A Workflow and Comparison of RAR, PS, and INA

An all-reduce operation, commonly used in High-Performance Computing (HPC), is a collective communication primitive that performs reductions on data across nodes and writes the result to each node. It can be implemented in different ways. A halving/doubling algorithm is employed in [19] which has two major drawbacks: 1) the overhead of data transfer is doubled for non-power-of-two case [72]; 2) the communication pattern involved may lead to network contention [58]. Baidu [18] introduces ring all-reduce (RAR) to deep learning, and it has become the most popular algorithm ever since as contention-free communication can be achieved.

Suppose that a homogeneous distributed DNN training system has P ($P \geq 2$) GPUs, and each machine is equipped with one GPU. To synchronize data with size M by using RAR, each node splits the data into P blocks and transmits a data block with a size of $\frac{M}{P}$ at each step. In the example in Figure 18, $P = 3$ and the data volume is divided into 3 pieces: a, b, and c.

An RAR process consists of two phases: `scatter-reduce` and `all-gather` (Figure 18). In scatter-reduce, the k^{th} node starts by sending the k^{th} block to its successor, and simultaneously receiving the $(k-1)^{th}$ block from its predecessor and aggregating it with the local one. After two steps, partial aggregation result on a specific block in each node (highlighted boxes after the 2^{nd} step in Figure 18) is obtained. In all-gather, the k^{th} node starts by sending the $(k+1)^{th}$ block, and receiving the k^{th} block and replacing the corresponding local block with the received one. Each node takes another two steps to finally obtain the entire aggregation result.

In summary, an RAR procedure completes in $2(P-1)$ steps with $\frac{2(P-1)}{P}M$ amount of data transmitted per node. As P increases, this amount is nearly twice the original model size. The time taken to complete a ring all-reduce operation can be modeled [72] as

$$T_{ring} = 2(P-1)\alpha + \frac{2(P-1)}{P}\frac{M}{B}$$

where α is the latency per message independent of M, including the time taken for data preparation and sending interface calls, etc.; B is the network bandwidth.

In-Network aggregation (INA) offloads gradients aggregation into the network switch. In RAR, each node receives different data (Figure 2) from the other node, which is unaggregated and the same as what its neighbor sends. In INA, on the contrary, different nodes receive the same data, which is the aggregation result from all nodes (Figure 3).

Compared to RAR, the communication cost of INA is independent of the number of nodes P. In RAR, each node needs to transmit different pieces of data to optimize bandwidth utilization. Instead, the INA switch performs gradients aggregation by letting all nodes send the same piece of data.

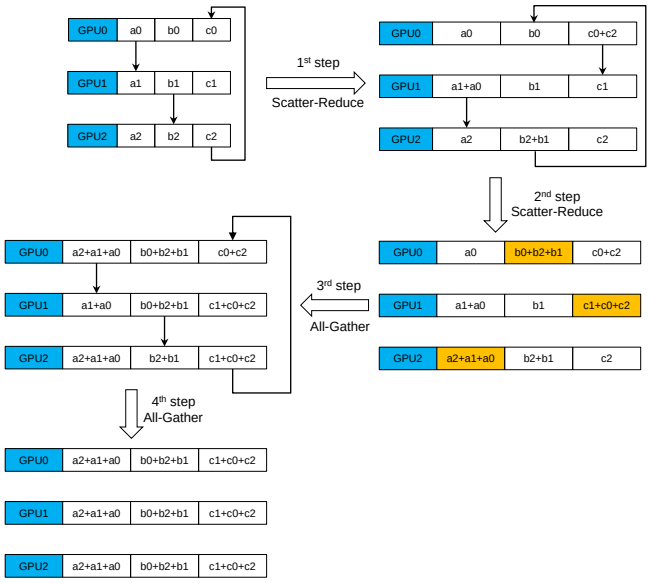

Figure 18. Illurstration of RAR process.

The communication cost of INA can be modeled as

$$T_{INA} = \alpha + \frac{M}{B}$$

Unlike that RAR transmits the message $2(P-1)$ times, INA only transmits once, reducing the latency from $O(P)$ to $O(1)$. Additionally, the data amount transmitted by each node is reduced from $\frac{2(P-1)}{P}M$ to M, by nearly 50% as P increases.

INA applies in the Parameter Server (PS) architecture by replacing the PS with a switch. PS-based approaches generally work in a "push + pull" way. At each iteration, all workers first push their computed gradients to a PS. Then the PS aggregates the gradients and updates the model with new weights. Finally, workers pull the updated weights from PSs and start the next computing iteration. PS can easily become a bottleneck: the push phase leads to the traffic incast, and the pull phase results in data redundancy in the network. Moreover, the benefits of using PS depend on the additional CPU resources provided [3, 31]. In a homogeneous training system (i.e., all the machines are equipped with the same number of GPUs), the PS does not save anything.

One thing worth mentioning is that INA functions like PS without putting new traffic into the network. PS completes an iteration in two RTTs: one for gradients (and their ACK) and one for results (and their ACK); and INA uses only one RTT for grediants (and results acting as ACK). A comparison of PS, RAR, and INA on traffic volume is summarized in Table 8.

B Modeling Communication

Modeling. We define symbols in Table 9 to model the communication time in FR, TA, and HN. In the multi-machines

Table 8. Traffic volume of PS, RAR, and INA in one iteration.

	PS	RAR	INA
Worker	M	$2\frac{P-1}{P}M \approx 2M$	M
PS	PM	-	-

Table 9. Symbols and their meaning.

Symbols	Meaning
M	The size of the gradient
n	Number of GPUs per machine
P	Total number of GPUs in a job
α	Per-hop latency in a ring for data preparation.
N	The window size
B_{intra}	Intra-machine bandwidth
B_{inter}	Inter-machine bandwidth, i.e., network bandwidth
T_{FR}	Communication time of one iteration in Flat Ring
T_{TA}	Communication time of one iteration in Tencent AllReduce
T_{NR}	Communication time of one iteration in NetAR

multi-GPUs scenario, the communication time taken by using the flat ring all-reduce algorithm is modeled as

$$T_{fr} = 2(P-1)\alpha + 2\frac{P-1}{P}\frac{M}{B_{inter}} \quad (1)$$

where B_{inter} refers to the inter-machine bandwidth where machines are connected via computer networks such as Ethernet or InfiniBand.

For Tencent all-reduce, consider Rabenseifner's reduce algorithm [60] and Van de Geijn's broadcast algorithm [5], and assume n is a power of 2, the communication cost can be modeled as

$$\begin{aligned} T_{tr} &= T_{tr1} + T_{tr2} + T_{tr3} \\ &= \left[2\alpha\log_2(n) + \frac{2(n-1)}{n}\frac{M}{nB_{intra}}\right] \\ &\quad + \left[2(\frac{P}{n}-1)\alpha + 2\frac{P/n-1}{P/n}\frac{M}{B_{inter}}\right] \\ &\quad + \left[(\log_2(n)+n-1)\alpha + 2\frac{n-1}{n}\frac{M}{B_{intra}}\right] \\ &= \frac{n^2+3n\log_2(n)-3n+2P}{n}\alpha \\ &\quad + \frac{4(n-1)PB_{inter}+2(P-n)nB_{intra}}{nPB_{intra}B_{inter}}M \end{aligned} \quad (2)$$

where B_{intra} refers to the intra-machine bandwidth where GPUs are connected via expansion bus such as PCIe or NVLinks.

The communication cost of hierarchical NetAR is given as

$$\begin{aligned} T_{nh} &= T_{nh1} + T_{nh2} + T_{nh3} \\ &= \left[(n-1)\alpha + (n-1)\frac{M}{nB_{intra}}\right] + \left(\alpha + \frac{M}{B_{inter}}\right) \\ &\quad + \left[(n-1)\alpha + (n-1)\frac{M}{nB_{intra}}\right] \\ &= (2n-1)\alpha + \frac{2(n-1)B_{inter}+nB_{intra}}{nB_{intra}B_{inter}}M \end{aligned} \quad (3)$$

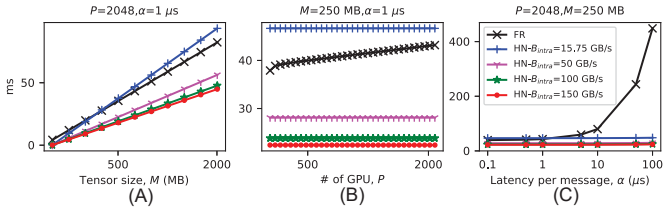

Figure 19. Communication cost taken by a single machine for parameter synchronization (n=8, B_{inter}=12.5 GB/s, varying B_{intra}): (A) v.s. Tensor size (M); (B) v.s. number of GPU (P); (C) v.s. latency per tensor (α).

When $n = 1$, $B_{intra} = B_{inter} = B$, Eq.(3) reduces to the single-GPU case as $T_{inet} = \alpha + \frac{M}{B}$.

Eq.(2) subtracting Eq.(3) gives

$$\begin{aligned} \Delta T_{tr-nh} &= T_{tr} - T_{nh} \\ &= (2P/n + 3\log_2(n) - n - 2)\alpha \\ &\quad + \frac{(P-2n)nB_{intra}+2(n-1)PB_{inter}}{nPB_{intra}B_{inter}}M \end{aligned} \quad (4)$$

When $P > 3n$, (4) is always larger than 0, considering n is usually no larger than 16.

Comparison. Eq.(1) subtracting Eq.(3) gives

$$\begin{aligned} \Delta T_{fr-nh} &= T_{fr} - T_{nh} \\ &= (2P - 2n - 1)\alpha \\ &\quad + \frac{(P-2)nB_{intra}-2(n-1)PB_{inter}}{nPB_{intra}B_{inter}}M \end{aligned} \quad (5)$$

Similarly, we can obtain a relaxed sufficient condition from (5) that hierarchical NetAR outperforms flat ring all-reduce on communication as follows

$$\frac{B_{intra}}{B_{inter}} \geq \frac{2P}{P-2} \qquad (P > n \geq 2) \quad (6)$$

We get the sufficient conditions where HN outperforms FR and TA: $P > 3n$ and $\frac{B_{intra}}{B_{inter}} \geq \frac{2P}{P-2}$ $(P > n \geq 2)$. In production network, the first is not hard to achieve, e.g., our testbed has $P = 32$ and $n = 8$; and the latter can be achieved with recent progress of intra-machine GPU interconnection: NVLink makes B_{intra}=150GB/s and typical high-speed Ethernet is B_{inter}=100Gbps.

Simulation. We conduct flow-level simulation to understand the impact factors influencing the communication time. We simulate a multi-machine multi-GPU cluster with $n = 8$ and $B_{inter} = 12.5GB/s$ (Ethernet), compare NetAR with Flat Ring, and tune the intra-machine bandwidth B_{intra} from 15.75GB/s (16-lane PCIe 3.0) to 150GB/s (NVLink), total number of GPUs P from 32 to 2048, and per-hop latency on a ring α from 0.1μs to 100μs. Part of the simulation results are shown in Figure 19.

First, FR's communication time varies with P and α, but NetAR's does not (Figure 19(B) and 19(C)). The reason is that FR has a ring structure, and the total latency on a ring is

decided by the number of hops P and the per-hop latency α. But NetAR intra-machine scatter-reduce and all-gather is one hop, and the inter-machine aggregation re-organizes the (logical) ring into a physical aggregation hierarchy with limited hops.

Second, for typical tensor transmission, the data transmission time dominates over the latency. A model of size 100X MB transmitted on a 10s GB/s link costs 10s ms; but a typical per-hop latency is 1s-10s μs. Increasing B_{intra} to reduce the transmission time could effectively reduce the overall time. For example, in Figure 19(A) and 19(B), with B_{intra} larger than 50GB/s, NetAR always outperforms FR.

C Additional Implementation Details about Accelerator

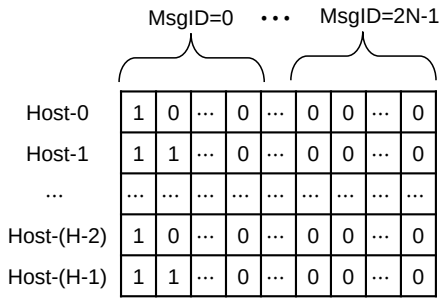

Figure 20. Arrival states of packets for each ring.

The internal architecture of the FPGA accelerator is shown in Figure 21. The bitmap, packet buffer, and value in aggregators are implemented as separate arrays — `State Record`, `Header Buffer`, `Payload Buffer`, and `Aggregation Value`. When a packet arrives, a `Parser` identifies the aggregation packet or directs the other kinds of the packet to the output port directly. The `Parser` further feeds the NetAR header to a `State Manager` which tracks the arrival states of the packets, i.e., the bitmaps of aggregators. Figure 20 shows the data structure to track the arrival states: it is a matrix of bits with the row index indicating a host and column index indicating packet sequence number (each column is a bitmap). The `Aggregation Value` is the array of tensor values field of aggregators. `Header Manager` is in charge of packet address translation. `Combinator` merges the header with the payload and sends the final packet out.

D Further Interpretation of Figures 16 and 17

In Figures 17 and 16, various batch sizes and floating-point precisions are chosen. Considering the specific case of BS=32 and FP16, the training performance is summarized in Table 10 and 11, respectively.

In Table 10, hierarchical NetReduce improves flat ring by 68.8%, 50.7% and 15.1% for AlexNet, VGG-16, and ResNet-50, respectively. Compared with Tencent all-reduce, hierarchical NetReduce speeds up training by 57.9%, 42.1%, and 12.3% for the three models, respectively. It is reported in [30] that Tencent hierarchical algorithm only brings performance gain for tensors with smaller sizes. For relatively larger tensors, the flat ring algorithm still outperforms the hierarchical algorithm. Recall in § B, we model the communication cost of each algorithm as a combination consisting of two items: message processing latency item with α and tensor transmission item with M. The α item is mostly affected by the number of GPUs participating in training, P. With increased P, the α item accounts for a larger proportion in flat ring all-reduce, resulting in poor scalability. Hierarchical approaches reduce the impact of α item by dividing a big ring into multiple small intra rings, improving the scalability. Therefore, for small tensors where the α item accounts for most communication cost, hierarchical approaches give superior performance. However, for big tensors where the M item accounts for most communication cost and the system becomes less sensitive to P, hierarchical approaches bring fewer benefits.

When the intra- and inter- machine bandwidth fulfill certain conditions, hierarchical approaches can outperform the flat ring regardless of tensor size. Specifically, hierarchical NetReduce would always outperform flat ring if condition (6) holds. Considering our hardware prototype, substituting P=32 and n=8 into (6) gives $\frac{B_{intra}}{B_{inter}} \geq 2.3$. Indeed intra and inter nodes being connected via NVLink and 100GbE, gives B_{intra} = 150 GB/s and B_{inter} = 12.5 GB/s, respectively. Therefore, in our hardware prototype, $\frac{B_{intra}}{B_{inter}}$=12 > 2.3.

In Table 11, NetRedcue improves AlexNet on the throughput by 35.6%, which is the most. This is because when using

Table 10. Training performance in Figure 17 (32 GPUs) with with BS=32 and FP16.

Model		Flat ring all-reduce	Tencent all-reduce	Hierarchical NetReduce
AlexNet	Images/s	307.5	328.8	519.2
(236 MB)	↑	68.8%	57.9%	-
VGG-16	Images/s	115.2	122.2	173.6
(528 MB)	↑	50.7%	42.1%	-
ResNet-50	Images/s	276.0	282.8	317.6
(98 MB)	↑	15.1%	12.3%	-

Table 11. Training performance in Figure 16 (4 GPUs) with BS=32 and FP16.

Model		Throughput (images/s)	Iteration (ms)	Communication (ms)
AlexNet	Ring all-reduce	527.9	60.62	47.12(77.7%)
(236 MB)	NetReduce	716.0	44.69	31.10(69.6%)
	↑	35.6%	26.3%	34.0%
VGG-16	Ring all-reduce	172.9	185.08	111.98(60.5%)
(528 MB)	NetReduce	215.3	148.63	74.64(50.2%)
	↑	24.5%	19.7%	33.3%
ResNet-50	Ring all-reduce	358.8	89.19	23.04(25.8%)
(98 MB)	NetReduce	383.6	83.42	19.29(23.1%)
	↑	6.9%	6.5%	16.3%

the ring all-reduce algorithm to train AlexNet, the time taken for communication occupies 77.7% (=47.12/60.62 as shown in the 5^{th} column in Table 11) of the whole iteration time, which has a significant potential to improve. Indeed, NetReduce improves AlexNet in communication by 34.0%. On the contrary, although VGG-16 is improved on communication by 33.3%, which is similar to AlexNet, the communication part occupies 60.5%, which is smaller than AlexNet, resulting in a smaller improvement in total training throughput (24.5%). Especially for ResNet-50, which is a computation-intensive model, with 16.3% improvement on the communication part, which accounts for only 25.8% of the iteration time, we only have 6.9% improvement on the training throughput.

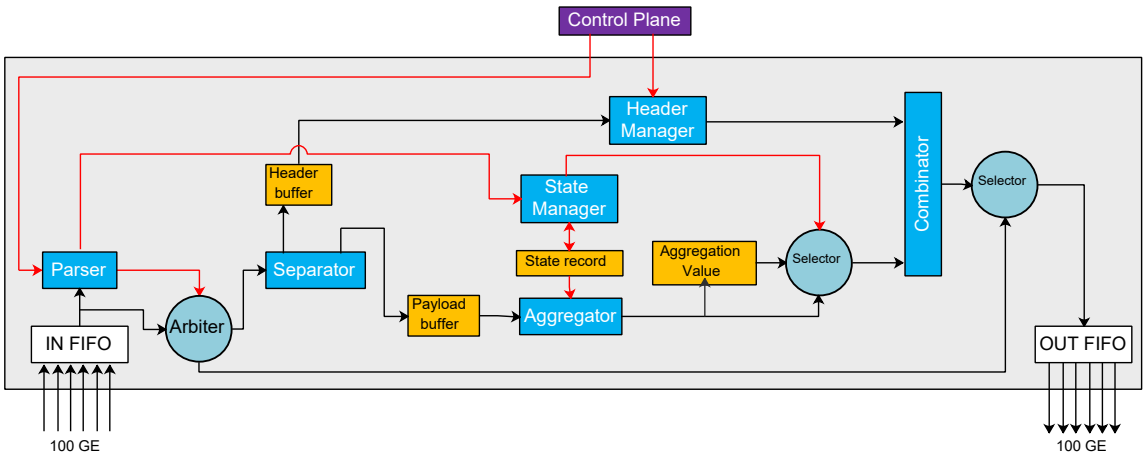

Figure 21. The accelerator architecture of in-network reduction (red and black arrow lines refer to control and data flows, respectively).