Redundant Hierarchical Ring All-Reduce in Hypercubes

Huimei Guo, Shuo Quan, Rong-Xia Hao, and Jie Wu, *Fellow, IEEE*

***Abstract*—Distributed training of large language models demands efficient gradient synchronization strategies. Existing All-Reduce algorithms often fail to fully leverage underlying topological properties, resulting in low edge utilization and unbalanced load in hypercube networks. This paper proposes a redundant hierarchical ring all-reduce algorithm tailored for hypercube topologies. The algorithm first designs a hierarchical Ring All-Reduce structure with high edge utilization by exploiting the hypercube's inherent high parallelism and symmetry, proving that the optimal number of hierarchical layers is $\lfloor \frac{n}{2} \rfloor$. It then introduces controlled redundancy, dynamically adjusting transmission strategies based on the relationship between the value of bandwidth B and the value of data volume D to reduce communication hops and transmission latency while ensuring reliability. Experiments demonstrate that our algorithm achieves the minimum number of gradient synchronization hops under varying bandwidth constraints, with end-to-end transmission time significantly outperforming existing schemes. Further reliability simulations verify its effectiveness in balancing redundancy to prevent resource waste while maintaining high reliability.**

***Index Terms*—Distributed deep learning, hierarchical design, hypercubes, reliability, ring all-reduce (RAR).**

NOMENCLATURE

Acronyms

LLMs	Large language models.
ML	Machine learning.
DLMs	Deep learning models.
HD	Halving-doubling.
BR	Baisc ring.
RAR	Ring all-reduce.
HAR	Hierarchical all-reduce.
2D-TAR	2-D-torus all-reduce.
HRA	Hierarchical ring all-reduce.
OHRA	Optimized hierarchical ring all-reduce.
2D-HRA	2-D-hierarchical ring all-reduce.
FRAR	Full ring all-reduce.
RHRA	Redundant hierarchical ring all-reduce.

Received 1 November 2025; revised 9 January 2026; accepted 21 February 2026. Date of publication 26 February 2026; date of current version 26 March 2026. The work of Rong-Xia Hao was supported by the National Natural Science Foundation of China under Grant 12471321 and Grant 12331013. Associate Editor: Y. Lin. *(Corresponding authors: Rong-Xia Hao; Jie Wu.)*

Huimei Guo is with the School of Mathematics and Statistics, Beijing Jiaotong University, Beijing 100044, China, and also with China Telecom Cloud Computing Research Institute, Beijing 100191, China (e-mail: luckyhuimeiguo@163.com).

Shuo Quan is with China Telecom Cloud Computing Research Institute, Beijing 100191, China (e-mail: quansh@chinatelecom.cn).

Rong-Xia Hao is with the School of Mathematics and Statistics, Beijing Jiaotong University, Beijing 100044, China (e-mail: rxhao@bjtu.edu.cn).

Jie Wu is with the China Telecom Cloud Computing Research Institute and the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: jiewu@temple.edu).

Digital Object Identifier 10.1109/TR.2026.3668208

I. INTRODUCTION

A. Background

WITH the exponential growth of data driven by the internet and sensor technologies, traditional ML algorithms face scalability challenges, necessitating distributed computing solutions [26]. This need is further intensified by advanced DLMs, especially LLMs with billions to trillions of parameters, which incur massive computational costs and communication bottlenecks during distributed training. Recent research focuses on optimizing distributed training protocols, particularly gradient synchronization.

Gradient descent is fundamental in large-scale ML/DL, where gradient synchronization is critical for parameter updates across distributed nodes. Research addresses this through algorithm design [10], system optimization [27], and performance evaluation [24].

The performance of communication algorithms is heavily influenced by network topology. TidalMesh introduces a topology-driven All-Reduce approach for mesh networks [17], demonstrating how adapting algorithms to topology enhances efficiency. The n-dimensional hypercube Q_n is particularly promising due to its excellent scalability, fault tolerance, high symmetry, and short paths, enabling efficient All-Reduce operations [3], [18], [21].

All-Reduce is a common gradient synchronization approach favored for its efficiency and scalability. Relevant research progress is summarized in Table I. For more research on All-Reduce, please refer to references [28], [30]

In recent years, research on optimizing All-Reduce algorithms, topology-adaptive algorithms, and mitigating communication bottlenecks in LLMs has become a focal point. For instance, the StragglAR algorithm improves All-Reduce efficiency by reducing communication steps [6]. In addition, the communication-efficient network topology algorithm significantly reduces latency by eliminating unnecessary communication links [25]. In LLM training, the cloud-edge collaborative framework effectively alleviates communication bottlenecks through cloud-edge collaboration [15], and a survey of decentralized LLM training emphasizes the need for optimizing communication efficiency [7]. These studies provide important theoretical foundations and practical guidance for understanding and addressing communication challenges in distributed training.

TABLE I
COMPREHENSIVE COMPARISON OF RAR ALGORITHMS

Algorithm	Core Idea	Key Optimization	Reference
BR	Sequential data circulation in a single logical ring.	Simple implementation, load imbalance.	Classic
RAR	Two phases (reduce-scatter and all-gather) ring-based procedure.	Bandwidth optimality, widely adopted.	Baidu [5]
HD	Power-of-two distance communication with halving/doubling data volume	Minimizes hop count, but poor link utilization.	Alibaba
HAR	Hierarchical group-based reduction (intra-group then inter-group).	Reduces effective ring size for large clusters.	Tencent [11], [13]
2D-TAR	Row-first then column reduction on 2-D torus/mesh topology.	Exploits 2D physical topology for shorter rings.	Sony [19]
2D-HRA	Combines hierarchical and 2-D torus.	Further reduces steps for specific topologies.	Jiang et al., [14]
HRA	Multi-layer ring decomposition with sequential processing.	Reduces per-ring size, adapts to layered networks.	Ueno [23]
IBing	Concurrent bidirectional data transmission to left/right neighbors.	Halves communication steps, improves bandwidth utilization.	Zong et al. [29]
DirectReduce	SmartNIC-offloaded RAR computation for torus topologies.	Eliminates CPU/GPU interrupts, reduces latency.	Hui et al. [12]
FRAR	Full-data transmission per hop when $B \geq D$.	Minimizes hops via full redundancy, for high bandwidth.	Proposed in Sec. V(Algorithm 2)
OHRA	Concurrent multi-layer execution on hypercube using edge-disjoint cycles.	Maximizes link utilization, hypercube-specific.	Proposed in Sec. III(Algorithm 1)
RHRA	Tunable-redundancy extension of OHRA, adapts to bandwidth B.	Optimizes redundancy-hop trade-off under constraints, hypercube-specific.	Proposed in Sec. V (Algorithms3,4,5)

B. Importance of Ring Structure and Limitations in Hypercube Topologies

The aforementioned All-Reduce algorithms are predominantly ring-based, a topology particularly critical in large-scale clusters due to its concise communication pattern, efficient bandwidth utilization, and inherent load-balancing characteristics. By restricting communication to adjacent nodes, the ring structure shortens data transmission paths, minimizes latency, and avoids bottleneck formation.

Nevertheless, when implemented on the n-dimensional hypercube Q_n, existing ring-based All-Reduce algorithms fail to fully exploit the topology's high parallelism and structural symmetry. For instance, while the HD algorithm minimizes hop count, it suffers from low edge utilization and unbalanced transmission. Similarly, algorithms, such as HAR, 2D-TAR, 2D-HRA, and HRA, primarily activate edges within a single layer, leaving interlayer connections under utilized and thus overlooking the intrinsic parallelism of Q_n. Consequently, developing optimized All-Reduce algorithms that better leverage the hypercubes topological advantages remains a research challenge of significant importance.

C. Contributions

In this article, the main contributions are as follows.

1) We formally define the h-hierarchical cycle decomposition for the hypercube Q_n and prove its existence (see Theorem 3.3). Furthermore, we establish the optimal number of layers that minimizes the communication hops for the h-OHRA algorithm (see Theorem 4.1).
2) We propose the h-OHRA algorithm, which enhances edge utilization and load-balancing by concurrently processing data segments across all hierarchical layers. Building upon h-OHRA, we introduce the RHRA algorithm, which innovatively employs redundancy not only for fault tolerance but as a tunable parameter to optimize the trade-off between communication hops and data volume under bandwidth constraints.
3) We conduct an extensive empirical study comparing RHRA against seven existing All-Reduce algorithms. Our evaluation spans three critical metrics: communication hops under varying bandwidths, end-to-end latency in a simulated realistic network, and transmission reliability under probabilistic failures.
4) Through reliability simulations on a large-scale Q_{14} topology, we provide a nuanced analysis of the redundancy-reliability tradeoff, offering practical guidance for configuring the algorithm to avoid over-provisioning while ensuring high data integrity.

The rest of this article is organized as follows. Section II introduces preliminaries and four basic All-Reduce algorithms. Section III presents the h-hierarchical cycle decomposition of Q_n. Section IV derives the optimal number of layers and compares the five nonredundant algorithms in terms of per-hop data transmission. Section V proposes the RHRA algorithm under different bandwidth regimes. Section VI evaluates RHRA against seven existing algorithms through hop-count, end-to-end latency, and reliability experiments. Finally, Section VII concludes this article.

II. PRELIMINARY AND RELATED WORKS ON ALL-REDUCE OPERATION

The topology of large-scale parallel distributed systems can be abstracted as a graph G, where processors are considered as nodes of the graph and physical connections are seen as edges of the graph. Let $V(G)$ and $E(G)$ represent the vertex set and edge set of G, respectively. A cycle that contains all vertices of a graph G is called a *Hamiltonian cycle*. In the rest of this article, for any positive integer n, define $[n] = \{1, 2, \ldots, n\}$. We refer to [2] for notations that are not described here specifically.

Definition 2.1: The *n-dimensional hypercube* Q_n has 2^n vertices and $n2^{n-1}$ edges so that each vertex $u \in V(Q_n)$ is identified by an n-bit binary string, represented by $u = u_n u_{n-1} \cdots u_2 u_1$ for $u_i \in \{0, 1\}$ and $i \in [n]$. If u and v in Q_n differ in precisely one position, then two vertices u and v are adjacent (i.e., $uv \in E(Q_n)$).

Definition 2.2: For an integer n, let G be a n-regular graph. It is said to have a *Hamiltonian decomposition* if either

i) n is even and $E(G)$ can be partitioned into $\frac{n}{2}$ edge disjoint Hamiltonian cycles, or
ii) n is odd and $E(G)$ can be partitioned into $\lfloor \frac{n}{2} \rfloor$ edge disjoint Hamiltonian cycles and a perfect matching.

Lemma 2.3 (See [1]): For $n \geq 2$, Q_n has a Hamilton decomposition.

Lemma 2.4: For $n \geq 2$, the length of shortest cycle in Q_n is 4.

Next, we introduce four known All-Reduce operations.

A. Halving-Doubling

HD is a widely used algorithm for implementing All-Reduce, and is employed in Alibabas communication library. In this algorithm, nodes communicate with others located at power-of-two distances, while the amount of data exchanged halves (or doubles) in each step. A key advantage of HD is its low number of communication hopsonly $2log_2 N$ hops are required to complete the process, where $N = 2^n$ denotes the number of

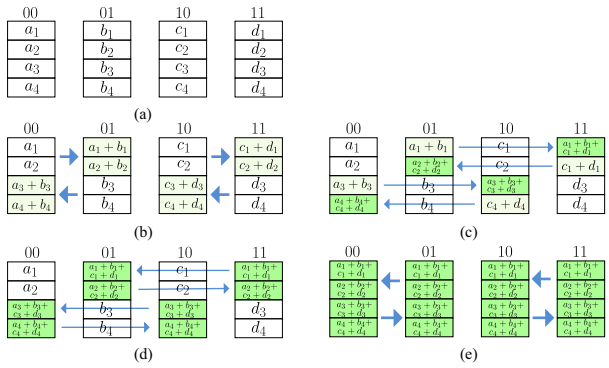

Fig. 1. Illustration of HD in Q_2. (a) Initial state. (b) and (c) Reduce-scatter operation. (d) and (e) All-gather operation.

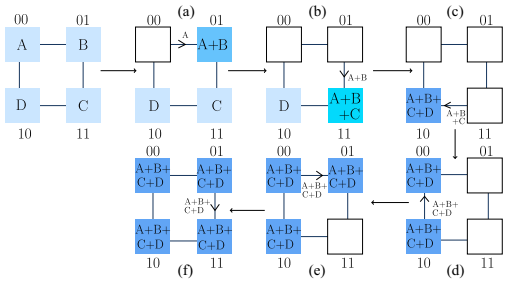

Fig. 2. BR in Q_2. (a)–(c): reduce-and-forward phase; (d)–(f): all-gather phase.

participating nodes thus reducing latency. In comparison, the ring algorithm requires $2(N-1)$ hops. Let D represent the value of data volume per node (for convenience, all subsequent mentions of data volume refer to this numerical value). In HD, the amount of data transmitted per step decreases progressively: $D/2, D/4, D/8, \ldots, D/N$, summing to $D - D/N$. The total data transfer volume is $2D(N-1)$. Fig. 1 illustrates the HD operation on a Q_2. Notably, HD can be viewed as operations performed on rings with two nodes.

B. Basic Ring (BR)

The BR structure in the information transmission process transmission connects the nodes in the network end-to-end, forming a ring. This setup allows for convenient interaction between different modules without the need for central control or routing. Functional units send information onto the ring through network interfaces, and messages circulate through each node on the ring, advancing one node at a time. When a message reaches the node connected to its destination functional unit, it is taken off the ring, directed to the network interface, and then delivered to the target functional unit. Fig. 2 demonstrates the operation of the BR with a 4-cycle. In a network with N nodes and D data per node, the message synchronization for BR operation requires $2(N-1)$ hops, and the total amount of data transmitted is $2D(N-1)$. The data volume transmitted by each node is D.

C. Ring All-Reduce

The [5] algorithm, initially pioneered in the realm of DL by Baidu, presents a novel strategy for synchronizing gradients to alleviate network bottlenecks during synchronous updates. Illustrated in Fig. 1, the RAR algorithm employs a ring topology to interconnect graphics processing units (GPUs), ensuring communication solely among adjacent nodes. This process encompasses two distinct phases: reduce-scatter [see Fig. 3(a)–(c)] and all-gather [see Fig. 3(d)–(f)]. In a distributed system comprising N nodes, the RAR algorithm mandates $2(N-1)$ communication hops [9] to successfully execute the synchronization procedure with a total data transfer of $2D(N-1)$. Because the total data sent by each node in the RAR algorithm is balanced, the total data volume for each node is $2D(N-1)/N$. The amount of data transferred per vertex per hop is $\frac{D}{N}$. The data fragments obtained during the reduce-scatter operation are assigned serial numbers from 1 to N in a clockwise direction.

By Lemma 2.3, Q_n has $\lfloor \frac{n}{2} \rfloor$ edge disjoint Hamiltonian cycles, denoted by $C_N^1, C_N^2, \cdots, C_N^{\lfloor \frac{n}{2} \rfloor}$ with $N = 2^n$. We can perform RAR algorithm on Q_n with $\lfloor \frac{n}{2} \rfloor$ rings simultaneously. For the first step, we divide each vertex information equally into $\lfloor \frac{n}{2} \rfloor$ segments. For the second step, the $\lfloor \frac{n}{2} \rfloor$ information segments from each point in Q_n are independently subjected to RAR algorithms within $\lfloor \frac{n}{2} \rfloor$ edge disjoint Hamiltonian cycles.

D. Hierarchical Ring All-Reduce

In this section, we first introduce a new ring-based communication reduction algorithm in distributed computing, said *HRA algorithm*, which was proposed in [23]. Then, we propose optimizations for this algorithm. Let G be a parallel distributed network. The proceedings of HRA are as follows.

1) *Step one:* Partition the network into h layers, denoted as $L_1, L_2, \ldots, L_h$, such that the following holds.
   a) For each $i \in [h]$, L_i is a set of subgraphs of G with $V(L_i) = V(G)$ and any two elements in L_i are edge disjoint.
   b) Every element of L_i is a cycle and any two elements in L_i have equal lengths.
   c) $E(L_1) \cap E(L_2) \cap \cdots \cap E(L_h) = \emptyset$.

   For $i \in [h]$, let y_i be the number of elements in L_i, ℓ_i be the length of cycles in L_i, and $L_i = \{C_{\ell_i}^{i,j} : 1 \le j \le y_i\}$. Note that the order of layers can be switched around as practical. However, once the position is determined, the operation is executed in that order.
2) *Step two:* The reduce-scatter operation in RAR algorithm is performed in the order of the label of the layers from the smallest (L_1) to the largest (L_h).
   a) *Hierarchical data partitioning:* At each layer L_i, the system evenly partitions the input data (initially of size D, subsequently $D/(\ell_1 \cdots \ell_{i-1})$) into ℓ_i data blocks, where each block has a size of $D/(\ell_1 \cdots \ell_i)$.
   b) *Ring-based reduction communication:* The vertex sequentially sends and receives data blocks on each independent cycle of length ℓ_i in the current layer, performs reduce operations, and ultimately each process retains only one valid reduction block.
   c) *Hierarchical data propagation:* The reduced data blocks (of size $D/(\ell_1 \cdots \ell_i)$) serve as input for the next layer. After processing through all h layers, each vertex obtains its final reduced fragment of size D/N, where $N = \prod_{i=1}^{h} \ell_i$.
3) *Step three:* The all-gather operation in RAR algorithm is performed on the results obtained in Step Two along the

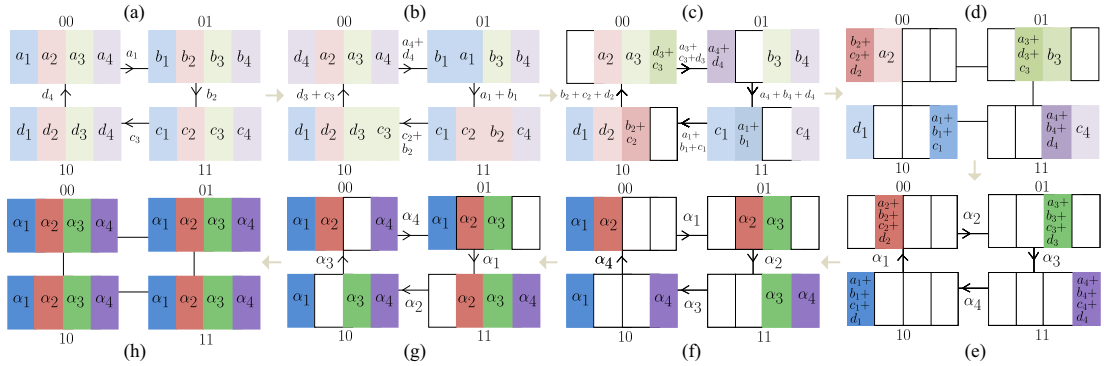

Fig. 3. RAR in Q_2. (a)–(d) Perform reduce-scatter operation. Each processor partitions its data into four blocks. At each step, a processor receives a block from the preceding processor while simultaneously sending one of its own blocks to the next processor.The received block is immediately reduced (e.g., summed) with the corresponding local block. After steps (a) to (d), each processor holds exactly one fully reduced block (e. g. , processor 00 hold $\alpha_2 = a_2 + b_2 + c_2 + d_2$.) (e)–(h) Perform all-gather operation. Processors exchange the partial-result blocks generated during the reduce-scatter phase over the same ring. Each step increases the number of reduced blocks held by every processor by one. After steps (e) to (h), all processors obtain the complete set of globally reduced results $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$.

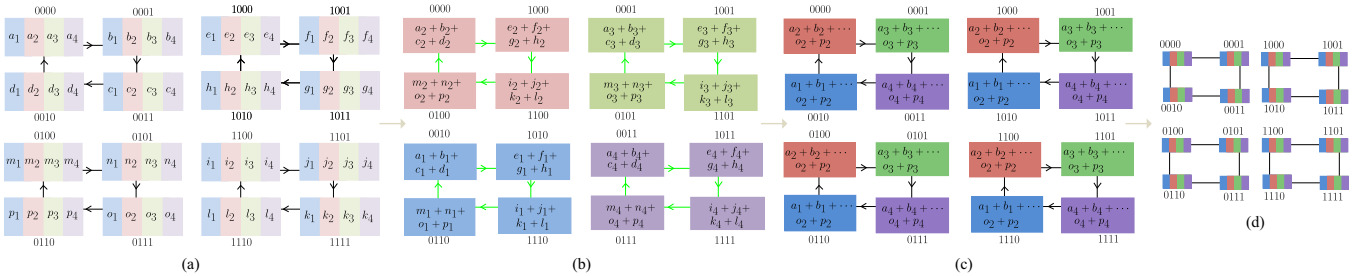

Fig. 4. HRAR in Q_4. (a) Intragroup reduce-scatter (L_1): Four black processor rings execute parallel reduce-scatter; each processor retains one partially reduced block. (b) Intergroup reduction (L_2): Four green rings perform FRAR in parallel: reduce-scatter aggregates blocks across groups; all-gather distributes globally reduced blocks within each ring. (c) Intragroup all-gather (L_1): Black rings synchronize globally reduced data within groups. (d) Final state: All processors hold identical globally reduced data.

reverse of the order of their corresponding layers (from L_h to L_1).

a) *Data exchange:* Each vertex begins with a data block of size $D/(\ell_1 \cdots \ell_i)$. Within its assigned logical ring (length ℓ_i), it executes all-gather by transmitting its current data block and concurrently receiving blocks from peer vertices.

b) *Data aggregation:* Through this collective communication, each process accumulates ℓ_i distinct blocks, which, when merged, form a composite data segment of size $D/(\ell_1 \cdots \ell_{i-1})$.

We use Q_4 as an example to illustrate the HRA algorithm (see Fig. 4). HRA requires $2\sum_{i=1}^{h}(\ell_i - 1)$ communication hops to complete synchronization. The hierarchical decomposition follows a recursive partitioning: at layer L_1, the N vertices are divided into N/ℓ_1 disjoint type-1 groups, each forming a cycle of length ℓ_1. For $k \geq 2$, layer L_{k-1} yields $N/\prod_{i=1}^{k}\ell_i$ type-k groups, each containing $\prod_{i=1}^{k-1}\ell_i$ disjoint cycles of length ℓ_k. This recursion ends at layer L_h, where exactly one type-h group $\left(N/\prod_{i=1}^{h}\ell_i = 1\right)$ contains $\prod_{i=1}^{h-1}\ell_i$ disjoint cycles of length ℓ_h, giving the product relation $N = \prod_{i=1}^{h}\ell_i$. The data transmission volume is derived as follows. In layer i (cycle length ℓ_i), each vertex sends $D/(\ell_1 \cdots \ell_i)$ data per hop over $\ell_i - 1$ hops, resulting in a layer-wise total of $D/(\ell_1 \cdots \ell_i) \cdot (\ell_i - 1)N$. Summing over all layers and doubling for both reduce-scatter and all-gather phases yields the overall data volume: $\left(\sum_{i=1}^{h}\frac{D(\ell_i - 1)}{\prod_{j=1}^{i}\ell_j}\right)N \times 2 = 2D(N-1)$.

III. HRAR in Hypercube

Traditional hierarchical All-Reduce algorithms suffer from low link utilization and severe load imbalance due to their fixed sequential layer processing. At any time, a vertex v communicates only through edges in the active layer, leaving other incident edges idle. To address this imbalance, our algorithm employs concurrent logical paths, allowing the logical hierarchy level of each physical layer to vary across parallel operations. This naturally distributes traffic that would otherwise concentrate in specific layers.

Our core method partitions the data vector into disjoint sub-vectors, each assigned a distinct hierarchical execution sequence. During each communication hop, every vertex thus utilizes a broader set of incident links for parallel transmission, dynamically spreading traffic across links. To quantify the load-balancing effect, we introduce the variance of per-vertex data transmission volume across hops

$$\delta^2 = \frac{1}{H}\sum_{i=1}^{H}(c_i - \mu)^2 \tag{1}$$

where H is the total number of hops, c_i is the data volume transmitted per vertex on active edges at hop i, and

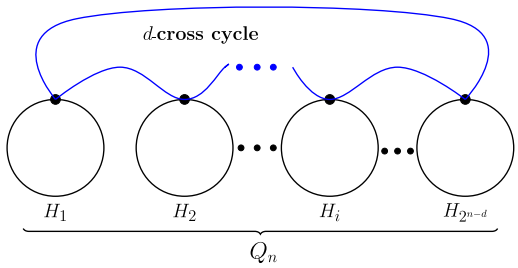

Fig. 5. Blue cycle is a d-cross cycle of Q_n. For each $i \in [2^{n-d}]$, $H_i \cong Q_d$.

$\mu = \frac{1}{H}\sum_{i=1}^{H} c_i$ is the mean. In traditional algorithms, large fluctuations in c_i across hops yield a high variance δ^2. By enhancing link-level concurrency and balancing the active layers, our algorithm maintains more stable c_i values, significantly reducing δ^2. This reduction directly reflects improved load balancing across both temporal (interhop) and spatial (interlink) dimensions.

To enable concurrent execution across all hierarchical layers, the algorithm requires a topology that supports uniform cycle lengths in all partitions. Disparities in cycle lengths (e.g., $L_1 \neq L_2$) would cause faster layers to wait for slower ones, leading to pipeline stalls, increased latency, and degraded throughput. Hence, a topology that permits hierarchical partitioning with uniform ring lengths is essential. To remedy the imbalanced edge utilization in traditional approaches, we propose the OHRA algorithm, built upon the highly symmetric hypercube Q_n. The method naturally extends to other Q_n-like topologies.

This section is organized as follows. First, we define h-hierarchical cycle decomposition. Then, we prove that Q_n admits an h-hierarchical cycle decomposition (see Theorem 3.3). Finally, leveraging the structural properties of Q_n, we propose the OHRA algorithm, which achieves high edge utilization and load balance.

Definition 3.1: For two integers n and h, let G be a n-regular graph containing N vertices. It is said that G has a *h-hierarchical cycle decomposition* if

i) G has h spanning subgraphs $L_1, L_2, \ldots, L_h$ with $\cap_{i=1}^{h} E(L_i) = \emptyset$ for each $i \in [h]$, and
ii) for $i \in [h]$, every component of L_i is isomorphic to C_{ℓ_i}, where ℓ_i is a positive integer with $\prod_{i=1}^{h} \ell_j = N$ and any two elements of L_i are edge disjoint.

Next, we will prove that Q_n has a h-hierarchical cycle decomposition. For $n \geq 4$ and $1 \leq d \leq n-1$, note that Q_n can be decomposed into 2^{n-d} subgraphs, say H_i ($i \in [2^{n-d}]$), where each H_i is isomorphic to Q_d. Let $\mathcal{X} = \{X_i \colon 1 \leq i \leq 2^{n-d}\}$ be a set of $(n-d)$-bit binary string. For $i \in [2^{n-d}]$ and $s \in [d]$, let $V(H_i) = \{X_i u_d u_{d-1} \cdots u_2 u_1 \colon u_s \in \{0,1\}, 1 \leq s \leq d\}$ with $X_i \in \mathcal{X}$. For $u \in V(H_i)$ and $v \in V(H_j)$ with $i, j \in [2^{n-d}]$, $uv \in E(Q_n)$ if and only if the bits from 1 to d of u and v are the same, respectively, and X_i and X_j differ in only one bit. For $j \in [2^d]$ and each $i \in [2^{n-d}]$, denote a cycle C^j as a *d-cross cycle* of Q_n (see Fig. 5) if C^j satisfies that $|V(C^j) \cap V(H_i)| = 1$ and $E(C^j) \cap E(H_i) = \emptyset$.

Lemma 3.2: For each $i \in [2^{n-d}]$, $n \geq 4$ and $2 \leq d \leq n-2$, Q_n has 2^d vertex disjoint d-cross cycles, say C^j $\left(j \in [2^d]\right)$, such that $|V(C^j) \cap V(H_i)| = 1$ and $E(C^j) \cap E(H_i) = \emptyset$.

Proof: For $j \in [2^d]$, let U_j be a vertex set of Q_n such that the value from the first to the d-th position of any two vertices in U_j are equal, respectively. Then, $|U_j \cap V(H_i)| = 1$, $|U_j| = 2^{n-d}$, and $Q_n[U_j]$ is isomorphic to Q_{n-d}. Since $n \geq 4$ and $2 \leq d \leq n-2$, we have $n-d \geq 2$, which implies that $Q_n[U_j]$ contains a Hamiltonian cycle, denoted as C^j. Thus, C^1, $C^2, \ldots, C^{2^d}$ are vertex disjoint. As $U_j = V(C^j)$ for $j \in [2^d]$, we get that $|V(C^j) \cap V(H_i)| = 1$ with $i \in [2^{n-d}]$, which implies that $E(C^j) \cap E(H_i) = \emptyset$. For each $j \in [2^d]$, one has that C^j is a d-cross cycle. The lemma holds.

Theorem 3.3: For positive integers n and h with $n \geq 2$ and $1 \leq h \leq \lfloor \frac{n}{2} \rfloor$, Q_n has h-hierarchical cycle decompositions.

Proof: For $i \in [h]$, let $\ell_i = 2^{t_i}$ be the length of a cycle in Q_n with $\sum_{i=1}^{h} t_i = n$. Since the shortest cycle length of Q_n is 4, one has that $t_i \geq 2$ for each $i \in [h]$. Let $t_0 = 0$ and $G_0 \cong Q_{n-t_0}$. By the definition of Q_n, one has that Q_n contains 2^{n-t_1} vertex disjoint t_1-dimensional hypercubes Q_{t_1}, say $Q_{t_1}^{0,1}$, $Q_{t_1}^{0,2}$, …, $Q_{t_1}^{0,2^{n-t_1}}$, such that $\cup_{j=1}^{2^{n-t_1}} V(Q_{t_1}^{0,j}) = V(Q_n)$. The graph obtained from Q_n by contracting each $Q_{t_1}^{0,j}$ into a vertex and replacing the parallel edges with a single edge is denoted as G_1. Actually, G_1 is isomorphic to Q_{n-t_1}. Similarly, for a given $k \in \{1, \ldots, h-1\}$ and $G_k \cong Q_{n-(\sum_{j=0}^{k} t_j)}$, G_k contains $2^{n-(\sum_{j=0}^{k+1} t_j)}$ vertex disjoint t_{k+1}-dimensional hypercubes $Q_{t_{k+1}}$, say $Q_{t_{k+1}}^{k,1}, Q_{t_{k+1}}^{k,2}, \ldots, Q_{t_{k+1}}^{k,2^{n-\left(\sum_{j=0}^{k+1} t_j\right)}}$, such that $V(G_k) = \cup_{p=1}^{t} V(Q_{t_{k+1}}^{k,p})$ with $t = 2^{n-\left(\sum_{j=0}^{k+1} t_j\right)}$. Then G_{k+1} is obtained from G_k by contracting each $Q_{t_{k+1}}^{k,p}$ to a vertex in G_k and replacing the parallel edges with a single edge. Actually, G_{k+1} is isomorphic to $Q_{n-\left(\sum_{j=0}^{k+1} t_j\right)}$.

Next, we construct h-hierarchical cycle decomposition of Q_n with $1 \leq h \leq \lfloor \frac{n}{2} \rfloor$ by inversing the process of construction of G_k.

For $k = 0$ and $j_0 \in [2^{n-t_1}]$, let $H\left(Q_{t_1}^{0,j_0}\right)$ be a Hamiltonian cycle of $Q_{t_1}^{0,j_0}$. Denote $L_1 = \{H(Q_{t_1}^{0,j_0}) \colon j_0 \in [2^{n-t_1}]\}$. For $k \in \{1, 2, \ldots, h-1\}$ and $j_k \in [2^{n-r}]$ with $r = \sum_{q=0}^{k+1} t_q$, the graph, obtained from $Q_{t_{k+1}}^{k,j_k}$ by blowing up each vertex to a copy of Q_ℓ with $\ell = \sum_{q=0}^{k} t_q$, and replacing each edge $xy \in E(Q_{t_{k+1}}^{k,j_k})$ by one matching with 2^ℓ edges between the two copies of Q_ℓ corresponding to the two ends of the edge xy, is denoted by $\mathrm{B}(Q_{t_{k+1}}^{k,j_k}, Q_\ell)$ such that $\mathrm{B}(Q_{t_{k+1}}^{k,j_k}, Q_\ell)$ is isomorphic to Q_r. Then, there are $2^{t_{k+1}}$ vertex disjoint Q_ℓ's in $\mathrm{B}(Q_{t_{k+1}}^{k,j_k}, Q_\ell)$. By Lemma 3.2, for each $j_k \in [2^{n-r}]$, $\mathrm{B}(Q_{t_{k+1}}^{k,j_k}, Q_\ell)$ contains 2^ℓ vertex disjoint ℓ-cross cycles. Let $H(Q_{t_{k+1}}^{k,j_k})$ be the set of ℓ-cross cycles of $\mathrm{B}(Q_{t_{k+1}}^{k,j_k}, Q_\ell)$. Denote $L_{k+1} = \{H(Q_{t_{k+1}}^{k,j_k}) \colon 1 \leq j_k \leq 2^{n-(\sum_{q=0}^{k+1} t_q)}\}$. For each $i \in [h]$, any two components of L_i are edge disjoint and every component of L_i is a cycle which is isomorphic to C_{ℓ_i} with $\ell_i = 2^{t_i}$. Furthermore, $\cap_{i=1}^{h} E(L_i) = \emptyset$ and $V(L_i) = V(Q_n)$. Thus, $L_1, L_2, \ldots, L_h$ is a h-hierarchical cycle decomposition of Q_n. Fig. 6 illustrates the process using Q_6 as an example.

Taking Q_4 as an example, we perform a 2-layer hierarchical cycle decomposition on Q_4. The first layer L_1 and the second layer L_2 are shown in Fig. 12 in the Appendix, represented by

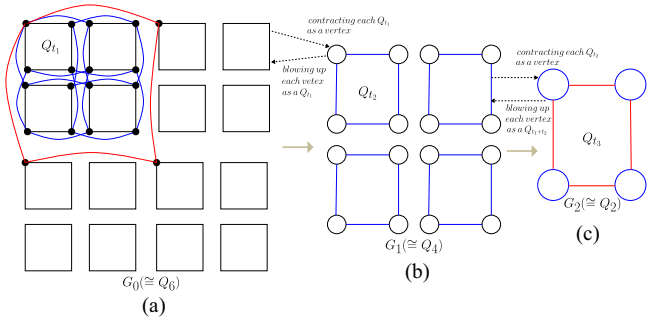

Fig. 6. Auxiliary diagram for the proof of Theorem 3.3. Using Q_6 as an example, it illustrates the specific operations when $h = 3$ and $t_i = 2$ for each $i \in [3]$. (a) Initial graph Q_6 ; (b) Graph Q_4 obtained by contracting each copy of Q_2 in Q_6 into a single vertex; (c) Graph Q_2 obtained by contracting each copy of Q_2 in Q_4 into a single vertex.

the set of cycles formed by the black edges and the green edges, respectively.

The design of our OHRA algorithm is grounded in the structural properties of the hypercube Q_n. By Theorem 3.3, Q_n admits an h-hierarchical cycle decomposition $\{L_1, L_2, \ldots, L_h\}$. For each $i \in [h]$, let ℓ_i denote the length of cycles in layer L_i. Crucially, we can select a decomposition in which the cycle length is identical across all layers, i. e., $\ell_1 = \ell_2 = \cdots = \ell_h$. Moreover, owing to the vertex-transitivity and edge-transitivity of Q_n, each vertex and each edge exhibits identical communication characteristics. This uniform cycle length enables a fully parallel slicing strategy: after partitioning the data at each vertex, the edge-disjoint cycles in all layers can perform ring-based reduce-scatter and all-gather operations concurrently and synchronously, without waiting or stalling. Consequently, at every communication hop, every edge becomes active, eliminating idle links and maximizing edge utilization. Building upon this insight, we integrate the hierarchical framework of HRA with the topological advantages of Q_n to formulate the OHRA algorithm, whose detailed procedure is given in Algorithm 1. To clarify the layer number h, we denote the OHRA algorithm executed on Q_n as the h-OHRA algorithm. As shown in Fig. 12 of Appendix, we perform the 2-OHRA operation using Q_4 as an example. The communication hop is $2 \cdot h \cdot (\ell_i - 1)$ and the total amount of data is $2D(N-1)$.

The h-OHRA algorithm is designed to improve load balancing. As shown in Table II, we compare edge utilization patterns at identical time instants, considering: 1) the number of globally active edges, and 2) the number of concurrently active edges within the neighborhood of any vertex v. Results show that for even n, h-OHRA achieves not only the highest global edge utilization, but also significantly improves the concurrent edge utilization at the vertex level. This verifies the algorithms load-balancing capability at both global and local scales.

IV. Analysis of Optimal Hierarchical Decomposition of Hypercubes Based on Minimum Communication Hops

The number of communication hops directly determines the latency and throughput of gradient synchronization in distributed training, and is a key factor affecting the performance

Algorithm 1: Optimized Hierarchical Ring All-Reduce.

Require: Graph G with h-hierarchical cycle decomposition $\mathcal{L} = \{L_1, L_2, \ldots, L_h\}$; Initial data D_v at each vertex $v \in V(G)$ with volume D.
Ensure: Each vertex v holds the complete globally reduced data fragment.

if $\ell_1 = \ell_2 = \cdots = \ell_h$ **then**
  Phase 1: Hierarchical Reduce-Scatter
  Partition D_v into h equal segments $\{M_1^v, M_2^v, \ldots, M_h^v\}$, $\dim(M_i^v) = D/h$.
  for $i = 1$ to h **do**
    $M_i^{(0)}(v) \leftarrow M_i^v$
    for $k = 1$ to h **do**
      $l \leftarrow \sigma_i(k) = (i + k - 2 \bmod h) + 1$
      Select cycle set $\{C_{\ell_l}^{l,j} : j \in [y_l]\}$ in layer L_l
      Partition $M_i^{(k-1)}(v)$ into ℓ_l sub-blocks $\{B_1^v, B_2^v, \ldots, B_{\ell_l}^v\}$ such that $M_i^{(k-1)}(v) = \sqcup_{s=1}^{\ell_l} B_s^v$ and $\dim(B_s^v) = \frac{D}{h \prod_{t=1}^{k} \ell_{\sigma_i(t)}}$
      Execute Reduce-Scatter on each cycle C in layer L_l:
        $M_i^{(k)}(v) \leftarrow \oplus_{u \in V(C)} \phi_{v,u}(B_{s(u)}^u)$
        where $s(u)$ is the sub-block index of u in C
    end for
  end for
  Phase 2: Hierarchical All-Gather
  for $i = 1$ to h **do**
    for $k = h$ down to 1 **do**
      Execute All-Gather on the corresponding cycle in layer $L_{\sigma_i(k)}$
    end for
  end for
else
  Perform Step Two from Section II-D (Standard HRA)
end if
Each vertex v combines all segments to form the complete result.

TABLE II
Number of Globally Active Edges and the Number of Active Edges Within the Neighborhood of Any Vertex v by Different Algorithms at the Same Moment

Algorithms	Globally Active Edges	Neighborhood of vertex v	Algorithms	Globally Active Edges	Neighborhood of vertex v
HD	$\frac{N}{2}$	1	2D-TAR	N	2
BR	1	1	2D-HRA	K or N, $4 \le \sqrt{K} \le 2^{n-2}$	2
RAR	N	2	HRA	N	2
HAR	K or N $4 \le K \le 2^{n-2}$	2	h-OHRA	N (odd n); $\frac{Nn}{2}$ (even n)	2 (odd n); n (even n)

of the All-Reduce algorithm. The end-to-end time T of All-Reduce is typically composed of host-to-device transfer time T_{H2D}, device-to-host transfer time T_{D2H}, link transmission time T_{Comm}, and computation time T_{Comp}, integrating GPU memory transfer [20], LogGP-based network communication [8], and gradient reduction costs [9]. The details can be expressed in the following form:

$$T_{\text{total}} = T_{\text{H2D}} + T_{\text{D2H}} + T_{\text{Comm}} + T_{\text{Comp}}$$

TABLE III
PARAMETERS IN THE END-TO-END TIME MODEL

Parameters	Description	Parameters	Description
H	Communication hops	i	Index of current hop
d_i	Data volume processed per vertex at i-th hop	B	Uniform link bandwidth in the network
B_{H2D}	Host-to-device memory bandwidth	B_{D2H}	Device-to-host memory bandwidth
Ops_i	Operations per data element at i-th hop	α_i	Computational efficiency factor
β_i	Bandwidth utilization efficiency factor	$\beta_{0,i}$	Fixed latency per hop
CompBW_i	Computational bandwidth at i-th hop		

$$= \sum_{i=1}^{H} \frac{d_i}{B_{\text{H2D}}} + \sum_{i=1}^{H} \frac{d_i}{B_{\text{D2H}}} + \sum_{i=1}^{H} \left(\frac{d_i}{B} \cdot \beta_i + \beta_{0,i} \right) + \sum_{i=1}^{H} \frac{d_i \cdot \text{Ops}_i}{\text{CompBW}_i} \cdot \alpha_i \quad (2)$$

where the parameter meanings are shown in Table III. In this model, we assume that within each communication hop i, the data volume d_i is uniform across all vertices, which reflects the load-balanced nature of collective communication algorithms like All-Reduce. In the latency model of (2), the hop count H directly contributes to T_{Comm}. Under moderate-to-high bandwidth regimes, minimizing H is the primary lever for reducing overall latency. An optimal reduction in the number of communication hops H, achieved through a well-designed hierarchical decomposition, can significantly reduce communication latency by minimizing the accumulation of per-hop overhead, thereby improving overall transmission efficiency. For Q_n, the number of layers and the cycle length play a crucial role in the number of communication hops. This naturally raises the question: What kind of h-hierarchical cycle decomposition can achieve the minimum number of communication hops in Q_n? Theorem 4.1 provides a clear answer to this question. The number of layers with the minimum number of communication hops is defined as the *optimal number of layers*.

Theorem 4.1: For $n \geq 2$, the minimum number of communication hops of h-OHRA algorithm on Q_n is as follows.

1) If n is even, then the optimal number of layers is $\frac{n}{2}$, the length of each cycle in each layers is 4 and the minimum number of communication hops is $3n$.
2) If n is odd, then the optimal number of layers is $\frac{n-1}{2}$, the length of each cycle in only one layer is 8, the length of each cycle in the rest layers is 4, and the minimum number of communication hops is $3n + 5$.

Proof:

1) If n is even, for each $i \in [h]$, let $t_i = 2$, then $\ell_i = 2^{t_i} = 4$. Combining $\prod_{i=1}^{h} \ell_i = N = 2^n$, one has that $h = \frac{n}{2}$. Thus, in $n/2$-hierarchical cycle decomposition, where the length of cycles of every layer is 4, the number of hops of $n/2$-OHRA algorithm is $\sum_{i=1}^{h} 2(\ell_i - 1) = 3n$. Let h' and r be two integer such that $1 \leq r \leq h' \leq h$. Assume that there exists at least one $r \in [h']$ with $t_r \geq 3$, then $h' \leq h - 1$. Let L'_1, L'_2, …, $L'_{h'}$ be the h'-hierarchical cycle decomposition. Thus, the number of communication hops of h'-OHRA algorithm is $\sum_{i=1}^{h'} 2(\ell_i - 1) \geq 2(\sum_{i=1}^{h'} 2^{t_i}) - 2 \cdot (n/2 - 1)$. Since $\sum_{i=1}^{h'} = n$ and $t_i \geq 2$ for each $i \in [h']$, one has that $\sum_{i=1}^{h'} 2(\ell_i - 1) - 3n > 0$. Thus, for even n, the optimal number of layers is $\frac{n}{2}$, the length of cycles in each layer is 4, and the minimum number of communication hops is $3n$.
2) If n is odd, then for $i \in [h], t_1, t_2, \ldots, t_h$ cannot all be even. Let $t_s = 3$ and $t_i = 2$ for each $i \in [h] \setminus \{s\}$. Combining $\prod_{i=1}^{h} \ell_i = N$, one has that $h = (n-1)/2$. Thus, in $(n-1)/2$-hierarchical cycle decomposition where the length of each cycle of the s-th layer is 8 and the length of each cycle of the i-th layers is 4 with each $i \in [h] \setminus \{s\}$. Then, the number of communication hops of $(n-1)/2$-OHRA algorithm is $\sum_{i=1}^{h} 2(\ell_i - 1) = 3n + 5$. Let h'' and s be two integer such that $1 \leq s \leq h'' \leq h$. Assume that there exists at least two $s_1, s_2 \in [h'']$ with $t_{s_1}, t_{s_2} \geq 3$, then $h'' \leq h - 1$. Let L''_1, L''_2, …, $L''_{h''}$ be the h''-hierarchical cycle decomposition. Thus, the number of communication hops of h''-OHRA algorithm is $\sum_{i=1}^{h''} 2(\ell_i - 1) \geq 2(\sum_{i=1}^{h''} 2^{t_i}) - 2((n-1)/2 - 2)$.

Since $t_i \geq 2$ for $i \in [h'']$ and there exists at least two $s_1, s_2 \in [h'']$ with $t_{s_1}, t_{s_2} \geq 3$, one has that $\sum_{i=1}^{h''} 2(\ell_i - 1) - (3n + 5) > 0$ Therefore, when n is odd, the optimal number of layers is $\frac{n-1}{2}$, the length of each cycle in only one layer is 8, the length of each cycle of the rest of layers is 4, and the minimum number of hops is $3n + 5$.

Next, we analyzed several known algorithms (BR, RAR, HD, HRA, and h-OHRA) from the perspective of hop-based load-balancing, while ensuring the minimum total data transmission volume (see Lemma 4.2).

Lemma 4.2: The lower bound on the total data transfer during data parallel computation in the algorithm of ring is $2D(N-1)$.

Proof: Let the total data transfer during data parallel computation in the algorithm of ring be P. In the ring algorithm for data parallel computation with N vertices, each node sends its data to the next vertex in the ring until it reaches back to the original vertex after $N-1$ steps. The data transfer for each vertex involves sending data to one other node in each step. Assuming D is the size of the data being transferred between vertices, the total amount of data transferred by each vertex is $D(N-1)$ since each node communicates with $N-1$ other vertices in the ring. Note that there are N vertices in total, then the total data transfer for all vertices in the ring can be calculated by multiplying the data transfer for each vertex by the total number of vertices. Thus, $P = 2D(N-1)$.

TABLE IV
PARAMETER SETTINGS

Parameter	value	Parameter	value	Parameter	value	Parameter	value	Parameter	value
D	1 GB	B_{H2D}	32 GB/s	B_{D2H}	32GB/s	B	12.5 GB/s	α_i	0.7
Ops_i	1 Flop/byte	CompBW_i	100TFLOPS	β_i	0.8	$\beta_{0,i}$	10 μs		

We conduct a case study with $n = 4$ (parameter configurations are provided in Table IV). The experimental results in Table V illuminate the inherent tradeoffs among hop count, load balance (quantified by δ^2), and total execution time (T_{total}) for the HD, BR, RAR, HRA, and 2-OHRA algorithms on Q_4. While algorithms, such as HD minimize hop count, they often introduce uneven data distribution, leading to suboptimal performance. BR consistently underperforms across all metrics, whereas RAR achieves excellent load balance at the cost of increased hop counts and moderate execution times. In contrast, the proposed 2-OHRA algorithm strikes an effective balance among these competing factors. By leveraging data fragmentation, 2-OHRA

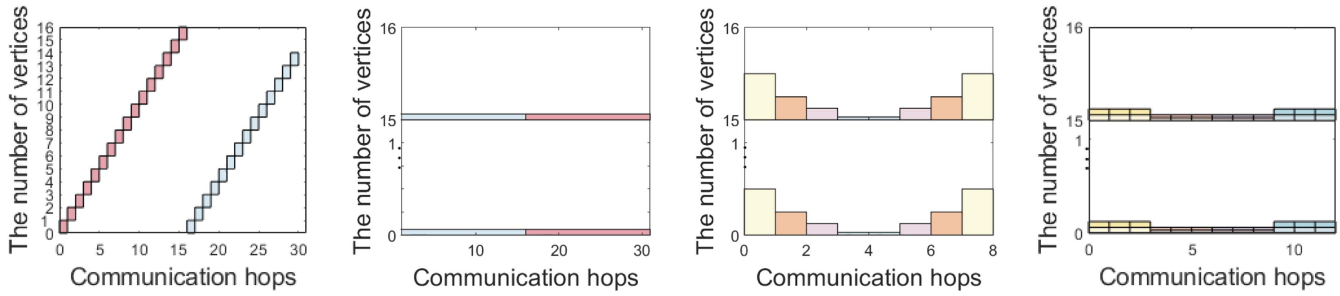

Fig. 7. Data transmission volume per vertex per hop on active edges throughout the entire data transmission process. Each block represents the data transmission volume of the vertex at a specific hop (corresponding to the second column in Table V). Note that (d) represents the HRA performed by Q_4 if colors are not distinguished. If colors are distinguished, it represents the 2-OHRA performed by Q_4, where the different shades of color blocks at each hop represent the amount of data that a vertex divides equally into two parts for parallel transmission. (a) BR in Q_4. (b) RAR in Q_4. (c) HD in Q_4. (d) HRA and 2-OHRA in Q_4.

TABLE V
DATA TRANSMISSION VOLUME PER VERTEX ON ACTIVE EDGES AT EACH HOP, THE VARIANCE IN THE AMOUNT OF DATA TRANSMITTED ACROSS ALL COMMUNICATION STEPS AND THE END-TO-END TIME

Algorithms	Data transmission volume on active edge at each hop	δ^2	$T_{\text{total}}(s)$
HD	$[1,2,3,4,5,6,7,8]=[\frac{D}{2},\frac{D}{4},\frac{D}{8},\frac{D}{16},\frac{D}{16},\frac{D}{8},\frac{D}{4},\frac{D}{2}]$	$\frac{115}{4096}$	0.23727
BR	$[1,2,\ldots,17,\ldots,30]=[D,0,\ldots,0,D,0,\ldots,0]$	$\frac{14}{225}$	3.79551
RAR	$[1,2,\ldots,29,30]=[\frac{D}{16},\frac{D}{16},\ldots,\frac{D}{16},\frac{D}{16}]$	0	0.23749
HRA	$[1,2,3,4,5,6,7,8,9,10,11,12]=[\frac{D}{4},\frac{D}{4},\frac{D}{16},\frac{D}{16},\frac{D}{16},\frac{D}{16},\frac{D}{16},\frac{D}{16},\frac{D}{4},\frac{D}{4},\frac{D}{4},\frac{D}{4}]$	$\frac{9}{1024}$	0.23731
2-OHRA	$[1,2,3,4,5,6,7,8,9,10,11,12]=[\frac{D}{8}\cdot 2,\frac{D}{8}\cdot 2,\frac{D}{8}\cdot 2,\frac{D}{32}\cdot 2,\frac{D}{32}\cdot 2,\frac{D}{32}\cdot 2,\frac{D}{32}\cdot 2,\frac{D}{32}\cdot 2,\frac{D}{8}\cdot 2,\frac{D}{8}\cdot 2,\frac{D}{8}\cdot 2,\frac{D}{8}\cdot 2]$	$\frac{9}{4096}$	0.11873

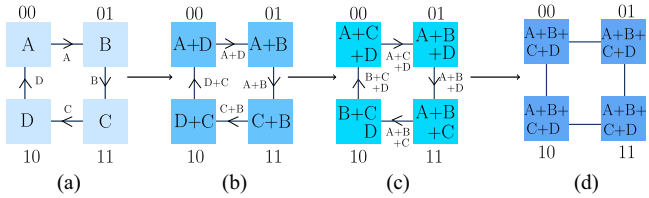

Fig. 8. FRAR in Q_2. Each vertex simultaneously transmits and receives complete data at each hop. (a) Initial data distribution (A at 00, B at 01, C at 11, D at 10) and first-hop transmission directions; (b) After the first simultaneous exchange: nodes accumulate received data (e.g., 00 holds A+D, 01 holds A+B, 10 holds D+C, 11 holds C+B) and prepare for next hop; (c) After the second simultaneous exchange: further accumulation (e.g., 00 holds A+C+D, etc.); (d) After the third simultaneous exchange: all nodes obtain the global sum A+B+C+D.

significantly reduces data transmission variance (δ^2) and total execution time compared to HRA, while maintaining a competitive hop count. Notably, 2-OHRA achieves the smallest δ^2 value among all compared algorithms (except RAR), demonstrating its superior load-balancing capability. This advantage is further corroborated by Fig. 7, which demonstrates that 2-OHRA's fragmented transmission enhances edge utilization and produces more uniform load distribution than HRA. These findings position h-OHRA as an effective approach to All-Reduce optimization, though further investigation is warranted to examine its performance across different network topologies and message sizes.

V. RHRA IN HYPERCUBES

Previous sections discussed nonredundant transmission methods. Here, we introduce a novel algorithm that strategically incorporates redundancy to optimize All-Reduce performance. The approach serves two purposes: improving reliability against packet loss or node failure, and-more critically-reconfiguring communication patterns to reduce hop count and minimize end-to-end latency under bandwidth constraints.

Our design aims to saturate link bandwidth by ensuring the data volume per hop approximates the edge bandwidth B, while minimizing hop count. Crucially, the solution with the fewest hops does not always yield minimal latency, as shown in (2). Redundancy balances increased data volume against substantial hop reduction and improved pipeline efficiency, typically lowering overall latency. In our RHRA algorithm, redundancy acts not only as fault tolerance but as a primary performance parameter, enabling communication strategies that minimize latency even when transmitting above the theoretical minimum.

If $B \geq D$, then we can employ a *full transmission* method in the All-Reduce based on a ring, called FRAR. The core idea is that each node transmits its entire current data buffer in every communication hop, without segmentation as in traditional ring algorithms. As illustrated in Fig. 8 for Q_2, in the initial state each node holds only its local data. In the first hop, all nodes simultaneously send their complete data clockwise to the next node; upon reception, each node performs a reduction operation and updates its local state to the reduced result of the two nodes data. This process repeats for $N-1$ hops. Although the data volume per node remains D at each hop, its content progressively accumulates information from more nodes. After $N-1$ hops, every node obtains the complete globally reduced result. The detailed steps of FRAR are summarized in Algorithm 2.

In this method, the total amount of data transmitted is $DN(N-1)$. The redundancy of data transmission R is typically represented by the following formula:

$$R = \frac{\text{redundant data volume}}{\text{effective data volume}}. \tag{3}$$

Thus, the redundancy of data transmission in this method is $\frac{DN(N-1)-2D(N-1)}{2D(N-1)} = \frac{N-2}{2}$.

Algorithm 2: Full Ring All-Reduce.

Require:
Ring network with N nodes: $Node_0, Node_1, \ldots, Node_{N-1}$
Each node i holds initial data D_i
Ensure: Each node contains the complete reduced data $\bigoplus_{j=0}^{N-1} D_j$
$current\_data \leftarrow D_i$ {Initialize with local data}
$received\_data \leftarrow \emptyset$
for $step = 1$ to $N-1$ **do**
  Transmit $current\_data$ to clockwise neighbor {Simultaneous transmission}
  Receive data from counter-clockwise neighbor $\rightarrow received\_data$
  $current\_data \leftarrow current\_data \oplus received\_data$ {Reduce operation}
  Store $received\_data$ in local buffer at position $(i - step) \bmod N$
end for
{At this point, each node has the complete reduced dataset}
Aggregate all buffered data segments to form final result

Define the RAR algorithm with redundancy as *Redundant RAR*. Based on the previous concept of layering, we design *RHRA algorithm* on Q_n based on the relationship between the bandwidth B of each edge and the amount of data D at each vertex. Let q be the number of layering methods of Q_n. Let h_i be the number of layers of ith layering method with $i \in [q]$ and let $L_{i,r}$ be the set of cycles in the rth layer of ith layering method with $1 \le r \le h_i$. Furthermore, let $\mathcal{L}_i = \{L_{i,r} \colon 1 \le r \le h_i\}$ such that the length of each cycle in $L_{i,r}$ is $\ell_{i,r}$ with $r \in [h_i]$. Let $\mathcal{H} = \{\mathcal{L}_i \colon 1 \le i \le q\}$.

A. RHRA Algorithm With $B \ge D$

When $B \ge D$, a FRAR operation is performed by sequentially transmitting data from each node along the layer numbers, starting from the smallest to the largest, within each layer's ring. Cease the process after all rings in the final layer have completed transmission. At this point, each node holds data information from all other nodes. The total number of hops in this method is $\sum_{r=1}^{h_i}(\ell_{i,r}-1)$ with $i \in [q]$, and the total data volume is $\sum_{r=1}^{h_i}(\ell_{i,r}-1)DN$. By Theorem 4.1, when n is even, the optimal layer of Q_n is $\frac{n}{2}$, with each layer having cycles with length 4, resulting in a minimum of $\frac{3n}{2}$ hops. The redundancy is $\frac{\frac{n}{2}\cdot 3DN - 2D(N-1)}{2D(N-1)} = \frac{3nN}{4(N-1)} - 1$.

For odd n, the optimal layer of Q_n is $\frac{n-1}{2}$, with one layer having a cycle with a length of 8 and the other layers having a cycle with a length of 4, resulting in a minimum of $\frac{3n-1}{2} + 3$ hops. The redundancy is $\frac{3nN+5N}{4(N-1)} - 1$. When $B \ge D$, none of the previously described methods are bandwidth-limited at this time. Table VI shows the relationships between the minimum number of hops for gradient synchronization in nine different algorithms.

TABLE VI
NUMBER OF MINIMUM COMMUNICATION HOPS WITH $B \ge D$

Algorithms	Minimum Communication hops
HD	$2\log_2 N$
RAR and BR	$2(N-1)$
HAR	$\min\{3(\frac{N}{K}-1)+2(K-1) : 4 \le \frac{N}{K}, K \le 2^{n-2}\}$
2D-TAR	$4(\sqrt{N}-1)$
2D-HRA	$\min\{4(\sqrt{K}-1)+3(\frac{N}{K}-1) : 4 \le \frac{N}{K}, \sqrt{K} \le 2^{n-2}\}$
FRAR	$N-1$
HRA	$\min\{2\sum_{r=1}^{h_i}(\ell_{i,r}-1) : i \in [q]\}$
RHRA	$\min\{\sum_{r=1}^{h_i}(\ell_{i,r}-1) : i \in [q]\}$

Algorithm 3: RHRA for Moderate Bandwidth ($D/4 \le B < D$).

Require:
Hypercube Q_n with h-hierarchical decomposition $\mathcal{L}_i = \{L_{i,1}, \ldots, L_{i,h_i}\}$
Each vertex v has data D_v with volume D
Bandwidth constraint: $D/4 \le B < D$
First layer $L_{i,1}$ has cycle length 4
Ensure: Each vertex obtains complete globally reduced data
Phase 1: First Layer Reduce-Scatter
Execute RING-ALL-REDUCE-REDUCE-SCATTER on $L_{i,1}$
▷ Each vertex retains one fragment of size $D/4$
Phase 2: Inter-layer Full Transmission
for $r = 2$ to h_i **do**
  Execute FULL-RING-ALL-REDUCE on layer $L_{i,r}$
  ▷ Propagates reduced data through higher layers
end for
Phase 3: First Layer All-Gather
Execute RING-ALL-REDUCE-ALL-GATHER on $L_{i,1}$
▷ Final synchronization within first layer cycles
return Complete globally reduced data at all vertices

B. RHRA Algorithm With $\frac{D}{4} \le B < D$

When $\frac{D}{4} \le B < D$, to ensure that the amount of data sent by each node is less than or equal to B, we propose Algorithm 3 to facilitate smooth data transmission in this section.

The total number of hops in this process is $2(\ell_{i,1}-1) + \sum_{r=2}^{h_i}(\ell_{i,r}-1)$, and the total data volume is $DN/4 \cdot \sum_{r=1}^{h_i}(\ell_{i,r}-1) + 3DN/4$.

When n is even, Theorem 4.1 shows the optimal layer count is $n/2$, with each layer containing cycles of length 4. This yields a minimum of $3n/2+3$ hops and total data $(n/2+1)\cdot 3DN/4$, giving redundancy $(3n+6)N/[16(N-1)] - 1$. For odd n, the optimal layer count is $(n-1)/2$, with one layer of cycle length 8 and the rest of length 4. This gives $(3n-1)/2+6$ hops and total data $(3n+11)DN/8$, with redundancy $(3n+11)N/[16(N-1)] - 1$.

Under moderate bandwidth $\frac{D}{4} \le B < D$, data transmission is bandwidth-limited. By Lemma 2.4, ring-based All-Reduce on Q_n requires data to be divided into at least 4 segments. Combining this with the bandwidth range yields the minimum hop counts for six algorithms, listed in Table VII. Note that the final step of HAR and 2D-HRA is changed from a broadcast to a RAR operation; otherwise a vertex would transmit D data, violating the bandwidth constraint. With this adjustment, each step transmits DK/N data, staying within the bandwidth limit. Hence, the hop counts become $4(N/K-1)+2(K-1)$ for HAR (with

TABLE VII
MINIMUM NUMBER OF COMMUNICATION HOPS BY DIFFERENT ALGORITHMS WITH $\frac{D}{4} \le B < D$

Algorithms	Minimum Communication hops
RAR	$2(N-1)$
HAR	$\min\{4(\frac{N}{K}-1)+2(K-1) : 4 \le \frac{N}{K}, K \le 2^{n-2}\}$
2D-TAR	$4(\sqrt{N}-1)$ with $\sqrt{N} \ge 4$
2D-HRA	$\min\{4(\sqrt{K}-1)+4(\frac{N}{K}-1) : 4 \le \frac{N}{K}, \sqrt{K} \le 2^{n-2}\}$
HRA	$\min\{2\sum_{r=1}^{h_i}(\ell_{i,r}-1) : \ell_{i,1} \ge 4, \in [h_i], i \in [q]\}$
RHRA	$\min\{(\ell_{i,1}-1)+\sum_{r=1}^{h_i}(\ell_{i,r}-1) : \ell_{i,1}=4, i \in [a]\}$

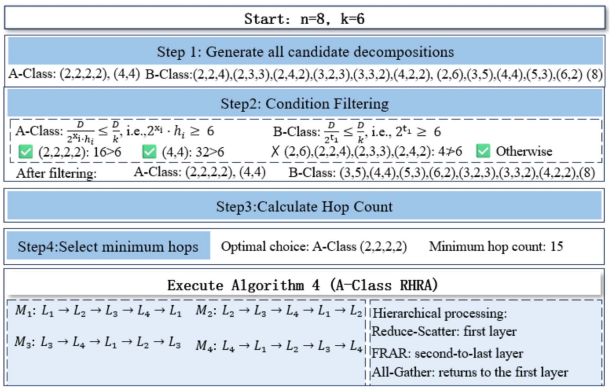

Fig. 9. Example diagrams for Algorithms 4 and 5, taking Q_8 with $k = 6$ as an example.

$4 \le N/K, K \le 2^{n-2}$) and $4(\sqrt{K}-1)+4(N/K-1)$ for 2D-HRA (with $4 \le N/K$, $\sqrt{K} \le 2^{n-2}$).

C. RHRA Algorithm With $\frac{2D}{Nn} \le B < \frac{D}{4}$

For the low bandwidth regime $\frac{2D}{Nn} \le B < \frac{D}{4}$, we consider two types of integer factorizations based on the parity of n. Let $\mathcal{T} = \{T_1, T_2, \ldots, T_q\}$ be a set of positive integer factorizations of n such that each factorization in $\mathcal{T}$ has the smallest positive integer as 2 and the largest positive integer as $n-2$. For $p \in [q]$, let h_p be the number of integers in T_p and $T_p = [t_{p,1}, t_{p,2}, \ldots, t_{p,h_p}]$. Define the integer factorizations of n satisfying $t_{p,1} = t_{p,2} = \cdots = t_{p,h_p}$ as an *A-class factorization*. Otherwise, define it as a *B-class factorization*. Correspondingly, we propose two RHRA algorithms (see Algorithms 4 and 5) tailored for these factorization classes.

For $i \in [s]$ and $2^{x_i} \cdot h_i \ge k$, let $h_{1,i} = (2^{x_i}-1)\cdot(h_i+1)$. For $j \in [q] \setminus [s]$ and $2^{t_{j,1}} \ge k$, let $h_{2,j} = \sum_{r=1}^{h_j}(2^{t_{j,r}}-1) + (2^{t_{j,1}}-1)$. From above, one has that the minimum communication hops is $\min\{H_1, H_2\}$, where $H_1 = \{h_{1,i} : i \in [s], 2^{x_i} \cdot h_i \ge k\}$ and $H_2 = \{h_{2,j} : j \in [q] \setminus [s], 2^{t_{j,1}} \ge k\}$. To clarify the operation of Algorithms 4 and 5, a small worked example is provided in Fig. 9.

When $\frac{2D}{nN} \le B < \frac{D}{4}$, data transfer is greatly limited by bandwidth. Combining the range of B, the relationships between the number of hops for gradient synchronization for the six algorithms are obtained, as shown in Table VIII. The reason for the condition $k \le \frac{N}{K}, K \le 2^{n-2}$ in the Table VIII is as follows. In the HAR and 2D-HRA algorithms, the first step involves partitioning the vertices into K groups, each one containing $\frac{N}{K}$ vertices. Due to bandwidth limitations, the $\frac{N}{K}$ vertices must perform a RAR operation in the second step. Combining with the bandwidth $B = \frac{D}{k}$, the length of the cycle derived by the $\frac{N}{K}$ vertices in each group in Q_n is at least k in order to allow for smooth transmission of data. Because K groups have to do RAR operation between them as well. This requires that $K \ge k$. Furthermore, because the bandwidth $B = \frac{D}{k}$, HAR and 2D-HRA should use RAR operations in the last step in order to guarantee the data transmission. Thus, the number of hops should be $4(\frac{N}{K}-1)+2(K-1)$ for HAR where $k \le \frac{N}{K}, K \le 2^{n-2}$, and $4(\sqrt{K}-1)+4(\frac{N}{K}-1)$ for 2D-HRA where $k \le \frac{N}{K}, \sqrt{K} \le 2^{n-2}$.

Algorithm 4: A-Class RHRA for Low Bandwidth.

Require:
Hypercube Q_n with A-class layering $\mathcal{L}_i$
Each layer has uniform cycle length 2^{x_i}, $x_i \cdot h_i = n$
Data volume D per vertex, bandwidth $B = D/k$, $k \ge 5$
Constraint: $2^{x_i} \cdot h_i \ge k$
Ensure: Each vertex obtains complete globally reduced data
Partition each vertex's data into h_i equal segments: $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_{h_i}$
for $j = 1$ to h_i **do**
  Define cyclic permutation $\delta_j(y) = (j+y-2) \bmod h_i + 1$ ▷ y is layer step index
  Reduce-Scatter at first layer ($y = 1$)**:**
  Execute Reduce-Scatter on $\mathcal{M}_j$ at layer $L_{\delta_j(1)}$
  Update: $\mathcal{M}_j^{(1)} \leftarrow$ result after Reduce-Scatter
  FRAR propagation through higher layers:
  for $y = 2$ to h_i **do**
    Execute Full-Ring-All-Reduce on $\mathcal{M}_j^{(y-1)}$ at layer $L_{\delta_j(y)}$
    Update: $\mathcal{M}_j^{(y)} \leftarrow$ result after layer y
  end for
  All-Gather at first layer:
  Execute All-Gather on $\mathcal{M}_j^{(h_i)}$ at layer $L_{\delta_j(1)}$
  $\mathcal{M}_j^{\text{final}} \leftarrow$ complete reduced segment
end for
Result Assembly:
Each vertex combines all $\mathcal{M}_j^{\text{final}}$ segments into final result

Performance:
Hops: $(2^{x_i}-1)\cdot(h_i+1)$
Total data: $\frac{DN\cdot(2^{x_i}-1)}{2^{x_i}} \cdot (h_i+1)$
Redundancy: $\frac{N(2^{x_i}-1)\cdot(h_i+1)}{2^{x_i}\cdot 2(N-1)} - 1$

The proposed RHRA algorithm fundamentally redefines the role of redundancy in All-Reduce communication. Unlike traditional hierarchical approaches (e.g., RAR, HRA, and 2D-HRA) that strictly minimize data volume through sequential layer execution—often at the cost of poor link utilization—RHRA intentionally introduces controlled redundancy to enable concurrent, pipelined execution across all hierarchical layers. This transforms redundancy from a mere reliability mechanism (as in fault-tolerant systems) into a first-class performance optimization parameter: each unit of redundant data strategically trades for hop-count reduction, directly targeting end-to-end latency minimization rather than just data volume minimization.

Algorithm 5: B-Class RHRA for Low Bandwidth.

Require:
Hypercube Q_n with B-class layering $\mathcal{L}_j = \{L_{j,1}, L_{j,2}, \ldots, L_{j,h_j}\}$
Cycle lengths: $\ell_{j,r} = 2^{t_{j,r}}$ for $r = 1, \ldots, h_j$
First layer length: $\ell_{j,1} \geq k$
Data volume D per vertex, bandwidth $B = D/k$, $k \geq 5$
Ensure: Each vertex obtains complete globally reduced data
Initial Data Partitioning:
Partition each vertex's data into $\ell_{j,1}$ equal segments
Reduce-Scatter at first layer:
Execute Reduce-Scatter on first layer $L_{j,1}$
After reduction, each vertex holds partial results
FRAR propagation through higher layers:
for $r = 2$ to h_j **do**
  Execute Full-Ring-All-Reduce on layer $L_{j,r}$
end for
All-Gather at first layer:
Execute All-Gather on first layer $L_{j,1}$
Final Result Assembly:
Each vertex assembles complete globally reduced data from all blocks

Performance Analysis:
Hops: $H = \sum_{r=1}^{h_j}(\ell_{j,r} - 1) + (\ell_{j,1} - 1)$
Total data: $T_{\text{data}} = \frac{DN}{\ell_{j,1}} \cdot \sum_{r=1}^{h_j}(\ell_{j,r} - 1) + \frac{DN}{\ell_{j,1}} \cdot (\ell_{j,1} - 1)$
Redundancy: $R = \frac{N(\sum_{r=1}^{h_j}(\ell_{j,r}-1)+(\ell_{j,1}-1))}{\ell_{j,1}\cdot 2(N-1)} - 1$

TABLE VIII
MINIMUM NUMBER OF COMMUNICATION HOPS BY DIFFERENT ALGORITHMS WITH $\frac{2D}{Nn} \leq B < \frac{D}{4}$

Algorithms	Minimum Communication hops
RAR	$2(N-1)$
HAR	$\min\{4(\frac{N}{K}-1)+2(K-1) : k \leq \frac{N}{K}, K \leq 2^{n-2}\}$
2D-TAR	$4(\sqrt{N}-1)$ with $\sqrt{N} \geq k$
2D-HRA	$\min\{4(\sqrt{K}-1)+4(\frac{N}{K}-1) : k \leq \frac{N}{K}, \sqrt{K} \leq 2^{n-2}\}$
HRA	$\min\{\{(2^{x_i}-1)\cdot h_i \cdot 2 : 2^{x_i} \cdot h_i \geq k, i \in [s]\},$ $\{\sum_{r=1}^{h_j}(2^{t_{j,r}}-1)\cdot 2 : 2^{t_{j,1}} \geq k, j \in [q] \setminus [s]\}\}$
RHRA	$\min\{\{(2^{x_i}-1)\cdot (h_i+1) : i \in [s], h_i \cdot 2^{x_i} \geq k\},$ $\{\sum_{r=1}^{h_j}(2^{t_{j,r}}-1)+(2^{t_{j,1}}-1) : j \in [q] \setminus [s], 2^{t_{j,1}} \geq k\}\}$

Crucially, this paradigm shift is uniquely enabled by Q_n's edge-disjoint cycle structure, which provides the contention-free parallel pathways necessary for redundant data to flow concurrently without degradation—a topological advantage not available in other networks.

VI. Experimental Results and Discussion

A. Hop Count Experiment

In this section, we evaluate eight All-Reduce algorithms on Q_n ($n = 4$–8) through simulation experiments, examining the relationship between minimum gradient synchronization hops and vertex count under varying bandwidth constraints.

1) High Bandwidth ($B \geq D$): When bandwidth imposes no transmission limits, RHRA achieves the lowest synchronization hops [see Fig. 10(a)], showing a 25% reduction compared to HD. The minimum hops among compared algorithms are $\frac{3n}{2}$ for even n (respectively, $\frac{3n-1}{2} + 3$ for odd n).

2) Moderate Bandwidth ($\frac{D}{4} \leq B < D$): Under moderate bandwidth conditions ($\frac{D}{2} \leq B < D$), the HD algorithm becomes preferable due to its fixed $2\log_2 N$ hop count; specifically, HD is recommended when $n \leq 6$ for even n or $n \leq 11$ for odd n, while RHRA maintains advantages in larger-scale hypercubes [see Fig. 10(b)].

3) Low Bandwidth Regime ($\frac{2D}{Nn} \leq B < \frac{D}{4}$): Under severe bandwidth constraints, ring-based All-Reduce on Q_n allows data segmentation into $\lfloor \frac{n}{2} \rfloor \cdot N$ parts, where $\lfloor \frac{n}{2} \rfloor$ is the maximum number of edge disjoint Hamiltonian cycles. The ring transmission becomes infeasible when $B < \frac{2D}{Nn}$ (even n) or $B < \frac{2D}{N(n-1)}$ (odd n), establishing $\frac{2D}{Nn}$ as our lower bandwidth bound.

Fig. 10(c) and (d) compare six algorithms for $k = n + 1$ and $k = 2^{n-2}$ cases (excluding 2D-TAR when $k = 2^{n-2}$ due to its $\sqrt{N}$ violating Table VIII criteria). RHRA consistently achieves the minimum synchronization hops in both scenarios.

RHRA demonstrates superior hop efficiency across all bandwidth conditions, particularly under large bandwidth, where it significantly accelerates gradient synchronization. The algorithm selection framework provides clear guidelines based on network parameters (n, B).

B. End-to-End Time Experiment

Building upon the hop-count experiment conclusions, we conducted experiments on Q_6 to further evaluate the performance of different algorithms in end-to-end transmission scenarios. In real-world network environments, link bandwidth exhibits significant time-varying characteristics due to factors, such as background traffic, hardware performance fluctuations, and competition, for shared resources. To accurately capture this behavior, this article models the bandwidth B_j of each link j using a truncated normal distribution: $B_j \sim \mathcal{TN}(\mu, \delta^2, B_{\min}, B_{\max})$, where μ represents the nominal (theoretical) bandwidth, δ denotes the standard deviation, which is empirically set within the range $[0.1\mu, 0.2\mu]$. This variability range is motivated by production measurements showing that aggregation link utilization in Facebook data centers fluctuates between 10%–20% median values, with top 5% links reaching 23%–46% utilization [22]. The chosen δ thus reflects the proportional bandwidth uncertainty observed in real-world data center networks. Denote $B_{\min}$ and $B_{\max}$ as the physical lower and upper bounds of the link capacity, respectively. In the network G, each link is assigned a unique identifier ranging from 1 to $|E(G)|$. Let E_i be the set of links traversed during the ith hop of the All-Reduce operation and d_i be the amount of data transmitted in ith hop. Then, $T_{\text{Comm}} = \sum_{i=1}^{H}(\max\{\frac{d_i}{B_j} : j \in E_i\} \cdot \beta_i + \beta_{0,i})$. To model the All-Reduce operation, we leverage the performance analysis framework from the NVIDIA NCCL whitepaper [20]. Based on (2), the condition for the redundant algorithm RHRA to outperform a non-redundant baseline algorithm (e.g., RAR or HRA) can be simplified to

$$\beta_0 \cdot \Delta H > \frac{\beta}{B} \cdot \Delta D. \tag{4}$$

Here, $\Delta H = H_{\text{base}} - H_{\text{RHRA}} > 0$ denotes the reduction in the number of communication hops, $\Delta D = D_{\text{total}}^{(\text{RHRA})} - D_{\text{total}}^{(\text{base})} > 0$

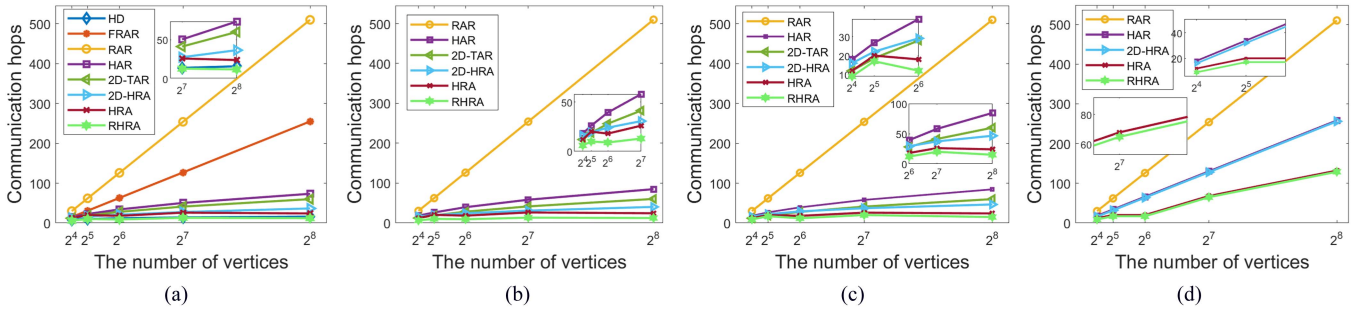

Fig. 10. RHRA shows the lowest communication hops in comparative experiments, where values represent each algorithm's minimum hop count. (a) $B \geq D$. (b) $\frac{D}{4} \leq B < D$. (c) $\frac{2D}{N_n} \leq B < \frac{D}{4}, k = n+1$. (d) $\frac{2D}{N_n} \leq B < \frac{D}{4}, k = 2^{n-2}$.

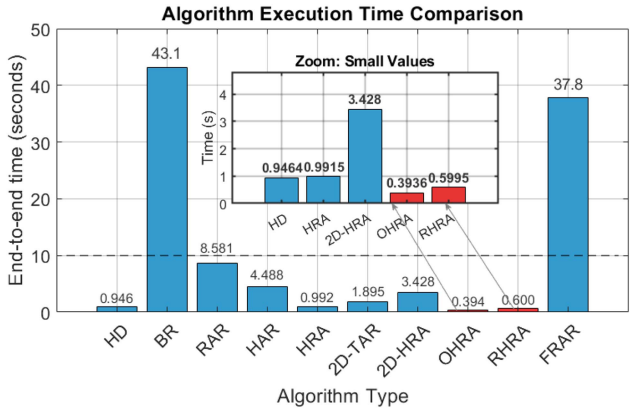

Fig. 11. Experimental results of end-to-end time measurements on Q_6 for 10 algorithms.

denotes the increase in total transmitted data due to redundancy, and β_0 denotes the per-hop constant latency. This inequality captures the essential tradeoff: the strategy of increasing data volume is beneficial only when the fixed-overhead saving from hop reduction ($\beta_0 \Delta H$) outweighs the extra transmission time caused by the redundant data ($\frac{\beta}{B}\Delta D$). In the low-bandwidth regime ($2D/(Nn) \leq B < D/4$), RHRA's hierarchical pipelined design simultaneously achieves a substantial hop reduction (large ΔH), and due to its efficient overlapping of transmissions, the data increment ΔD caused by redundancy is relatively small, making it easy for $\beta_0 \Delta H$ to exceed $\frac{\beta}{B}\Delta D$. Consequently, (4) is automatically satisfied, which explains the pronounced performance advantage of RHRA under severe bandwidth constraints [see Fig. 10(c) and (d)]. For moderate and high bandwidth, RHRA still maintains a hop-count advantage [see Fig. 10(a) and (b)], but whether (4) holds depends on the specific values of β_0, β, and B. To validate this theoretical analysis, we conducted experiments on Q_6 with specific parameters summarized in Table IV. As illustrated in Fig. 11, the end-to-end time evaluation reveals that our proposed OHRA and RHRA algorithms achieve the best performance, completing in only 0.39 and 0.60 s, respectively. Critically, although RHRA introduces redundant data transmission, it significantly outperforms traditional algorithms. These results confirm that a well-designed redundant strategy can effectively reduce communication latency by optimizing transmission paths, especially in bandwidth-constrained scenarios. Moreover, in high-bandwidth settings, the optimized nonredundant variant OHRA (0.39 s) slightly outperforms RHRA (0.60 s), indicating that when bandwidth is abundant, avoiding redundancy can be preferable. Collectively, these findings demonstrate that RHRA provides an effective, redundancy-controlled solution that significantly reduces hop count and thereby lowers latency in bandwidth-constrained environments, offering a valuable communication-optimization strategy for distributed training under such conditions.

C. Reliability Experiment

Redundancy is introduced primarily to mitigate packet loss and improve transmission reliability. To systematically validate this mechanism, we conduct reliability experiments examining how data transmission success varies with redundancy levels. Extensive experiments on Q_{14} (16 384 servers) comprehensively evaluate the reliability characteristics of RHRA. We focus on the relationship between redundancy levels and transmission failure rates, considering practical network conditions where both link failures (E_r) and data corruption (D_r) may occur concurrently. Each server holds 100 GB of data, with $P(E_r) = 10^{-12}$ and $P(D_r) = 10^{-20}$.

The system failure probability P_{fail} integrates two independent failure sources: link errors over Enum edges and data errors over Dvolum data units. The model $P_{\text{fail}} = 1 - (1 - E_r)^{\text{Enum}} \times (1 - D_r)^{\text{Dvolum}}$, combines series-system reliability for communication paths with probabilistic data integrity. By tuning the bandwidth parameter $B = D/k$, we systematically adjust redundancy level k, which simultaneously affects per-edge data volume and the overall transmission topology.

Table IX shows a nonmonotonic yet overall beneficial relationship between redundancy level r and failure rate P_{fail}. We define the *highest failure configuration* (at $k \in [257, 512]$, $r = 7.874 \times 10^{-3}$, $P_{\text{fail}} = 1.691 \times 10^{-5}$) as the baseline. The results reveal three distinct operational regimes. In the low-to-moderate redundancy regime ($1.831 \times 10^{-4} \leq r < 0.219$), the failure rate drops by an average of 52.59% per adjustment, achieving up to 99.05% improvement over the baseline; redundancy and failure rate exhibit the expected inverse correlation. The transitional regime ($0.219 \leq r < 0.488$) shows an anomalous 89.76% rise in failure rate, though it remains 90.7% below the global maximum. This nonmonotonicity occurs because the reduction in total edges (Enum) does not offset the growth in total data volume (Dvolum). In the high-redundancy regime

TABLE IX
REDUNDANCY IN Q_{14} WITH DIFFERENT BANDWIDTHS

k	Minimum Hops	Redundancy r	Enum	Dvolum	Failure Rate	Experimental Mean Value of Failure Rate
1	21	9.500	3.440×10^{5}	3.694×10^{16}	3.441×10^{-7}	3.444×10^{-7}
[2,28]	24	2.000	2.753×10^{6}	1.056×10^{16}	2.752×10^{-6}	2.756×10^{-6}
[29,32]	78	0.219	1.278×10^{6}	4.288×10^{15}	1.278×10^{-6}	1.280×10^{-6}
[33,64]	138	0.078	2.261×10^{6}	3.793×10^{15}	2.261×10^{-6}	2.264×10^{-6}
[65,128]	267	0.043	4.375×10^{6}	3.669×10^{15}	4.374×10^{-6}	4.380×10^{-6}
[129,256]	381	0.488	1.249×10^{7}	5.236×10^{15}	1.248×10^{-5}	1.250×10^{-6}
[257,512]	1032	7.874×10^{-3}	1.691×10^{7}	3.545×10^{15}	1.691×10^{-5}	1.693×10^{-6}
[513, 1024]	2052	2.014×10^{-3}	3.362×10^{7}	3.525×10^{15}	3.362×10^{-5}	3.660×10^{-5}
[1025,2048]	4101	1.282×10^{-3}	6.719×10^{7}	3.523×10^{15}	6.719×10^{-5}	6.727×10^{-5}
[2049,4096]	8193	1.831×10^{-4}	1.342×10^{8}	3.519×10^{15}	1.342×10^{-4}	1.344×10^{-4}

($0.488 \leq r \leq 9.500$), the failure rate declines by an average of 82.73% per adjustment, reaching maximum reliability with a 99.74% reduction from the peak; redundancy again shows a strong negative correlation with failure rate.

These observations stem from a fundamental tradeoff: for a redundancy increment Δr, when $(1 - E_r)^{\Delta E} < (1 - D_r)^{\Delta D}$, the reliability gain from fewer hops outweighs the harm of increased data volume. Monte Carlo simulations (5000 samples, noise intensity 0.05) confirm the models accuracy, showing close agreement between simulated and calculated failure rates (see Table IX).

RHRA optimizes the reliability-redundancy trade-off via bandwidth adaptation ($B = \frac{D}{k}$), achieving three core goals: minimal hops through path optimization, data integrity via controlled redundancy, and no overprovisioning via adaptive resource use. This makes it effective for large-scale distributed systems needing efficiency and reliability, with experiments showing over 99% reliability in optimal configurations and lower communication overhead than conventional methods.

With its bandwidth-aware redundancy adjustment, RHRA systematically balances redundancy, reliability, and hops—it adapts strategies to real network conditions, ensuring data accuracy and avoiding resource waste, thus enabling efficient, reliable transmission across scenarios and providing a robust framework for large-scale distributed learning.

VII. DISCUSSION AND CONCLUSION

A. Conclusion

The proposed RHRA algorithm has demonstrated significant improvements in optimizing communication hops for gradient synchronization in high-dimensional hypercube networks. This optimization has led to enhanced link utilization, making RHRA a promising approach for distributed ML tasks. The algorithms robust performance across various experiments underscores its potential for practical applications in large-scale distributed systems.

B. Limitations and Future Work

First, RHRA's strong dependence on Q_n's edge-disjoint Hamiltonian cycle structure limits its direct applicability to networks that violate this symmetry, including irregular datacenter networks (e.g., Facebook's fabric), bandwidth-asymmetric interconnects (e.g., hybrid NVLink+InfiniBand), and nonhypercube-derived regular topologies (e.g., Torus). Future work should explore adaptation mechanisms that preserve RHRA's concurrent-pipelining principle while relaxing its topological assumptions, such as generalized hierarchical decomposition for maximal edge-disjoint cycle sets in arbitrary regular graphs, virtual topology embedding with bounded congestion guarantees, and degree-aware adaptive layering that adjusts h and ℓ_i based on local connectivity.

Second, the implementation complexity of RHRA may impose a burden on system resources. Simplifying the implementation process and reducing computational overhead are crucial for practical deployment. Finally, there remains room for further optimization in the end-to-end time of the RHRA algorithm. Future work could conduct simulation experiments to establish data ranges and analyze under which conditions RHRA outperforms other algorithms in terms of end-to-end latency. In addition, mechanisms could be introduced to allow RHRA to dynamically adjust its operations in response to changing network conditions, thereby adapting to various topologies and load scenarios.

Based on the above analysis, we recommend that future algorithm designs strike a balance among link utilization, transmission hops, data volume, transmission time, and reliability to further enhance performance and practicality.

DECLARATION OF COMPETING INTEREST

The authors declare that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

APPENDIX

2-OHRA Operation in Q_4

Assuming that the 16 vertices of Q_4 carry data segments named $A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P$.

Step 1: Divide Q_4 into two layers such that the length of cycles in each layer is 4. Let

$$L_1 = \{[0000, 0001, 0011, 0010], [0100, 0101, 0111, 0110], [1000, 1001, 1011, 1010], [1100, 1101, 1111, 1110]\}$$

$$L_2 = \{[0000, 0100, 1100, 1000], [0001, 0101, 1101, 1001], [0010, 0110, 1110, 1010], [0011, 0111, 1111, 1011]\}.$$

Step 2: Divide each vertex's data segment into two parts, named $A_1, A_2, B_1, B_2, \ldots, P_1, P_2$.

Step 2.1: Perform reduce-scatter operation in RAR individually on the data segments@@ with index 1 in the cycles of L_1 [see Fig. 12(a)]. Divide each data segment with index 1 into the following parts: see Table X.

Let the resulting data segments are as follows: see Table XI.

Step 2.2: Perform reduce-scatter operation in RAR individually on the data segments with index 2 in the cycles of L_1 [see Fig. 12(d)]. Divide each data segment with index 2 into the following parts: see Table XII.

Let the resulting data segments are as follows: see Table XIII.

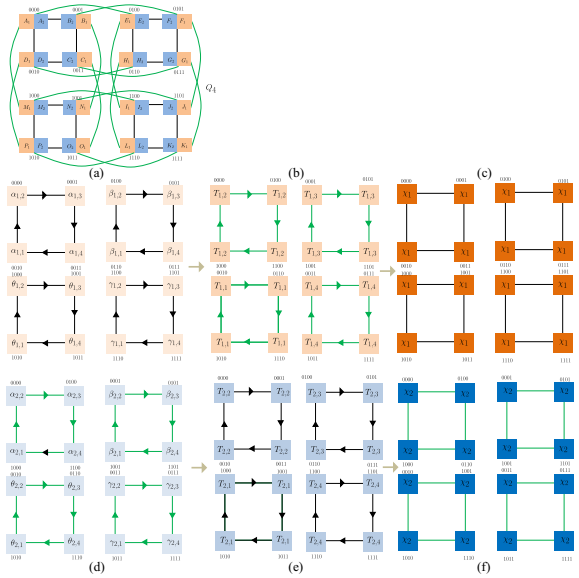

Fig. 12. 2-OHRA in Q_4. (a)–(c): The first half of the data segments perform the HRA algorithm on Q_4. (d)–(f): The second half of the data segments perform the HRA algorithm on Q_4.

TABLE X
STEP 2.1

$A_1 = \{a_{1,1}, a_{1,2}, a_{1,3}, a_{1,4}\}$	$B_1 = \{b_{1,1}, b_{1,2}, b_{1,3}, b_{1,4}\}$
$C_1 = \{c_{1,1}, c_{1,2}, c_{1,3}, c_{1,4}\}$	$D_1 = \{d_{1,1}, d_{1,2}, d_{1,3}, d_{1,4}\}$
$E_1 = \{e_{1,1}, e_{1,2}, e_{1,3}, e_{1,4}\}$	$F_1 = \{f_{1,1}, f_{1,2}, f_{1,3}, f_{1,4}\}$
$G_1 = \{g_{1,1}, g_{1,2}, g_{1,3}, g_{1,4}\}$	$H_1 = \{h_{1,1}, h_{1,2}, h_{1,3}, h_{1,4}\}$
$I_1 = \{i_{1,1}, i_{1,2}, i_{1,3}, i_{1,4}\}$	$J_1 = \{j_{1,1}, j_{1,2}, j_{1,3}, j_{1,4}\}$
$K_1 = \{k_{1,1}, k_{1,2}, k_{1,3}, k_{1,4}\}$	$L_1 = \{\ell_{1,1}, \ell_{1,2}, \ell_{1,3}, \ell_{1,4}\}$
$M_1 = \{m_{1,1}, m_{1,2}, m_{1,3}, m_{1,4}\}$	$N_1 = \{n_{1,1}, n_{1,2}, n_{1,3}, n_{1,4}\}$
$O_1 = \{o_{1,1}, o_{1,2}, o_{1,3}, o_{1,4}\}$	$P_1 = \{p_{1,1}, p_{1,2}, p_{1,3}, p_{1,4}\}$

TABLE XI
RESULTING DATA SEGMENTS IN STEP 2.1

$\alpha_{1,1} = a_{1,1} + b_{1,1} + c_{1,1} + d_{1,1}$	$\alpha_{1,2} = a_{1,2} + b_{1,2} + c_{1,2} + d_{1,2}$
$\alpha_{1,3} = a_{1,3} + b_{1,3} + c_{1,3} + d_{1,3}$	$\alpha_{1,4} = a_{1,4} + b_{1,4} + c_{1,4} + d_{1,4}$
$\beta_{1,1} = e_{1,1} + f_{1,1} + g_{1,1} + h_{1,1}$	$\beta_{1,2} = e_{1,2} + f_{1,2} + g_{1,2} + h_{1,2}$
$\beta_{1,3} = e_{1,3} + f_{1,3} + g_{1,3} + h_{1,3}$	$\beta_{1,4} = e_{1,4} + f_{1,4} + g_{1,4} + h_{1,4}$
$\gamma_{1,1} = i_{1,1} + j_{1,1} + k_{1,1} + \ell_{1,1}$	$\gamma_{1,2} = i_{1,2} + j_{1,2} + k_{1,2} + \ell_{1,2}$
$\gamma_{1,3} = i_{1,3} + j_{1,3} + k_{1,3} + \ell_{1,3}$	$\gamma_{1,4} = i_{1,4} + j_{1,4} + k_{1,4} + \ell_{1,4}$
$\theta_{1,1} = m_{1,1} + n_{1,1} + o_{1,1} + p_{1,1}$	$\theta_{1,2} = m_{1,2} + n_{1,2} + o_{1,2} + p_{1,2}$
$\theta_{1,3} = m_{1,3} + n_{1,3} + o_{1,3} + p_{1,3}$	$\theta_{1,4} = m_{1,4} + n_{1,4} + o_{1,4} + p_{1,4}$

TABLE XII
STEP 2.2

$A_2 = \{a_{2,1}, a_{2,2}, a_{2,3}, a_{2,4}\}$	$B_2 = \{b_{2,1}, b_{2,2}, b_{2,3}, b_{2,4}\}$
$C_2 = \{c_{2,1}, c_{2,2}, c_{2,3}, c_{2,4}\}$	$D_2 = \{d_{2,1}, d_{2,2}, d_{2,3}, d_{2,4}\}$
$E_2 = \{e_{2,1}, e_{2,2}, e_{2,3}, e_{2,4}\}$	$F_2 = \{f_{2,1}, f_{2,2}, f_{2,3}, f_{2,4}\}$
$H_2 = \{h_{2,1}, h_{2,2}, h_{2,3}, h_{2,4}\}$	$G_2 = \{g_{2,1}, g_{2,2}, g_{2,3}, g_{2,4}\}$
$I_2 = \{i_{2,1}, i_{2,2}, i_{2,3}, i_{2,4}\}$	$J_2 = \{j_{2,1}, j_{2,2}, j_{2,3}, j_{2,4}\}$
$K_2 = \{k_{2,1}, k_{2,2}, k_{2,3}, k_{2,4}\}$	$L_2 = \{\ell_{2,1}, \ell_{2,2}, \ell_{2,3}, \ell_{2,4}\}$
$M_2 = \{m_{2,1}, m_{2,2}, m_{2,3}, m_{2,4}\}$	$N_2 = \{n_{2,1}, n_{2,2}, n_{2,3}, n_{2,4}\}$
$O_2 = \{o_{2,1}, o_{2,2}, o_{2,3}, o_{2,4}\}$	$P_2 = \{p_{2,1}, p_{2,2}, p_{2,3}, p_{2,4}\}$

TABLE XIII
RESULTING DATA SEGMENTS IN STEP 2.2

$\alpha_{2,1} = a_{2,1} + e_{2,1} + i_{2,1} + m_{2,1}$	$\alpha_{2,2} = a_{2,2} + e_{2,2} + i_{2,2} + m_{2,2}$
$\alpha_{2,3} = a_{2,3} + e_{2,3} + i_{2,3} + m_{2,3}$	$\alpha_{2,4} = a_{2,4} + e_{2,4} + i_{2,4} + m_{2,4}$
$\beta_{2,1} = b_{2,1} + f_{2,1} + j_{2,1} + n_{2,1}$	$\beta_{2,2} = b_{2,2} + f_{2,2} + j_{2,2} + n_{2,2}$
$\beta_{2,3} = b_{2,3} + f_{2,3} + j_{2,3} + n_{2,3}$	$\beta_{2,4} = b_{2,4} + f_{2,4} + j_{2,4} + n_{2,4}$
$\gamma_{2,1} = c_{2,1} + g_{2,1} + k_{2,1} + o_{2,1}$	$\gamma_{2,2} = c_{2,2} + g_{2,2} + k_{2,2} + o_{2,2}$
$\gamma_{2,3} = c_{2,3} + g_{2,3} + k_{2,3} + o_{2,3}$	$\gamma_{2,4} = c_{2,4} + g_{2,4} + k_{2,4} + o_{2,4}$
$\theta_{2,1} = d_{2,1} + h_{2,1} + \ell_{2,1} + p_{2,1}$	$\theta_{2,2} = d_{2,2} + h_{2,2} + \ell_{2,2} + p_{2,2}$
$\theta_{2,3} = d_{2,3} + h_{2,3} + \ell_{2,3} + p_{2,3}$	$\theta_{2,4} = d_{2,4} + h_{2,4} + \ell_{2,4} + p_{2,4}$

Step 3: Perform a reduce-scatter operation on the data segments obtained in Step 2.

Step 3.1: Perform a reduce-scatter operation on the data segments obtained in Step 2.1 within the cycles of L_2, followed by an all-gather operation [see Fig. 12(b)]. The resulting data segments are divided as follows:

$$\begin{aligned} T_{1,1} &= \alpha_{1,1} + \beta_{1,1} + \gamma_{1,1} + \theta_{1,1} \\ T_{1,2} &= \alpha_{1,2} + \beta_{1,2} + \gamma_{1,2} + \theta_{1,2} \\ T_{1,3} &= \alpha_{1,3} + \beta_{1,3} + \gamma_{1,3} + \theta_{1,3} \\ T_{1,4} &= \alpha_{1,4} + \beta_{1,4} + \gamma_{1,4} + \theta_{1,4}. \end{aligned}$$

Step 3.2: Perform a reduce-scatter operation on the data segments obtained in Step 2.2 within the cycles of L_1, followed by an all-gather operation [see Fig.12(e)]. The resulting data segments are divided as follows:

$$\begin{aligned} T_{2,1} &= \alpha_{2,1} + \beta_{2,1} + \gamma_{2,1} + \theta_{2,1} \\ T_{2,2} &= \alpha_{2,2} + \beta_{2,2} + \gamma_{2,2} + \theta_{2,2} \\ T_{2,3} &= \alpha_{2,3} + \beta_{2,3} + \gamma_{2,3} + \theta_{2,3} \\ T_{2,4} &= \alpha_{2,4} + \beta_{2,4} + \gamma_{2,4} + \theta_{2,4}. \end{aligned}$$

Step 4: Perform an all-gather operation on the data segments obtained in Step 3.

Step 4.1: Perform an all-gather operation on the data segments obtained in Step 3.1 within the cycles of L_1, followed by an all-gather operation [see Fig. 12(c)]. The resulting data segments are divided as $\chi_1 = T_{1,1} + T_{1,2} + T_{1,3} + T_{1,4}$.

Step 4.2: Perform an all-gather operation on the data segments obtained in Step 3.2 within the cycles of L_2, followed by an all-gather operation [see Fig. 12(f)]. The resulting data segments are divided as $\chi_2 = T_{2,1} + T_{2,2} + T_{2,3} + T_{2,4}$.

Through the aforementioned operations, each vertex eventually contains data segments from both χ_1 and χ_2, meaning that every vertex holds information from all vertices.

REFERENCES

[1] B. Alspach et al., *Decomposition Into Cycles I: Hamilton Decompositions, Cycles and Rays*. Berlin, Germany: Springer, 1990.
[2] J. A. Bondy and U. S. R. Murty, *Graph Theory*. Berlin, Germany: Springer, 2008.
[3] J.-C. Bermond, T. Kodate, S. Perennes, A. Bonnecaze, and P. Sole, "Broadcasting in hypercubes in the circuit switched model," in *Proc. 14th Int. Parallel Distrib. Process. Symp.*, Cancun, Mexico, 2000, pp. 21–26.
[4] N. R. Adiga et al., "Blue Gene/L torus interconnection network," *IBM J. Res. Dev.*, vol. 49, no. 2.3, pp. 265–276, Mar. 2005.
[5] Baidu, "Baidu ring all-reduce," 2017. [Online]. Available: https://github.com/baidu-research/baidu-allreduce
[6] A. Devraj et al., "Accelerating All reduce with a persistent straggler," 2025. [Online]. Available: https://arxiv.org/abs/2505.23523
[7] H. Dong et al., "Beyond a single AI cluster: A survey of decentralized LLM training," 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2503.11023
[8] C. David et al., "LogP: Towards a realistic model of parallel computation," *ACM SIGPLAN Notices*, vol. 28, no. 7, pp. 1–12, 1993.
[9] A. Gibiansky, "Bringing HPC techniques to deep learning," (n.d.). [Online]. Available: http://research.baidu.com/bringing-hpc-techniques-deep-learning
[10] H. Geng and X. Li, "A gradient-based topology design algorithm applied to second-order network for pursuing best possible synchronization," in *Proc. 39th Youth Academic Annu. Conf. Chin. Assoc. Automat.*, Dalian, China, 2024, pp. 618–623.
[11] J. Geng et al., "HiPS: Hierarchical parameter synchronization in large-scale distributed machine learning," in *Proc. Workshop Netw. Meets AI ML*, 2018, pp. 1–7.

[12] L. Hui, W. Yang, F. Wu, Y. Wang, F. Lyu, and Y. Zhang, "DirectReduce: A scalable ring AllReduce offloading architecture for torus topologies," *IEEE Internet Things J.*, vol. 12, no. 16, pp. 32951–32964, Aug. 2025.
[13] X. Jia et al., "Highly scalable deep learning training system with mixed precision: Training imagenet in four minutes," in *Proc. 32nd Annu. Conf. Neural Inf. Process. Syst. (NeurIPS 2018), Workshop Syst. ML Open Source Softw.*, Canada, 2018, [Online]. Available: https://arxiv.org/abs/1807.11205
[14] Y. Jiang, H. Gu, Y. Lu, and X. Yu, "2D-HRA: Two-dimensional hierarchical ring-based all-reduce algorithm in large-scale distributed machine learning," *IEEE Access*, vol. 8, pp. 183488–183494, 2020.
[15] H. Jin and Y. Wu, "CE-CoLLM: Efficient and adaptive large language models through cloud-edge collaboration," 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2411.02829
[16] S. Kumar and N. Jouppi, "Highly available data parallel ML training on mesh networks," (n.d.). [Online]. Available: https://doi.org/10.48550/arXiv.2011.03605
[17] D. Lim and J. Kim, "TidalMesh: Topology-driven AllReduce collective communication for mesh topology," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, Las Vegas, NV, USA, 2025, pp. 1526–1540.
[18] H. Masuyama and E. Masuyama, "An all-to-all broadcasting algorithm for faulty hypercubes," in *Proc. 3rd ACS/IEEE Int. Conf. Comput. Syst. Appl.*, Cairo, Egypt, 2005, pp. 18–21.
[19] H. Mikami et al., "ImageNet/ResNet-50 training in 224 seconds," 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1811.05233
[20] NVIDIA, NCCL: Optimized primitives for collective multi-GPU communication, 2022. [Online]. Available: https://github.com/NVIDIA/nccl
[21] M. Ruefenacht, M. Bull, and S. Booth, "Generalisation of recursive doubling for allreduce: Now with simulation," *Parallel Comput.*, vol. 69, pp. 24–44, 2017.
[22] A. Roy et al., "Inside the social network's (datacenter) network," in *Proc. ACM Conf. Special Int. Group Data Commun.*, 2015, pp. 123–137.
[23] Y. Ueno and R. Yokota, "Exhaustive study of hierarchical all reduce patterns for large messages between GPUs," in *Proc. 19th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, Larnaca, Cyprus, 2019, pp. 430–439.
[24] S. Wang, D. Li, J. Geng, Y. Gu, and Y. Cheng, "Impact of network topology on the performance of DML: Theoretical analysis and practical factors," in *Proc. IEEE INFOCOM 2019-IEEE Conf. Comput. Commun.*, Paris, France, 2019, pp. 1729–1737.
[25] J. Wang, B. Liang, Z. Zhu, E. Thepie Fapi, and H. Dalal, "Communication-efficient network topology in decentralized learning: A joint design of consensus matrix and resource allocation," *IEEE Trans. Netw.*, vol. 33, no. 2, pp. 761–776, Apr. 2025.
[26] E. P. Xing et al., "Strategies and principles of distributed machine learning on Big Data," *Engineering*, vol. 2, no. 2, pp. 179–195, 2016.
[27] H. Zhang et al., "Poseidon: An efficient communication interface for distributed deep learning on GPU clusters," in *Proc. Annu. Tech. Conf.*, Santa Clara, 2017, pp. 181–193.
[28] L. Zhang, S. Shi, X. Chu, W. Wang, B. Li, and C. Liu, "DeAR: Accelerating distributed deep learning with fine-grained all-reduce pipelining," in *Proc. IEEE 43rd Int. Conf. Distrib. Comput. Syst.*, Hong Kong, 2023, pp. 142–153.
[29] R. Zong et al., "IBing: An efficient interleaved bidirectional ring all-reduce algorithm for gradient synchronization," *ACM Trans. Archit. Code Optim.*, vol. 22, no. 1, pp. 1–23, 2025.
[30] R. Zong, J. Zhang, Z. Tang, and A. Datta, "Topology decoupled all-reduce algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Hyderabad, India, 2025, pp. 1–5.

Huimei Guo received the M.S. degree in applied mathematics from the College of Mathematics and Systems Science, Xinjiang University, Xinjiang, China, in 2022. She is currently working toward the Ph.D. degree in statistics with the Department of Mathematics, Beijing Jiaotong University, Beijing, China.

Her research interests include graph theory, high performance computing, deep learning, parallel computing optimization, and fault diagnosis.

Shuo Quan received the bachelor's degree in software engineering and the master's degree in software engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2013 and 2016, respectively.

He is a Researcher at China Telecom Cloud Computing Research Institute, Beijing, China. His research interests include cloud computing, networking and communication systems, Cloud Network Convergence, Cloud-native and Virtualization, etc.

Rong-Xia Hao received the Ph.D. degree in mathematics from Beijing Jiaotong University, Beijing, China, in 2002.

From 1998 to 2006, she was an Associate Professor. Since 2006, she was a Professor with the Department of Mathematic, Beijing Jiaotong University. Her research interests include graph theory, interconnection network, fault tolerant computing and parallel computing optimization.

Dr. hao was the recipient of Beijing Jiaotong University Zhi Jin Foundation Outstanding Youth Teaching Award in 2007 and the First Prize of 2008 Excellent Paper Awarded by Beijing Operations Research Society.

Jie Wu (Fellow, IEEE) received the B.S. degree and the M.S. degree in computer science from Shanghai University of Science and Technology, Shanghai, China, and the Ph.D. degree in computer engineering from Florida Atlantic University, Boca Raton, FL, USA, in 1989.

He is currently the Director of Center for Networked Computing and a Laura H.Carnell Professor with Temple University, Philadelphia, PA, USA. He previously served as Department Chair, Associate Vice Provost, and NSF Program Director. His research interests include mobile computing, wireless networks, cloud computing, and network security.

Dr. Wu is on multiple editorial boards and has chaired major conferences including IEEE INFOCOM and IPDPS. He was the recipient of the 2011 CCF Overseas Outstanding Achievement Award. He is currently on leave working as a Scientist in Cloud Computing at China Telecom.