Incorporating Serverless Computing into P2P Networks for ML Training: In-Database Tasks and Their Scalability Implications (Student Abstract)

Amine Barrak

Department of Computer Science and Mathematics
University of Quebec at Chicoutimi, Québec, Canada
mabarrak@uqac.ca

Abstract

Distributed ML addresses challenges from increasing data and model complexities. Peer to peer (P2P) networks in distributed ML offer scalability and fault tolerance. However, they also encounter challenges related to resource consumption, and communication overhead as the number of participating peers grows. This research introduces a novel architecture that combines serverless computing with P2P networks for distributed training. Serverless computing enhances this model with parallel processing and cost effective scalability, suitable for resource-intensive tasks. Preliminary results show that peers can offload expensive computational tasks to serverless platforms. However, their inherent statelessness necessitates strong communication methods, suggesting a pivotal role for databases. To this end, we have enhanced an in memory database to support ML training tasks.

Introduction

Various topologies have been proposed in the literature to facilitate distributed training, including parameter server and peer-to-peer architectures. Distributed training within peer-to-peer (P2P) networks brings advantages like enhanced scalability and fault tolerance due to the absence of a single point of failure. However, as the network expands, challenges in communication, synchronization, and model updating arise (Guerraoui et al. 2021). Moreover, implementing parallel batch processing in such networks, particularly with frameworks like PyTorch, poses another challenge. These frameworks, depending on the finite resources of individual workers, often grapple with inefficiencies in parallel batch processing. In cases of resource constraints, they might shift to sequential batch processing, prolonging training durations (Kepner et al. 2018).

We propose a peer-to-peer (P2P) machine learning framework built on serverless computing platforms. Recognizing the constraints of limited-resource machines, our research proposes offloading resource-intensive training tasks to serverless platforms. Furthermore, we aim to tackle P2P communication challenges by incorporating in-database ML operations for various components of the distributed training. This approach counters the stateless nature of serverless architectures, reducing frequent database interactions.

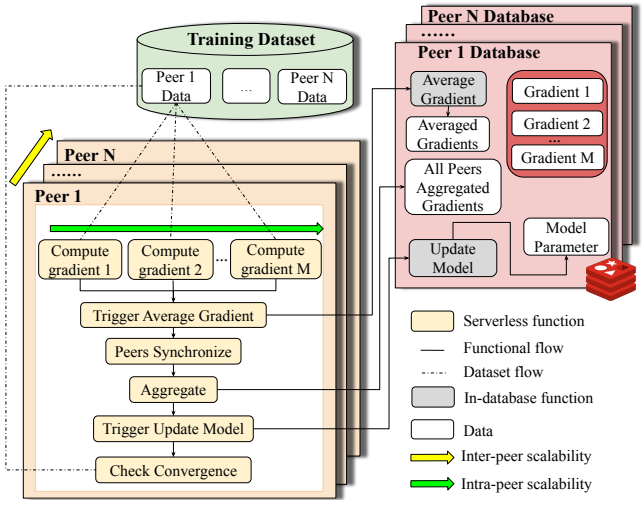

Figure 1: Overview of the proposed Peer To Peer training based on Serverless computing

Serverless P2P Distributed Training

Figure 1 illustrates the proposed serverless Peer-to-Peer (P2P) framework for distributed training. Each peer, identifiable by a unique IP and port, is linked to its stateful component: a dedicated Redis database. Within this framework, the dataset assigned to each peer is partitioned into smaller batches, facilitating parallel gradient computations. Once computed, these gradients are stored in the peer's Redis database, which triggers the averaging of gradients. Subsequently, synchronization among peers is signaled. Peers exchange averaged gradients and aggregate them. Using these aggregated gradients, each peer updates its model parameters within-database using a tailored version of RedisAI. To ensure smooth training progression, we've incorporated a state machine service that orchestrates the flow of each epoch for every peer. This service manages epoch transitions and checks for model convergence every ten epochs.

Our architecture supports dual scalability features: firstly, *intra-peer scalability*, which enables efficient management of simultaneous gradient computations within a single peer; and secondly, *inter-peer scalability*, which provides the capability to adeptly adjust to variations in the total number of peers in the network.

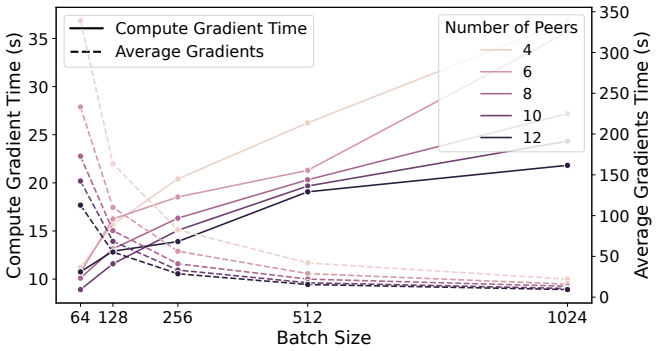

Figure 2: Gradient Computation for DenseNet-121: Impact of Batch Size and Peer Count Variations

Batch size / # of batches	Time (sec)		Cost (USD)	
	WoS	**WS**	WoS	**WS**
64 / 235	394.8	**10.5**	0.01017	**0.05435**
128 / 118	330.4	**12.9**	0.00851	**0.03451**
512 / 30	278.4	**28.1**	0.00717	**0.03069**
1024 / 15	258	**47.8**	0.00665	**0.03567**

Table 1: P2P Training: Comparative Analysis of Time and Cost With (WS) and Without Serverless (WoS)

Experiments

Serverless platforms, renowned for their ability to execute multiple functions concurrently. In our study, evaluating the VGG11 model trained on the MNIST dataset, we compared two architectures: the first, a traditional approach where all computations are performed on a single *t2.large* machine, and the second, a serverless approach that leverages parallel processing without relying on any specific machine instance.

The comparison, detailed in Table 1, shows a significant reduction in computation time for the serverless approach. For example, with a batch size of 64, the computation time dropped from 394.8 seconds to just 10.5 seconds. However, this efficiency comes at a slightly higher cost, with the serverless approach incurring $0.05435 per peer for the same batch size, compared to $0.01017 per peer in the traditional model.

To further understand the efficiency and scalability of serverless Peer-to-Peer (P2P) architectures, we conducted experiments using the Densenet121 model applied to the MNIST dataset, focusing on two critical aspects of scalability. For *intra-peer scalability*, we varied batch sizes from 64 to 1024 to observe the impact on gradient computation times within peers. Secondly, in addressing *inter-peer scalability*, we adjusted the number of peers from 4 to 12 to understand effects on aggregation and training times per epoch.

Our research, illustrated in Figure 2, reveals that increasing batch size leads to higher gradient computation times, but also results in more efficient gradient averaging due to fewer gradients. However, adding more peers doesn't significantly decrease computation time for larger batches, as the benefit is offset by increased network communication. For example, with a batch size of 64, we process 235 batches in parallel. Despite faster gradient computation per batch, the time for averaging escalates, especially with fewer peers. This highlights the need for a balanced approach between inter and intra-peer scalability to achieve optimal scaling in serverless P2P architectures.

Batch size / # of batches	Average time (sec)		Update time (sec)	
	W. DB	O. DB	**W. DB**	O. DB
64 / 235	**717,66**	947,32	**4,8**	27,52
128 / 118	**255,44**	457,24	**4,8**	27,55
512 / 30	**73,35**	120,23	**4,8**	27,44
1024 / 15	**37,41**	67,32	**4,8**	27,52

Table 2: Model Update Times and Gradient Averaging in P2P Training: Within (W) vs. Outside (O) Database

In a subsequent experiment, we evaluated the efficiency of in-database operations for gradient averaging and model parameter updates using the ResNet-18 model. The results, presented in Table 2, highlight the significant time savings achieved by executing these operations within the database. The computed time outside the database, include fetching the data, performing the computation, and saving results back to the database.

For gradient averaging, the in-database approach consistently outperformed the outside database method across all batch sizes. For instance, with a batch size of 1024, gradient averaging within the database took only 37.41 seconds, making a 44% reduction from the 67.32 seconds required outside the database. Similarly, model update times were drastically reduced when processed within the database. The in-database approach consistently took only 4.8 seconds across all batch sizes, marking an impressive 82% reduction compared to the time taken outside the database.

Conclusion

In this study, we propose to employ serverless computing within the Peer-to-Peer (P2P) architecture for distributed ML training. Recognizing the limitations posed by the stateless nature of serverless functions, we integrated databases. Specifically, we enhanced RedisAI to realise ML training tasks directly within the database environment. Results demonstrate a significant time gain by inter scaling (parallelizing the gradient computation tasks) and executing ML training tasks within the database (average gradients and model updates), though it's important to note that this efficiency improvements come with a slightly higher cost compared to traditional methods.

References

Guerraoui, R.; Guirguis, A.; Plassmann, J.; Ragot, A.; and Rouault, S. 2021. Garfield: System support for byzantine machine learning. In *2021 IEEE/IFIP International Conference on Dependable Systems and Networks*, 39–51. IEEE.

Kepner, J.; Gadepally, V.; Jananthan, H.; Milechin, L.; and Samsi, S. 2018. Sparse deep neural network exact solutions. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, 1–8. IEEE.