Efficient Partial Reduce Across Clouds
Renyi Wang  Shouxi Luo  Ke Li  Huanlai Xing
Southwest Jiaotong University
Chengdu, China

CCS CONCEPTS
• Networks → Cloud computing.

KEYWORDS
Partial reduce, flow scheduling, cloud computing

ACM Reference Format:
Renyi Wang  Shouxi Luo  Ke Li  Huanlai Xing. 2022. Efficient
Partial Reduce Across Clouds. In 6th Asia-Pacific Workshop on Networking
(APNet 2022), July 1–2, 2022, Fuzhou, China. ACM, New York, NY, USA,
2 pages. https://doi.org/10.1145/3542637.3543707

1 INTRODUCTION
Geo-distributed machine learning (Geo-DML) systems, which provide
the ability of learning models from massive data across the
globe, have become an essential infrastructure for the design, de-
velopment, and deployment of large-scale artificial intelligence (AI)
services [1, 4]. In these systems, recent studies show that perform-
ing model synchronizations across wide-area networks (WANs)
could dominate the time cost of the entire training; thus, optimiz-
ing the involved communication becomes the key to improve the
efficiency of Geo-DML training [4].

Basically, the synchronization of models can be carried out using
the collective operation of All- or Partial- Reduce, depending on
the training algorithm designs. Partial Reduce is a recently proposed
variant of All Reduce [3]. By allowing \( p \) out of \( n \) workers to conduct
All Reduce operations for a round of synchronization, it is able
to ease the impacts of stragglers with a slowed-yet-controllable
convergence speed, thus promising for data-parallel distributed
training in heterological environments [3]. Then, an interesting
question is: how could we achieve efficient partial reduce for data-
parallel distributed training in the context of inter-cloud Geo-DML,
where workers are hosted in different clouds networked with hetero-
genous WAN connections?

In this work, we propose FMReduce, a flexible and efficient par-
tial reduce implementation to pursue the goal. Compared with
existing solutions, the novelties of FMReduce are two-fold, making it
distinguished for inter-cloud data-parallel training.

Corresponding author: Huanlai Xing (hxll@swjtu.edu.cn). This work was partially
supported by NSFC Project 62002300, NSFC Project 2022NSFC0944, and Project of
Network and Data Security Key Laboratory in Sichuan Province NDS2022-1.

Permission to make digital or hard copies of all or part of this work for personal or
classroom use is granted without fee provided that copies are not made or distributed
for profit or commercial advantage and that copies bear this notice and the full citation
on the first page. Copyrights for components of this work owned by others than ACM
must be honored. Abstracting with credit is permitted. To copy otherwise, or republish,
to post on servers or to redistribute to lists, requires prior specific permission and/or a
fee. Request permissions from permissions@acm.org.
APNet 2022, July 1–2, 2022, Fuzhou, China
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9748-3/22/07. $15.00
https://doi.org/10.1145/3542637.3543707

• Firstly, as the involved computation of reduce is generally
not computationally intensive, instead of employing only the
\( p \) selected workers, FMReduce leverages all the \( n \) available
workers to conduct the operations needed by a partial reduce
(via ”scatter=reduce=broadcast” as Figure 1 shows and
see §2 for details), thus can make efficient usage of all the
abundant inter-cloud WAN connections.

• Secondly, rather than starting the involved transmissions
only after all the \( p \) workers are already established, if desired,
FMReduce allows a worker to scatter the updated model to
others immediately once it is ready, making the spare yet
scarce and expensive WAN link capacities get better usages.

2 FMREDUCE
Overview. As Figure 1 shows, FMReduce (Full-Mesh Reduce) ab-
stracts the inter-cloud network involving \( n \) workers as a fully con-
ected graph, where the directed link from worker \( i \) to worker \( j \)
is with the available bandwidth of \( b_{ij} \) for partial reduce.1 And be-
cause of the presence of coexisting traffic, \( b_{ij} \) is probably unequal
to \( b_{ji} \). To make full use of the processing capacity of all workers
and all the available bandwidth, when worker \( i \) completes its local
training, it splits its model parameters \( W_i \) into \( n \) non-overlapping
blocks: \( W_{i1}, \ldots, W_{i\ell}, \ldots, W_{in} \) for synchronization, respecting the
available bandwidth around the network. Then, the \( j \)-th block \( W_{ij} \)
is scattered to worker \( j \). Note that, the scatter of \( W_{ij} \) will not trigger
network transmissions as the destination and the source are the
same. On getting these blocks, the worker would cache them to
conduct all or partial reduce (Figure 1.b), and broadcast/multicast
the result back once the computation is done (Figure 1.c).

To release the full power of the above design, however, three
problems must be addressed carefully. Next, we briefly introduce
what the problems are and sketch how FMReduce solves them.

1) How to split the model into blocks to make the most usage
of the available bandwidth? Intuitively, each worker should split
its model parameters in the proportion of its up- and down- link
bandwidths to others for the best usage of link capacities. However,
if each worker only considers its own bandwidths, inconsistent

1For each \( i \), we assume that \( b_{ii} \) is set to a large value and would not be the bottleneck.
scatters of the model would occur. To address this, FMReduce computes the splitting scheme upon a global view of the involved links’ available bandwidth.

Let \( v \) be the volume of model \( W \) and \( x_j \) be the size of the \( j \)-th partition \( W_j \) \((1 \leq j \leq n)\) under consistent splitting. Then, obviously, the time costs of the scatter and broadcast phases specified in Figure 1, can be formulated by Eq. (1a) and (1b), respectively.

\[
T_{\text{scatter}} := \max_{i} T^i_{\text{scatter}} := \max_{i} \max_{j} \frac{x_j}{b_{i,j}} = \max_{j} \frac{x_j}{\min_{i} b_{i,j}} \quad (1a)
\]
\[
T_{\text{broadcast}} := \max_{i} T^i_{\text{broadcast}} := \max_{i} \frac{x_i}{\min_{j} b_{i,j}} \quad (1b)
\]

Following this, the upper bound of the communication time involved in a round of partial reduce can be formulated as \( T_{\text{scatter}} + T_{\text{broadcast}} \). To make partial reduce more efficient, FMReduce tries to minimize this upper bound by splitting the model appropriately, according to the results of linear programming (LP) (2). As the implementation, FMReduce lets the worker with the lowest ID compute the splitting scheme based on the updated view of the inter-cloud network and send the results to involved workers on demand.

\[
\text{Minimize } \{T_{\text{scatter}} + T_{\text{broadcast}} : \sum_{i=1}^{n} x_i = n, \text{ where } x_i \geq 0\} \quad (2)
\]

2) How to guarantee consistent partial reduce operation? FMReduce supports both blocking and non-blocking partial reduce. In the blocking mode, workers could not begin the scatter until FMReduce has figured out which \( p \) workers are involved and has decided how to split the model based on that; while in the non-blocking mode, workers could immediately start the scatter following predefined splitting plans. When performing non-blocking partial reduce for \( p < n \), despite the launch of scatters being non-blocking, FMReduce still needs to decide which \( p \) workers could participate in a round of synchronization and ensures that all workers do agree with this for consistency. To this end, FMReduce configures the worker with the smallest ID to act as the leader. Once getting \( p \) scattered model blocks from \( p \) workers (including that of itself if it is ready), it broadcasts which \( p \) are involved in this round to all workers. On getting this information, each worker conducts partial reduce on the selected \( p \) blocks, then broadcasts the result back to these corresponding \( p \) workers immediately, or later in case that some blocks are not obtained yet, as Figure 1b and Figure 1c show.

3) How to schedule concurrent transfers for the optimization of training QoS? Sometimes, flows triggered by the new scatter might compete with flows belonging to a previous ongoing broadcast on some links. It is obvious that, in such a scenario, the default max-min fairness (i.e., FS) is sub-optimal for the allocation of bandwidth; and instead, the principle of First-Come-First-Service (FCFS) is a simple-yet-efficient design.

3 PRELIMINARY EVALUATION

Mainly, FMReduce embodies four types of alternative designs for the implementation of partial reduce:

- **p vs. n** workers used for each partial reduce;
- **Blocking vs. non-blocking** scatter for partial reduce;
- **Evenly-divided vs. network-aware** model splits/scatters;
- **FS vs. FCFS** flow scheduling for concurrent scatters/broadcasts.

![Figure 2: FMReduce vs Original Partial Reduce when p = 50](image)

To verify their benefits, we develop a flow-level simulator with Python 3, which precisely simulates the behaviors of partial reduce operations with and without the enhancement of FMReduce. We conduct 12 comparative simulation experiments and assess them by the number of training rounds in a fixed period of time.

**Network and workloads.** We consider a geo-DML environment involving 60 workers, in which, the available bandwidths of inter-cloud connections (i.e., \( b_{i,j} \)) are calculated by \( k \times u \), where \( k \) is an integer in the range of \([1, 10]\) sampled from \( N(5, 1) \) and \( u \) is the minimum bandwidth unit with the value of 25Mbps. We assume that the model is with the size of 200MB; the time of workers’ each round of local training follows the distribution reported by [2].

**Case Study.** As shown in Figure 2, when other conditions keep the same, network-aware and FCFS scheme can always efficiently increase the train rounds compared to evenly-divided and FS. Regarding blocking and non-blocking, non-blocking performs better than blocking unless the model splitting scheme is network-aware. This is because the composition of \( p \) workers can be known under blocking, which can help to calculate a more precise model splitting results according to the bandwidths of the \( p \) workers when \( p < n \). However, if \( p = n \) exactly, the composition of \( p \) workers is known in advance, then non-blocking can reduce the waiting time without cost, outperforming blocking.

Most importantly, as Figure 2 shows, by employing all \( n \) workers to conduct each partial reduce, FMReduce achieves a remarkable speedup for model synchronization compared to the original partial reduce.

4 CONCLUSION

We have presented FMReduce, an efficient partial reduce implementation for inter-cloud Geo-DML training. By balancing the involved communication and reduce operations among all available workers respecting the state of inter-cloud connections, and conducting FCFS flow scheduling, FMReduce could make effective usage of the available inter-cloud network to achieve efficient partial reduce.

REFERENCES


