance as an aggregator with AWS S3 storage (5b). We observe the following

- The maximum time in a mini-batch completion is consumed by the aggregator in waiting to receive gradients from all the workers. This is due to the synchronization of workers by the aggregator. This is followed by time spent in sending the gradient from the workers to the aggregator and back.
- Minimum time per mini-batch is consumed when worker VMs, aggregator, and associated storage reside in AWS cloud (Fig.5b) and the maximum time is consumed when worker VMs, aggregators, and storage are from GCP.
- While the total time to complete a mini-batch remains constant within a specific cloud, the duration of each phase varies based on aggregator placement. In Fig. 5a, although the total mini-batch time remains around 14 seconds regardless of the aggregator's region, there are significant variations in the time distribution across different stages when the aggregator's location is altered.

6.3 Effect of aggregator placement and cloud vendor

In this experiment, we study the effect of aggregator placement and the choice of cloud vendor on performance. In this set of experiments, we choose worker VMs for model training in multiple geographies but all from one cloud vendor at a time. However, the aggregator for each of these experiments is chosen from a combination of AWS Lambda, and Google functions with AWS S3 and GCP storage (GS). For example, Fig.6a shows data for worker VMs in AWS distributed in four different geographies. The latency and cost comparison is done by placing aggregator and storage combinations (Lambda+S3, Lambda+GS, GCP+S3, and GCP+GS) in Mumbai, London, Oregon, and Sydney. We observe the following:

- For both the use cases, we get minimum latency by placing VMs in the GCP cloud and using Lambda and S3 for aggregator and storage in Mumbai regions (Fig.6c and Fig.7c).
- For both the use cases, we get the minimum cost of deployment when placing all VMs in AWS and using Lambda and S3 for aggregator and storage respectively in the Oregon region (Fig. 6a and Fig. 7a).
- The choice of GCP function and GCP storage as aggregator results in higher latency in most of the cases irrespective of the cloud chosen for worker VMs.
- As expected, the use of worker VMs, aggregators, and storage from the same CSP is cost-efficient. However, the same is not true for the latency.

These observations can be attributed to the unique cost models and available network bandwidth between cloud services as well as cloud vendors.

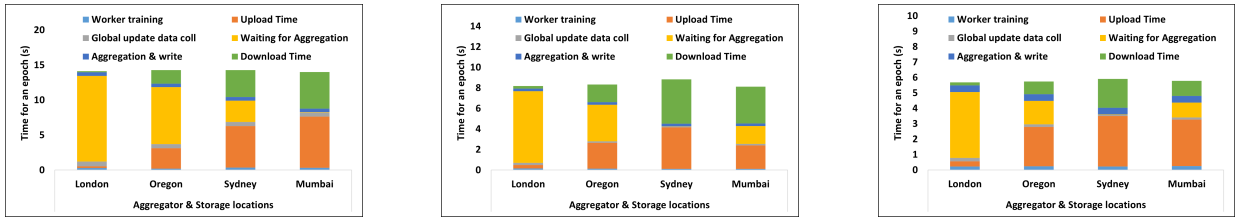
6.4 Effect of aggregator hierarchy

In this experiment, we analyze how the aggregator's placement impacts the cost and performance of training models for the NISER use case. Fig. 8 illustrates the cost and time required to complete one mini-batch when using global and regional aggregators with workers hosted in AWS and GCP clouds. It's evident that having a single global aggregator yields lower latency compared to an architecture employing regional aggregators in each geography alongside a global aggregator. This is mainly because, for a small number of workers, network latency overhead outweighs aggregate computation delays. Also, AWS Lambda instance as an aggregator results in a lower latency as compared to the GCP function instances. This is due to the higher compute capacity and network bandwidth observed in AWS as compared to GCP.

In case we use multiple regional aggregators (Fig. 8), the gradients are aggregated locally and transferred to storage. Hence, the total cost of gradient transfer is due to aggregator to storage transfer in each geography. Unlike the case of one global aggregator where all VMs in each geography transfer data to global storage. The difference between the overall cost for both of these deployments depends on participating geographies. For example, the cost of using only a global aggregator is 50% higher than having regional aggregators when the aggregator and storage are placed in London (Fig. 8) in AWS. This is due to the reason that gradients are coming to London aggregator and storage for aggregation from other geographies and VM egress cost is 5× and 4× higher in Sydney and Mumbai respectively compared to London [2].

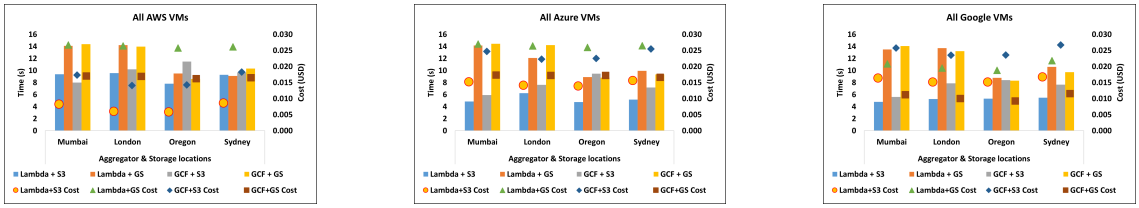
6.5 Cost Model Validation

To validate our cost model with use case NISER, we execute a multi-regional training experiment. This involves deploying 4 workers across distinct geographical regions utilizing AWS EC2 instances. S3 bucket is used as an intermediate storage with Lambda serving as an aggregator at one of the locations. Our experimentation encompassed 100 batches of training, during which we meticulously track costs from AWS cost management console. Furthermore, we



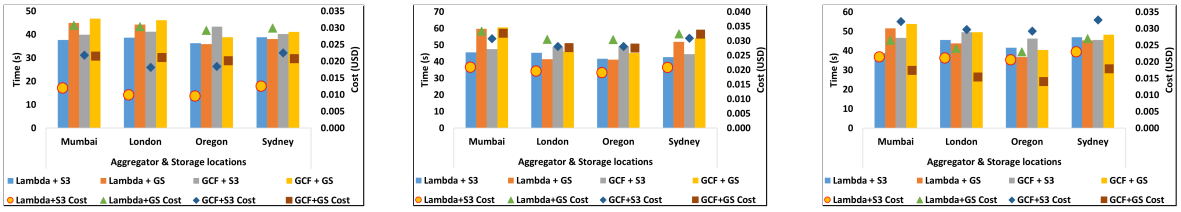
(a) (b) (c)

Figure 5: Use case NISER: Time split for mini-batch execution with various deployment scenarios. (a) Worker VMs (GCP), Aggregator (GCP Functions) and Storage (GCP) (b) Worker VMs (AWS), Aggregator (AWS Lambda) and Storage (AWS S3) (c) Worker VMs (GCP), Aggregator (AWS Lambda) and Storage (AWS S3)



(a) (b) (c)

Figure 6: Use case NISER - One mini-batch completion time comparison when worker VMs run in (a) AWS only (b) Azure (c) GCP cloud. The aggregator (FaaS) and storage combinations are chosen from AWS and GCP and placed in one of the 4 geographies at a time in each of the three clouds



(a) Only AWS VMs (b) Only Azure VMs (c) Only GCP VMs

Figure 7: Use case Sentiment Analysis - One mini-batch completion time comparison when workers VMs run in (a) AWS only (b) Azure (c) GCP cloud. The aggregator (FaaS) and storage combinations are chosen from AWS and GCP and placed in one of the 4 geographies at a time in each of the three clouds

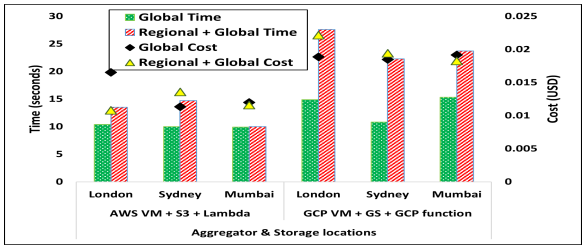


Figure 8: Use case NISER: Regional Aggregator Effect in AWS and GCP

projected costs for an identical experimental configuration using the cost model elucidated in section 4. Our cost model accurately forecasts costs with a marginal error of less than 3.5%.

7 CONCLUSION

In this work, we presented our study on geo-distributed training in a multi-cloud environment. We propose the use of serverless functions as gradient aggregators in conjunction with storage services from multiple CSPs. We study the performance of hierarchical aggregator architecture. Our experiments show that the choice of cloud vendor and placement of aggregators in geo-distributed training has a significant effect on the performance and cost of deployment. Additionally, we presented a cost model for estimating the cost of one mini-batch training in a multi-cloud environment. The proposed cost model predicts the cost of model training with a significant accuracy. We believe that the proposed cost model can be used for estimating the cost of a distributed training architecture in a multi-cloud environment.

REFERENCES

[1] Syeda Nahida Akter and Muhammad Abdullah Adnan. 2020. WeightGrad: Geo-Distributed Data Analysis Using Quantization for Faster Convergence and Better Accuracy. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 546–556. https://doi.org/10.1145/3394486.3403097

[2] Amazon. [n. d.]. AWS Data Transfer. Accessed Sept. 30, 2023. https://docs.aws.amazon.com/cur/latest/userguide/cur-data-transfers-charges.html

[3] Amazon. [n. d.]. AWS Lambda. Accessed Apr. 12, 2023. https://aws.amazon.com/lambda/

[4] Amazon. [n. d.]. AWS S3. Accessed Apr. 12, 2023. https://aws.amazon.com/s3/

[5] Christopher Briggs, Zhong Fan, and Peter Andras. 2020. Federated learning with hierarchical clustering of local updates to improve training on non-IID data. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–9.

[6] Rafaela C Brum, Maria Clicia Stelling de Castro, Luciana Arantes, Lúcia Maria de A Drummond, and Pierre Sens. 2023. Multi-FedLS: a Framework for Cross-Silo Federated Learning Applications on Multi-Cloud Environments. *arXiv preprint arXiv:2308.08967* (2023).

[7] Leandro Costa da Silva, Robson De Medeiros, and Nelson Rosa. 2023. COSTA: A Cost-Driven Solution for Migrating Applications in Multi-Cloud Environments. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing* (Tallinn, Estonia) *(SAC '23)*. Association for Computing Machinery, New York, NY, USA, 57–63. https://doi.org/10.1145/3555776.3577718

[8] GitHub. [n. d.]. Diginetica Dataset. Accessed Sept. 29, 2023. https://darel13712.github.io/rs_datasets/Datasets/diginetica/

[9] Google. [n. d.]. Google Cloud Storage. Accessed Apr. 12, 2023. https://cloud.google.com/storage

[10] Google. [n. d.]. Google Functions. Accessed Apr. 12, 2023. https://cloud.google.com/functions

[11] Priyanka Gupta, Diksha Garg, Pankaj Malhotra, Lovekesh Vig, and Gautam M Shroff. 2019. NISER: normalized item and session representations with graph neural networks. *arXiv preprint arXiv:1909.04276* (2019).

[12] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R. Ganger, Phillip B. Gibbons, and Onur Mutlu. 2017. Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 629–647. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/hsieh

[13] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R Ganger, Phillip B Gibbons, and Onur Mutlu. 2017. Gaia:{Geo-Distributed} machine learning approaching {LAN} speeds. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. 629–647.

[14] Jiawei Jiang, Shaoduo Gan, Yue Liu, Fanlin Wang, Gustavo Alonso, Ana Klimovic, Ankit Singla, Wentao Wu, and Ce Zhang. 2021. Towards Demystifying Serverless Machine Learning Training. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 857–871. https://doi.org/10.1145/3448016.3459240

[15] Mashal Khan, Frank G. Glavin, and Matthias Nickles. 2023. Federated Learning as a Privacy Solution - An Overview. *Procedia Computer Science* 217 (2023), 316–325. https://doi.org/10.1016/j.procs.2022.12.227 4th International Conference on Industry 4.0 and Smart Manufacturing.

[16] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. 2014. Scaling Distributed Machine Learning with the Parameter Server. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. USENIX Association, Broomfield, CO, 583–598. https://www.usenix.org/conference/osdi14/technical-sessions/presentation/li_mu

[17] Yongyao Li, Chenyu Fan, Xiaoning Zhang, and Yufeng Chen. 2023. Placement of parameter server in wide area network topology for geo-distributed machine learning. *Journal of Communications and Networks* (2023).

[18] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William Dally. 2018. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. https://openreview.net/pdf?id=SkhQHMW0W

[19] Lumin Liu, Jun Zhang, S. H. Song, and Khaled B. Letaief. 2020. Client-Edge-Cloud Hierarchical Federated Learning. In *2020 IEEE International Conference on Communications, ICC 2020 - Proceedings*. https://doi.org/10.1109/ICC40277.2020.9148862

[20] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M Hellerstein. 2012. Distributed graphlab: A framework for machine learning in the cloud. *arXiv preprint arXiv:1204.6078* (2012).

[21] Yucheng Low, Joseph E Gonzalez, Aapo Kyrola, Danny Bickson, Carlos E Guestrin, and Joseph Hellerstein. 2014. Graphlab: A new framework for parallel machine learning. *arXiv preprint arXiv:1408.2041* (2014).

[22] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html

[23] Microsoft. [n. d.]. Azure Functions,. Accessed Apr. 12, 2023. https://azure.microsoft.com/en-in/services/functions/

[24] Chetan Phalak, Dheeraj Chahal, Manju Ramesh, and Rekha Singhal. 2023. mSIRM: Cost-Efficient and SLO-aware ML Load Balancing on Fog and Multi-Cloud Network. In *Proceedings of the 13th Workshop on AI and Scientific Computing at Scale using Flexible Computing*. 19–26.

[25] Chetan Phalak, Dheeraj Chahal, and Rekha Singhal. 2023. SIRM: Cost efficient and SLO aware ML prediction on Fog-Cloud Network. In *2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE, 825–829.

[26] Manju Ramesh, Dheeraj Chahal, and Rekha Singhal. 2023. Multicloud Deployment of AI Workflows Using FaaS and Storage Services. In *2023 15th International Conference on COMmunication Systems NETworkS (COMSNETS)*. 269–277. https://doi.org/10.1109/COMSNETS56262.2023.10041365

[27] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems 24*, Vol. 24. 693–701.

[28] Pankaj Sahu, Shubhro Roy, Mangesh Gharote, Sutapa Mondal, and Sachin Lodha. 2022. Cloud Storage and Processing Service Selection considering Tiered Pricing and Data Regulations. In *2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC)*. 92–101.

[29] Mehdi Salehi Heydar Abad, E. Ozfatura, Deniz Gündüz, and Ozgur Ercetin. 2020. Hierarchical Federated Learning ACROSS Heterogeneous Cellular Networks. 8866–8870. https://doi.org/10.1109/ICASSP40776.2020.9054634

[30] Alexander Sergeev and Mike Del Balso. 2018. Horovod: fast and easy distributed deep learning in TensorFlow. *CoRR* abs/1802.05799 (2018). arXiv:1802.05799 http://arxiv.org/abs/1802.05799

[31] Shaohuai Shi, Xianhao Zhou, Shutao Song, Xingyao Wang, Zilin Zhu, Xue Huang, Xinan Jiang, Feihu Zhou, Zhenyu Guo, Liqiang Xie, et al. 2021. Towards scalable distributed training of deep learning on public cloud clusters. *Proceedings of Machine Learning and Systems* 3 (2021), 401–412.

[32] Wenting Tan, Xiao Shi1, Cunchi Lv, and Xiaofang Zhao. 2023. Cloudless-Training: A Framework to Improve Efficiency of Geo-Distributed ML Training. arXiv:2303.05330 [cs.DC]

[33] Paul Voigt and Axel von dem Bussche. 2017. *The EU General Data Protection Regulation (GDPR): A Practical Guide* (1st ed.). Springer Publishing Company, Incorporated.

[34] Yang Xiang, Zhihua Wu, Weibao Gong, Siyu Ding, Xianjie Mo, Yuang Liu, Shuohuan Wang, Peng Liu, Yongshuai Hou, Long Li, et al. 2022. Nebula-I: A general framework for collaboratively training deep learning models on low-bandwidth cloud clusters. *arXiv preprint arXiv:2205.09470* (2022).

[35] Jie Zhu, Shenggui Li, and Yang You. 2022. Sky Computing: Accelerating Geo-distributed Computing in Federated Learning. *arXiv preprint arXiv:2202.11836* (2022).