Efficient Collective Communication in Interconnection Networks with Shortcut Network Topologies

Ke Cui

Doctor of Philosophy

Department of Informatics
School of Multidisciplinary Sciences

The Graduate University for Advanced Studies, SOKENDAI
March 2024

Abstract

Many applications running on large parallel computers are designed to solve various challenging problems, such as weather forecasting and drug discovery. To solve these challenging problems more efficiently, modern large parallel systems (supercomputers) rely on hundreds of thousands of compute nodes to provide powerful computational capabilities. Applications running on parallel computers usually require the collaborative work of a large number of compute nodes to complete the computational task; the collaboration between compute nodes involves a great deal of mutual communication. Collective communication, a mode of communication involving multiple processes or compute nodes, is widely used in applications running on parallel computers.

For many parallel applications, the collective communication operations dominate the execution time. Improving the performance of collective communication operations is a crucial way to shorten the execution time of the applications. The performance of collective communication operations is affected by both hardware and software. At the hardware level, a crucial factor is the interconnection network, especially network topology. Network topology dictates the connectivity among compute nodes within parallel computers. A low latency network topology reduces the communication overhead between compute nodes and thus improves the performance of collective communication operations. At the software level, the implementation of algorithms for collective communication operations and the mapping strategy of jobs and processes can affect the performance of collective communication operations.

In this dissertation, we present efficient collective communication in parallel computers with shortcut network topologies. The shortcut network topology consists of a baseline topology with shortcut links, which shorten the diameter and average shortest path length (ASPL). We propose the implementation of efficient collective

communication operations on two types of network topologies: the random shortcut topology and the non-random shortcut topology.

The first approach proposes implementing efficient collective communication using random shortcut network topologies. The random network topologies can be used to achieve low hop counts between nodes and, thus, low latency on average. We describe three process mapping strategies on random shortcut topologies: random mapping, hierarchical-tree mapping, and ring-based consecutive mapping. Then, we apply the two-opt algorithm to the mapped compute nodes to optimize the rank placement for building efficient collective communication operations on random shortcut network topologies. The two-opt approach minimizes the total path hops or possible communication contention of point-to-point communications that form the target collective communication by implementing the process rank re-placement for efficient collective communication. Our proposed two-opt approach can significantly reduce the number of hops for collective communication operations such as Broadcast, Allreduce, and Alltoall. SimGrid discrete-event simulation results show that the two-opt approach can dramatically improve the performance of collective communication and, in parallel applications where collective communication operations dominate, the two-opt approach can improve the overall performance of the application.

The second approach proposes using circulant network topologies. Unlike random shortcut topologies, a circulant topology is obtained by adding non-random links to a ring topology. The circulant network topologies provide algorithmic features that reduce the total hop counts of some typical collective communication algorithms; these features make them ideal for collective communication operations. We propose two process mapping strategies for circulant network topologies: ring-based consecutive mapping and circulant mapping. These two mapping strategies result in very low path hops of collective communication operations, and it is even possible to achieve the optimal hops. SimGrid discrete-event simulation results show that ring-based consecutive mapping and circulant mapping significantly improve the performance of collective communication operations. We also evaluate the parallel applications on circulant network topologies; the application evaluation results also show that when communication operations dominate the performance, the low hops mapping strategies on circulant network topologies achieve better performance.

Finally, we compare these two shortcut network topologies. Circulant network

topologies have a higher diameter and average shortest path length(ASPL) than random shortcut network topologies in the case of having the same degree. However, the collective communication operations(e.g., Broadcast, Allreduce, and Alltoall) on circulant network topologies have far fewer hops than random shortcut network topologies, which results in better performance for collective communication operations on a circulant network topology. Moreover, we compare the cable length of random shortcut topologies and circulant topologies with the same degree; the circulant network topologies have much less cable length than random topologies, which makes the circulant network topologies less costly than the random shortcut network topologies in constructing network interconnection for parallel computers.

Acknowledgments

I want to express my deepest gratitude to all those who have made the journey of my doctoral studies both possible and fruitful.

First and foremost, my utmost appreciation goes to my supervisor, Michihiro Koibuchi, for the unwavering mentorship, patience, and invaluable insights that have significantly shaped this work. His profound knowledge and astute guidance have been instrumental in navigating the complexities of my research.

I am particularly indebted to my colleagues in the research group, whose camaraderie, collaboration, and shared wisdom have immensely enriched my research experience. Your companionship has been one of the most rewarding aspects of my doctoral journey.

My heartfelt thanks go to my friends, who have provided a sense of home and a reprieve from the pressures of academic life. Your unwavering support, humor, and kindness have made this journey more enjoyable and possible. To my dear parents, words cannot express the gratitude I hold in my heart for you. You have instilled in me the values of hard work and perseverance and have supported me unconditionally throughout this process. Your love, sacrifice, and belief in my abilities have been the bedrock of my resilience and success. I am forever grateful for your nurturing guidance and for always being my steadfast source of strength and inspiration.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Network Topology

Current parallel computers consist of hundreds of thousands of compute nodes. For example, the world's most powerful supercomputer, Frontier, has 9,472 compute nodes as of Nov. 2022 [4]. As the demand for computing power increases, the number of compute nodes of supercomputers will increase.

Large-scale parallel computers suffer from communication latency between compute nodes. The design of low-latency network interconnections is a crucial component in developing high-performance parallel computers. In large parallel computers, the latency between compute nodes generally falls within tens to hundreds of nanoseconds. The 36-port non-blocking 200Gb/s EDR InfiniBand switch in supercomputers Summit and Sierra has a latency of 90 nanoseconds [5].

When it comes to interconnecting large-scale parallel computers, it is crucial to consider not only the cutting-edge high throughput and low latency hardware components, such as low-latency switches [6], but also the network topology design. A

network topology refers to the arrangement or structure of its node connections and can significantly impact the system's efficiency and scalability [7]. A broad spectrum of network topologies has been employed in high-performance parallel computers, while many topologies have been proposed as potential candidates.

In the context of high-performance parallel computing, it is common for applications to be executed on multiple compute nodes, necessitating message exchanges between them. A message originating from a source node can traverse several intermediate nodes before reaching its destination node. The distance between the source and destination node is called the hop count and refers to the number of channels and nodes that a message must traverse to reach its destination [7]. The communication latency between two nodes is directly proportional to the number of intermediary nodes, or hops, along the transmission path. The objective of minimizing communication latency between two nodes can be achieved by ensuring that the distance between the nodes is as tiny as possible. The most efficient way to accomplish this is by having a fully connected network topology, in which each node is directly connected to every other node in the network. In this network topology, the hop count between any two nodes is always one, which minimizes communication latency. Utilizing fully connected network topology is a strategy specific small-scale networks employ to achieve their low latency objectives[8].

However, due to scalability, cost, and complexity limitations, fully connected network topology is not typically used in large-scale networks. Upon evaluating the trade-offs associated with scalability, cost, and complexity, a diverse range of non-fully connected network topologies have been suggested as viable candidates for high-performance computer systems to attain the objective of minimizing latency. Diameter and average shortest path length (ASPL) are crucial factors typically considered for interconnection network topology. Diameter refers to the maximum distance between any two nodes in a network. ASPL, on the other hand, is the average distance between any two nodes in the network. In a network topology with a small diameter and low ASPL, the distance between any two nodes is short, meaning messages can travel between nodes with fewer hops and less latency.

The small world effect can lead to the formation of networks with a low diameter and low ASPL [9]. Adding random shortcuts to regular network topologies, such as torus and ring, can significantly decrease the network's diameter and ASPL. Random

shortcut network topologies have been proposed to construct interconnection networks with low latency [2].

Another approach is to borrow a finding of graph theory. The order/degree problem is computationally demanding to identify the graph with the minor diameter and ASPL for a given number of nodes and degrees [10]. The graph golf competition imposes constraints on the number of nodes and degrees, requiring participants to identify the graph with the most minor diameter and ASPL that satisfies these constraints [11]. We can use the graphs from the database of the graph golf competition. Fixed low-diameter topologies, such as Dragonfly [12] and Slim Fly [13], have been recognized as effective alternatives for achieving low-latency interconnects, and have been implemented in supercomputers, including Frontier, which is currently the fastest computer [14]. Frontier employs the Dragonfly network topology for its high-speed interconnects [15]. Dragonfly and Slimfly are characterized by diameters of 3 and 2, signifying that messages can traverse between nodes within a maximum of 3 and 2 hops, respectively.

1.2 Collective Communication

The low-latency hardware components and the low-latency network topology described above are crucial for parallel computers. Other aspects, such as routing and collective communication, also significantly impact the performance of applications running in parallel computers. Collective communication is a critical aspect of parallel computing, and it plays a significant role in determining the performance of parallel applications. In numerous supercomputer applications, a substantial portion of the total program runtime is attributed to collective communication [16]. The optimization of collective communication yields a reduction in communication overhead and time, resulting in enhanced application performance [17, 18].

Collective communication involves multiple processes or nodes in a parallel computer communicating with each other simultaneously [19]. In contrast to point-to-point communication, where two processes exchange messages directly, collective communication involves a group of processes performing a coordinated communication operation, and the outcome depends on the collective behavior of all processes or nodes. Collective communication comprises a range of distinct operations contingent on the behavior of nodes or processes during the communication. These operations

encompass various types, such as broadcast, reduce, gather, scatter, all-to-all, all-reduce, all-gather, and barrier, among others.

Collective communication operations can be implemented in both software and hardware. In software implementations, collective communication operations can be realized by means of high-level APIs. MPI(Message Passing Interface) is a widely used standard for implementing collective communication operations.

MPI provides a set of functions, including MPI_Bcast, MPI_Reduce, MPI_Gather, MPI_Scatter, MPI_Alltoall, and MPI_ Barrier, that facilitate communication among processes [20]. These functions are typically built using lower-level communication primitives, such as peer-to-peer communication. MPI implementations are typically provided by software libraries that offer a range of functions for conducting message-passing operations among processes. Some MPI libraries implementations include Open MPI, MPICH, MVAPICH2, and Intel MPI [21, 22, 23, 24]. Each library has features and performance characteristics, and users can choose the one that best fits their needs and requirements.

Hardware-based implementations of collective communication can be achieved through specialized communication hardware such as network interface cards (NICs), routers, switches, and FPGAs [25, 26, 27, 28]. These hardware devices can provide efficient communication among processes by offloading some communication tasks from the CPU to the specialized hardware. One instance of hardware-based collective communication is exemplified by the NVIDIA InfiniBand switch [6, 28] that employs the Scalable Hierarchical Aggregation Protocol (SHArP) to facilitate the implementation of a computation segment via the switch without requiring compute nodes, thereby significantly decreasing the latency of reduction operations in collective communication [29]. Another example of hardware-based collective communication is the Cray Gemini interconnect used in Cray supercomputers [30]. The Aries collective engine provides hardware support for reduction and barrier operations [31]. In general, hardware-based collective communication can provide faster and more efficient communication among processes than software-based solutions, as it offloads communication tasks to specialized hardware devices, reducing the burden on the CPU and communication latency. However, hardware-based solutions can be more expensive and complex than software-based ones, requiring technical hardware devices.

1.3 Process Mapping

In parallel computers, the execution of an application involves a set of compute nodes, but it is not necessary that all nodes actively participate in the application. Instead, typically, only a subset of nodes cooperatively perform a specific task, which is required for the efficient and timely completion of the application. This selective participation of nodes is essential in large-scale parallel computers, where communication overheads may increase with the number of nodes involved. The allocation of compute nodes, processors, cores, and other resources can significantly affect the performance of an application. As a result, mapping techniques that determine the allocation of resources are critical for parallel computers [32, 33, 34]. The growth of high-performance computing systems in size and complexity has increased the importance of mapping techniques that assign compute nodes and resources to applications.

In a parallel computer, the placement of processes on compute nodes can affect collective communication performance because the latency of communication can be affected by the distance between processes. Process mapping, which is the assignment of processes to compute nodes, can significantly impact communication performance. When processes that require communication are assigned to distant compute nodes, the communication latency increases due to the increased number of communication hops, leading to performance degradation. Many studies focus on developing process mapping techniques to improve communication performance in parallel computers. These techniques aim to reduce communication latency by minimizing the distance between communicating processes and reducing communication contention [35, 36, 37]. The task mapping processes onto a specific network topology, considering the underlying network topology and the application's communication patterns, referred to as topology-aware task/process mapping. Many studies have been conducted on process mapping techniques for specific network topologies commonly found in parallel computers, such as torus [38, 39, 40, 41], fat-tree [42, 43, 44], and Dragonfly[45, 46, 47]. Topology-aware process mapping techniques consider the underlying network topology and exploit its geometric characteristics to assign tasks or processes in a way that reduces communication latency.

1.4 Motivation and Objects

The research problem is centered on designing efficient collective communication in parallel applications to enhance their overall performance. Collective communication is paramount for the effective execution of parallel algorithms, as it necessitates the simultaneous exchange of data among multiple processes. A wide array of factors, including both hardware and software components, exert influence on the performance of collective communication. As explicated in Chapters 1.1 and 1.3, the interconnection network topology utilized in parallel computing architectures, and the employed process mapping strategies substantially affect the effectiveness of collective communication. Suboptimal communication can result in bottlenecks and considerably impede the performance of parallel applications.

This research aims to pursue efficient collective communication by examining network topology and process mapping strategies within parallel computers. As elaborated in Chapter 1.1, communication latency between compute nodes becomes an increasingly critical factor influencing application performance as parallel computers scale up. In this dissertation, we tackle this challenge by selecting a random shortcut network topology [2] as a high-speed interconnection and formulating a process's rank placement strategy tailored to the random shortcut network topology. This approach aims to promote efficient collective communication, ultimately enhancing the performance of parallel applications.

Moreover, we investigate a class of non-random shortcut network topologies as high-speed interconnects, which fall into distinct circulant network topologies [48]. Although their diameters and average shortest path lengths (ASPL) exceed those of random shortcut network topologies with an identical degree, we leverage the inherent geometric characteristics of these topologies to devise process mapping strategies that facilitate remarkably efficient collective communication.

The objectives of this research are:

1. Develop a process's rank placement strategy for random shortcut network topology to reduce the number of hops required for collective communication, ultimately resulting in efficient collective communication.

2. Develop a process mapping strategy for particular circulant network topologies.

Each point-to-point communication that makes up a collective communication achieves the theoretical minimum number of hops, i.e., exactly one hop. This approach minimizes the number of hops for collective communication and promotes efficient collective communication.

3. Compare the performance of collective communication, point-to-point communication, and parallel applications on random shortcut network topology and circulant topology. And compare the cost of constructing interconnection networks using these two topologies. Draw a conclusion against the question, "which network topology is better?".

By achieving these objectives, this research contributes to the advancement of knowledge in the field of parallel computing, especially collective communication through the synergistic combination of network topology and process mapping strategies.

1.5 Contributions

1.5.1 Process's Rank Placement Strategy for Random Shortcut Network Topology

Random network topologies have been proposed as a low-latency network for parallel computers. Although collective communication operations are frequently used in a parallel application, collective communication operations are not well-optimized for random network topologies. In this dissertation, We applied the two-opt approach to replace processes' rank for building efficient collective communication on random shortcut network topologies.

The two-opt approach replaces the processes's rank, changing the point-to-point communication that makes up the collective communication, which affects the overall performance of the collective communication.

The discrete-event simulation results significantly enhance collective communication performance using the two-opt approach. This improvement is achieved by reducing the hops of collective communication operations and boosting the over-

all performance of parallel applications, particularly in scenarios where collective communication plays a dominant role in application performance.

1.5.2 Process Mapping for Circulant Network Topologiy

Circulant network topology is widely used in various network designs. They are used as a ring extension to design and implement local area networks. The networks are also called distributed loop computer networks [49, 50]. Circulant network topology has a higher ASPL than that of random network topology. Nevertheless, we are interested in circulant network topology regarding hop counts of collective communication. Circulant network topology can fit with some typical collective communication algorithms.

We present a process mapping strategy for a circulant network topology, which enables a theoretical minimum hop count on some collective communication operations such as Broadcast, Allreduce, and Alltoall.

The discrete-event simulation results in SimGrid show that our proposed mapping strategy, which aims to minimize the number of hops in collective communication, significantly improves the collective communication performance compared to other mapping strategies.

1.5.3 Comparison of Shortcut Network Topologies

The random shortcut network topology and the circulant topology can be obtained by adding links to a basic ring topology, with the difference that the random shortcut network topology is obtained by adding links randomly. In contrast, the torus is obtained by adding links non-randomly.

The random shortcut network topology has a lower diameter and average shortest path length(ASPL). This intuitively suggests that communication within this topology involves fewer hops. Our comparison includes collective communication and point-to-point communication in these two shortcut network topologies.

Despite the circulant network topology's higher diameter and average shortest path length, adopting an appropriate mapping strategy can significantly reduce the hops of collective communication, leading to its efficiency. Furthermore, our proposed mapping

strategy enables even more lower hops point-to-point communication, especially when dealing with fewer nodes in communication scenarios.

We also compare the cost of building interconnection networks for parallel computers using random shortcut network topology and circulant topology. The circulant network topology allows for a relatively low cost.

1.6 Dissertation Organization

The rest of this dissertation is organized as follows.

In Chapter 3, we describe implementing efficient collective communication in random shortcut network topology.

In Chapter 4, we describe the efficient collective communication in circulant network topology.

In Chapter 5, we mainly compare the two shortcut network topologies quantitatively.

In Chapter 6, we illustrate the conclusion of this dissertation and future direction.

Figure 1.1 shows the relationship of chapters.

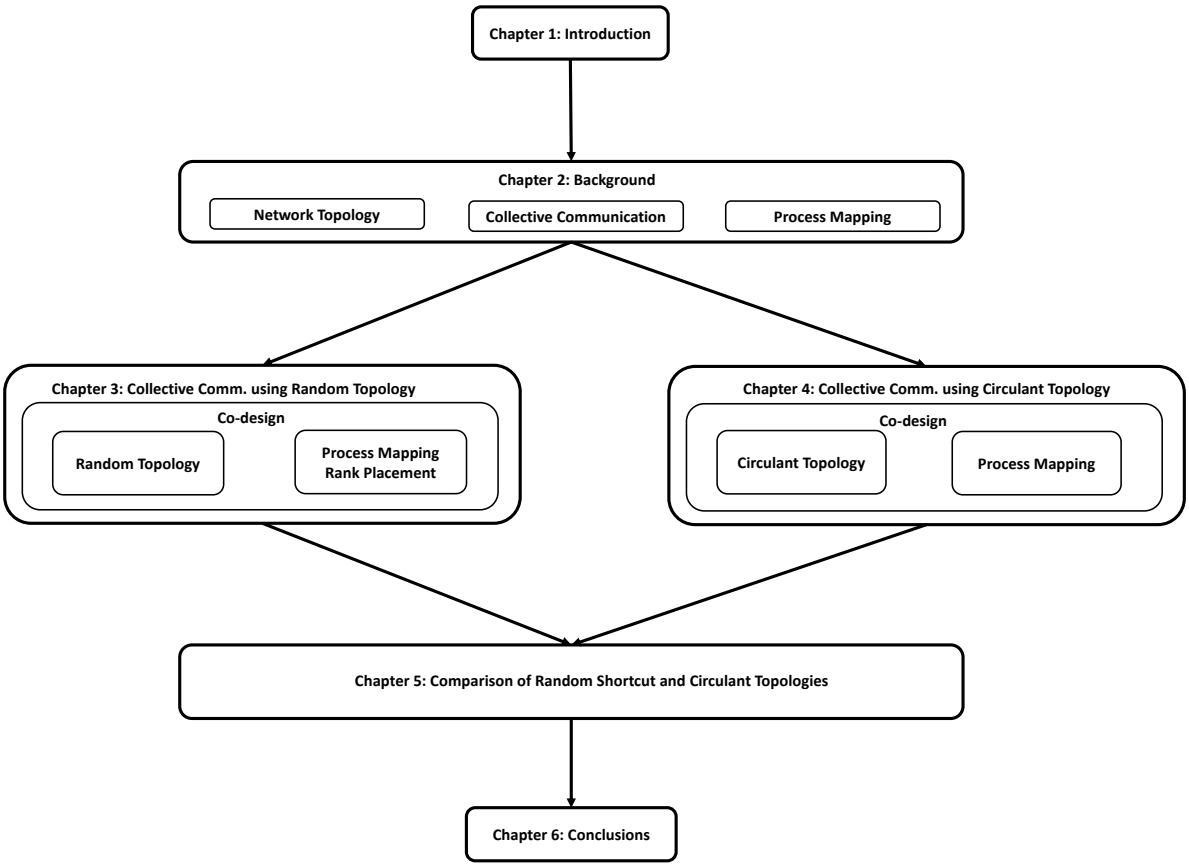


Figure 1.1: Dissertation organization.

2

Background

In this chapter, we provide an introduction to the background of the factors that affect the communication performance of parallel applications on parallel computers. We introduce the basic concept of network topology in parallel computers and explore the various network topologies with low diameter and ASPL(Average Shortest Path Length). Moving forward, we delve into collective communication, which plays a crucial role in parallel computing. We discuss the common operations involved in collective communication, such as Broadcast, Allreduce, and Alltoall, and introduce some common algorithms used to implement these operations. Finally, we introduce the concept of process mapping and its significance in parallel computing, highlighting how mapping tasks to the underlying hardware can impact communication performance. Their relationship is illustrated in Figure 2.1.

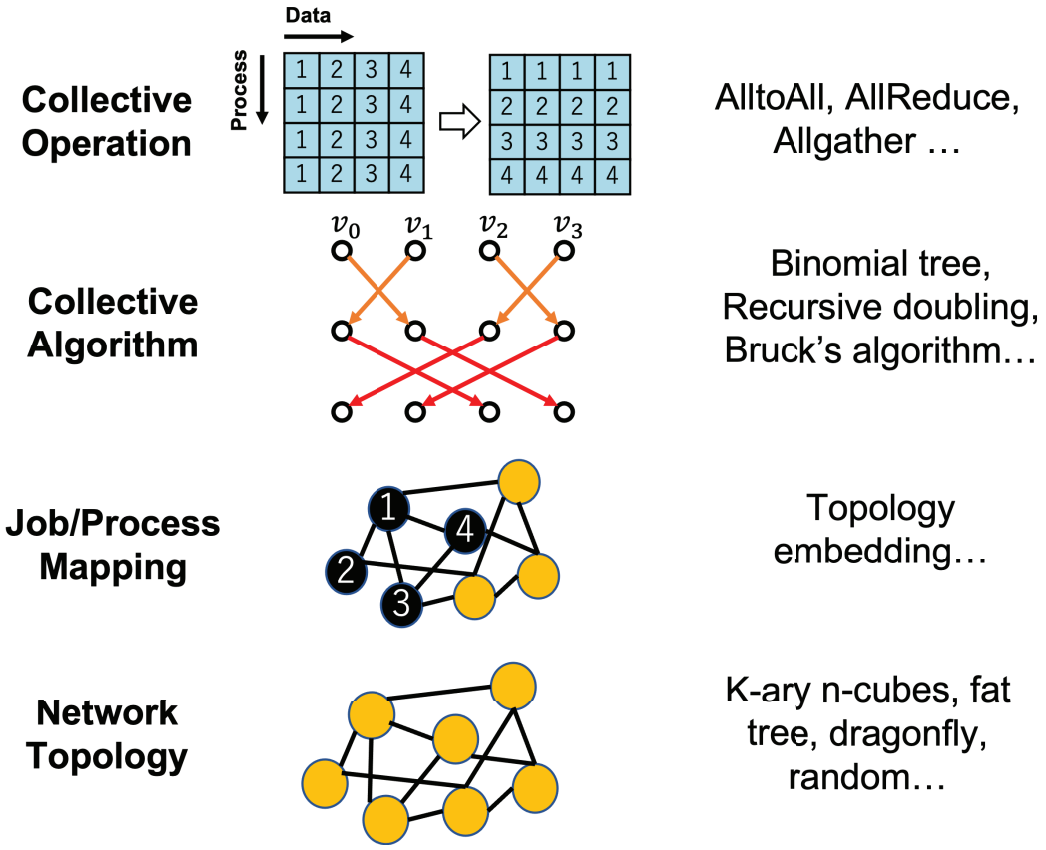


Figure 2.1: Four performance factors in collective communication.

2.1 Interconnection Network

2.1.1 Concept of Network Topology

An interconnection network serves as a communication infrastructure connecting various components within a computing system, including processing elements (PEs) such as processors, cores within multi-core processors, and specialized hardware accelerators like Graphics Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs), as well as routers, switches, memory modules and peripheral devices [51]. Interconnection networks are pivotal in determining a system's overall performance, communication latency, fault tolerance, and scalability. They facilitate the efficient exchange of data and messages between components, ensuring the effective execution of computational tasks. In parallel computers, interconnection networks are crucial in

establishing connections between various components, such as PEs and switches. These networks facilitate communication and cooperation among PEs, allowing them to work together to solve complex computational problems by dividing tasks into smaller subtasks and processing them simultaneously.

The topology of an interconnection network in parallel computers refers to the arrangement of nodes and communication links in a network, specifically focusing on how these elements are interconnected [51]. The chosen network topology in a parallel computer defines the potential routes for transmitting packets throughout the interconnected network. As a result, the selection of network topology considerably impacts system performance, communication latency, fault tolerance, and scalability [52, 2, 53, 54, 55, 56].

A network topology can be described as a graph G = (V, E), where V represents the set of vertices (or nodes) and E represents the set of edges (or links) connecting the nodes. In parallel computing, each vertex in V corresponds to either a PE or a switch, depending on the specific network architecture. Meanwhile, each edge in E represents a link between two nodes, facilitating data flow and cooperation among PEs or between PEs and switches.

The properties of a network topology can be quantified using various metrics, such as:

- **Degree (d):** The degree of a node in a network topology is defined as the number of direct connections to other nodes in a node. A higher degree usually implies excellent connectivity and lower communication latency within the parallel computer.

- **Diameter (D):** The diameter of a network topology is the longest path between any two nodes in the network. This metric is essential for assessing the worst-case communication latency in a parallel computer and understanding the maximum delay between any two nodes.

- **Average Shortest Path Length (ASPL):** The ASPL is the mean of the shortest paths between all pairs of nodes in the network. It measures the average number of hops or communication links that must be traversed to reach any destination node from a given source node, considering the shortest possible path between

them. A lower ASPL suggests more efficient communication within the network, resulting in reduced communication latency and improved overall performance for the parallel computer.

In parallel computing, network topologies are primarily classified into direct and indirect networks [51].

- **Direct networks:** Direct networks consist of direct connections between PEs without intermediate devices, facilitating low-latency communication. Notable examples of direct networks include mesh, torus, and hypercube topologies.

- **Indirect networks:** Indirect networks, commonly known as switched networks, entail communication through intermediate devices such as switches or routers. Topologies such as Clos [57, 58], fat-tree [59], and butterfly [60, 51, 61] fall under this category.

Direct and indirect networks have unique advantages and disadvantages when employed in parallel computers. Direct networks are characterized by the absence of intermediate devices between PEs, leading to lower latency communication and dedicated bandwidth for each pair of PEs. This setup enables faster data exchange between PEs and reduces communication overhead. However, intermediate devices in indirect networks increase routing complexity and potentially add latency to communication between PEs.

intermediate devices, such as switches or routers. While this configuration may introduce higher latency communication and shared bandwidth managed by the intermediate devices, it offers superior scalability, fault tolerance, and resource utilization compared to direct networks. Indirect networks can accommodate many PEs more efficiently, making them more suitable for extensive parallel computers. However, intermediate devices in indirect networks increase routing complexity and can potentially add latency to communication between PEs. Ultimately, choosing a network topology for a parallel computer depends on several factors: the number of PEs, the characteristics of the computational tasks, communication patterns, and the balance between performance, fault tolerance, and scalability.

2.1.2 Traditional Network Topologies Used in Supercomputers

A few network topologies are traditionally used to interconnect PEs in most HPC systems and these network topologies can be used to interconnect high-radix switches. Interconnection networks are surveyed for top30 supercomputers in top500 ranking [62], as of Nov. 2019 [1], as shown in Figure 2.2. Torus, Fat Tree, and Dragonfly are frequently employed in the top 30 supercomputers.

Topology	# of Switch Ports	Link Bandwidth, Node Bandwidth	Ranking of Top500 (as of Nov. 2019)
Fat Tree(indirect)	36(Tier 2), 648(Tier 1)[1,2,10] 32 (Tier 2), 576(Tier 1)[4] 40 (Tier 2), 800(Tier 1)[5] 36(Tier 2,3), 648(Tier 1)[8] 48 (Tier 2), 768(Tier1) [14,15,19,23,30] 648 (Tier1)[20]	100Gbps, 200Gbps[1,2,8,11] 112Gbps, 112Gbps[4] 200Gbps, 100Gbps[5] 100Gbps, 100Gbps[14,18,19,30] 100Gbps, 800Gbps[20,26,29] 100Gbps, 400Gbps[23]	19 Machines (1,2,4,5,8-11, 14,15,17-20, 23,24,26,29,30)
Dragonfly (direct)	48	37.5-42Gbps, 84Gbps	5 Machines(6,7,13,27,28)
5-D Torus (direct)	10	16Gbps, Integrated	2 Machines (12,22)
Others			4 Machines (3,16,21,25)

Figure 2.2: Network topologies of top 30 supercomputers (picked up from [1]).

Mesh and Torus

Mesh and torus topologies are direct network topologies frequently employed in parallel computers. Both topologies exhibit grid-like structures, with PEs interconnected to their nearest neighbors. Nevertheless, their edge connections differ, which results in unique characteristics and advantages for each. In a mesh topology, PEs are arranged in a multi-dimensional grid, typically 2D or 3D, and each PE is connected to its closest neighbors along each dimension. However, wraparound connections at the edges are absent. Consequently, PEs at the edges and corners possess fewer connections than those in the center. Mesh topologies offer various benefits, including straightforward routing algorithms and ease of implementation. Nonetheless, the

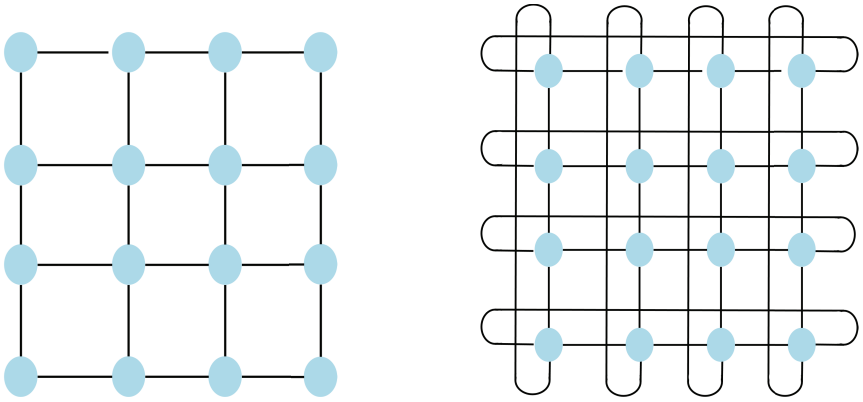
absence of wraparound connections can lead to increased communication delays for PEs at the grid edges.

On the other hand, a torus topology resembles a mesh topology but incorporates wraparound connections at the grid edges. These connections create closed loops in each dimension, rendering the network a toroidal shape. The wraparound connections furnish multiple paths between PEs, enhancing fault tolerance and yielding more uniform communication latency throughout the network. Consequently, torus topologies can exhibit greater scalability and outperform mesh topologies, particularly in systems with extensive communication between PEs.

Both mesh and torus topologies can be considered as k-ary n-cube topologies, with the primary distinction being the presence or absence of wraparound connections at the edges [51]. A k-ary n-cube topology is characterized by a multi-dimensional grid with n dimensions and k nodes in each dimension. Here, both k and n are positive integers. The k-ary n-cube topology has the following properties:

- Number of Nodes: There are k^n nodes in the network, where the total number of nodes is the product of the size of each dimension.
- Degree: Each node has $2n$ connections in the case of a torus topology, connected to its nearest neighbors in each dimension (one in the positive direction and one in the negative direction). For mesh topologies, the degree of a node depends on its position within the grid. Nodes in the interior have $2n$ connections, while nodes on the edges and corners have fewer connections.
- Diameter: For a torus network topology, the diameter is given by $n \cdot \lceil \frac{k}{2} \rceil$. For a mesh network topology, the diameter is given by $n \cdot (k-1)$.
- Average shortest path length(ASPL): For a torus network topology, the ASPL is given by $\frac{n}{4}(k-1)$. For a mesh network topology, the ASPL is given by $\frac{n}{3}(k-1)$ for k > 2 and $\frac{n}{2}(k-1)$ for k = 2.
- Coordinate Representation: Every node in the n-dimensional k-ary n-cube topology can be denoted by an n-tuple of coordinates $(x_1, x_2, \ldots, x_n)$, where $0 \leq x_i < k$ for each i. Nodes with coordinates $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$ are connected if and only if they differ in exactly one dimension by 1 in the mesh.

Nodes with coordinates $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$ are connected if and only if they differ in exactly one dimension by one or $k - 1$ in the torus.



(a) 2-D Mesh (4-ary 2-cube mesh).

(b) 2-D Torus (4-ary 2-cube torus).

Figure 2.3: Mesh and torus topologies.

Figures 2.3(a) and 2.3(b) are 4-ary 2-cube mesh and 4-ary 2-cube torus, respectively.

Over the past few decades, supercomputers have utilized various interconnection network topologies, including mesh and torus. Mesh topologies have been employed in systems like the Intel Paragon [63] with a 2D mesh, the MIT J-Machine [64], and ASCI Red [65] with 3D mesh. Torus topologies have also been widely used, with examples such as iWarp [66] using a 2D torus, Cray T3D, Cray T3E, BlueGene/L and BlueGene/P using 3D tori[67, 68, 69, 70]. Cray supercomputers based Cray Gemini System Interconnect using 3D torus network topology such as Cray XE6 and CrayXK6 [71, 72, 30]. BlueGene/Q [73] uses a 5D torus network topology, while the K supercomputer and Fugaku supercomputer use Tofu and TofuD [74, 75, 76], which are 6D torus. Some of the latest domain-specific architecture supercomputers also employ torus as the network topology for high-performance computing, such as Google's TPU v4 [77] and Anton 3 [78].

Hypercube

A hypercube, also known as an n-cube or binary n-cube, is a direct network topology used in parallel computers. It is a multi-dimensional, recursive, hierarchical structure

with each dimension corresponding to a binary power. In a hypercube network topology, the number of nodes doubles with each additional dimension [79].

Here are some key properties of hypercube topologies:

- Number of Nodes: There are 2^n nodes in an n-dimensional hypercube, where n is a non-negative integer. The total number of nodes is the product of 2 raised to the power of n.
- Dimensions: The hypercube has n dimensions, where n is a non-negative integer. A 0-dimensional hypercube consists of a single node, a 1-dimensional hypercube is a line with two nodes, a 2-dimensional hypercube is a square with four nodes, a 3-dimensional hypercube is a cube with eight nodes, and so on.
- Degree: Each node in a hypercube has n connections connected to its nearest neighbors along each dimension.
- Diameter: The diameter of a hypercube network topology is the longest path between any pair of nodes in the network. For an n-dimensional hypercube, the ASPL is given by $\frac{n}{2}$, as the average number of dimensions that must be traversed to reach another node is half the total.
- ASPL: The ASPL of a hypercube network topology is the average of the shortest path lengths between all pairs of nodes in the network. For an n-dimensional hypercube, the ASPL is given by $\frac{n}{2}$, as the average number of dimensions that must be traversed to reach another node is half the total dimensions.
- Coordinate Representation: Every node in the n-dimensional hypercube can be denoted by an n-tuple of binary coordinates $(b_1, b_2, ..., b_n)$, where $b_i \in 0, 1$ for each i. Nodes with coordinates $(x_1, x_2, ..., x_n)$ and $(y_1, y_2, ..., y_n)$ are connected if and only if they differ in exactly one dimension, and that difference is exactly 1, which can also be expressed as:

$$\sum_{i=1}^{n} |x_i - y_i| = 1 \tag{2.1}$$

Figure 2.4 is an example of 4-dimensional hypercube.

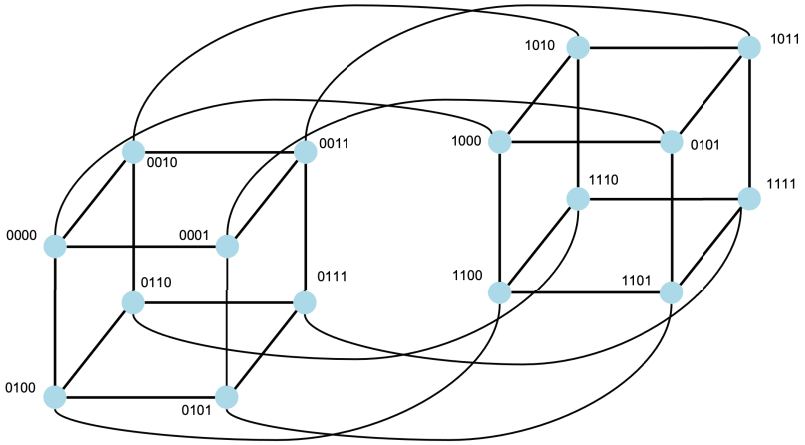


Figure 2.4: Four-dimensional Hypercube.

Hypercube topologies are well-suited for parallel computers due to their high bisection width and support for efficient routing algorithms [80, 81]. However, they may suffer scalability issues as the number of nodes grows exponentially with each additional dimension. Many early parallel computers used hypercube as their network topology, such as iPSC/860 and Ncube 6400 [82]

Fat Tree

Fat trees are hierarchical topologies designed for parallel computers [59]. Fat tree topologies feature a tree-like structure with multiple levels of nodes, and the bandwidth increases as one moves up the hierarchy toward the root. The primary objective of fat tree topologies is to provide high bisection bandwidth, enabling efficient support for high-bandwidth communication between nodes. Fat trees serve as indirect networks in parallel computers. Within fat-tree topologies, all PEs, such as compute nodes, are located at the leaf level of the hierarchical structure. Instead of being directly connected, these PEs are linked to switches.

The k-ary n-trees [83] are a particular class of fat-tree topologies often employed in supercomputers. Here, we will briefly describe k-ary n-tree topologies. In k-ary n-tree topologies, k represents the internal node's branching factor or degree. Each internal node has k children, which can be other internal or leaf nodes. The higher the value of k, the more children each internal node has, resulting in a more comprehensive tree. In

a switch-based k-ary n-tree parallel computers, the number of the PEs connected to a switch is k. The n in k-ary n-tree topologies represents the depth or the number of levels in the tree structure, excluding the root level. A more significant value of n implies a deeper tree with more hierarchical levels. The depth of the tree affects communication latency and routing complexity within the network. Figure 2.5 is an example of a 4-ary 2-tree.

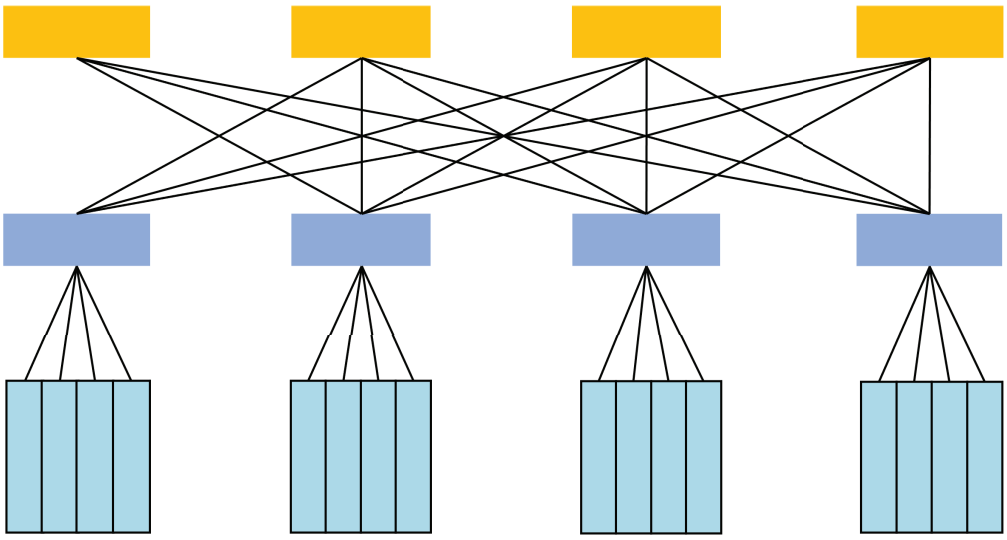


Figure 2.5: 4-ary 2-tree

Fat tree topologies are currently the most dominant interconnect architecture in parallel computers. In the Top500 list of supercomputers published in November 2022 [14], five of the top ten fastest systems employ the fat-tree interconnection network topology. These include Summit, Sierra, Sunway TaihuLight, Selene, and Tianhe-2A [5, 5, 84, 85, 86].

2.1.3 Low Diameter/ASPL Network Topology

Dragonfly

The dragonfly network topology [87] is a high-performance interconnection network topology designed explicitly for large-scale parallel computers, such as supercomputers and data centers. High-radix routers in interconnection networks can reduce diameter and latency; however, they may require longer cables, increasing costs. The dragonfly

network topology was proposed to minimize the number of global channels and reduce costs while maintaining low diameter and latency in the network. This approach provides an efficient and cost-effective solution for large-scale parallel computers.

The dragonfly network topology consists of a hierarchical structure with local groups at the lowest level, each containing a fixed number of routers and PEs. Within a local group, routers are fully connected, providing low-latency communication. At a higher level, local groups are interconnected through global links, where each router in a local group connects to routers in other local groups. A formal description of the dragonfly network topology can be presented as follows:

- Network Terminals (PEs): Let N be the total number of network terminals or PEs in the Dragonfly network. We can calculate N using the formula $N = a \times \frac{p}{2}$, where a is the number of routers in each group, and p is the number of terminals connected to each router.

- Router radix: Let k be the radix of the routers, which is the total number of ports or links connected to each router. In the Dragonfly network topology, we can calculate the radix as $k = p + a$, since each router connects to p terminals and a routers in other local groups.

- Local groups: The Dragonfly network topology consists of multiple local groups, with each local group containing a routers. These routers are interconnected in a fully connected network topology within the group. Let G be the total number of local groups in the network.

- Global links: The effective radix of the group, represented by k', is the number of global links in the network connecting different local groups. We can calculate the effective radix as $k' = a \times \frac{p}{2}$, as each router has $\frac{p}{2}$ global links connecting to other local groups, and there are a routers in each group.

- System-level properties:

  - Total number of routers in the network: $R = a \times G$
  - Total number of PEs in the network: $P = N \times G$

– Total number of global links in the network: $L_g = \frac{a \times p \times G}{4}$ (each global link connects two routers)

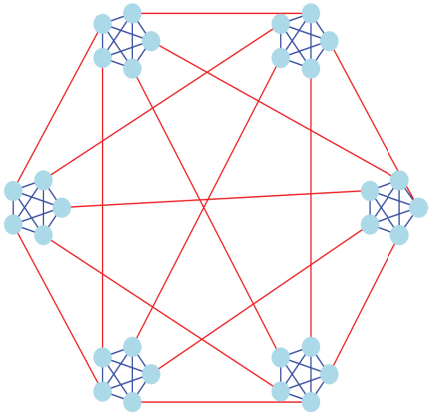


Figure 2.6: Dragonfly (p=4, h=1).

Figure 2.6 is an example of dragonfly (p=4, h=1). In the Top500 list of supercomputers published in November 2022 [14], four of the top ten fastest systems employ the dragonfly network topology or its variant dragonfly+ network topology [88]. These include Leonardo, Frontier, LUMI, and Perlmutter [89, 90, 91, 92].

Random Network Topology

Random shortcut network topologies are a type of network topology where nodes are primarily connected in a regular structure, such as a lattice or a ring, with additional random connections (shortcuts) between nodes. These shortcuts decrease the average path length between nodes, making the network more efficient in communication and information exchange.

Random network topologies are generated either as fully random graphs [93] or by adding random links to a baseline network topology [2] so that each node has the same degree. The random network topologies achieve low-diameter. and low ASPL [2]. The graph can correspond to the network topology. Given a (regular) graph with degree d and diameter k, the number of vertices in the graph is at most $1 + d \sum_{i=0}^{k-1} (d-1)^i$ and $\sum_{i=0}^{k} d^i$ for graphs, respectively, which is termed the Moore bound.

Only a few graphs achieve this bound [11]. While a random graph provides a low diameter and ASPL close to the lower bound.

In this dissertation, our one target is random network topology. In particular, we consider the random network topology generated as follows: consider a procedure to add links randomly by picking two nodes among the possible nodes that have lower links than the network degree. We enforce that all the vertices have the same degree as possible, as shown in Figure 2.7.

Random shortcut network topology is suitable for applications where efficient communication and robustness are essential, such as peer-to-peer networks, social networks, and distributed systems. It can also be used to model the structure of complex systems in areas like biology, physics, and sociology. However, the potential drawbacks of suboptimal shortcuts and increased routing complexity should be considered when implementing a random shortcut network topology.

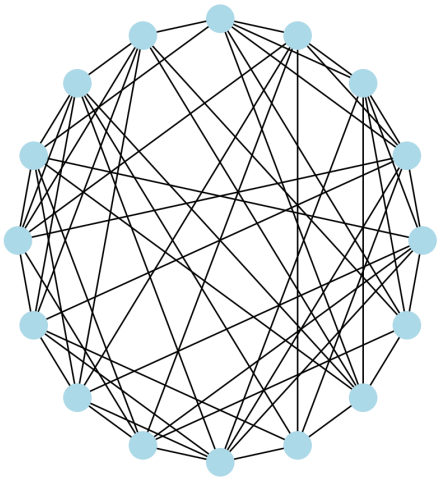


Figure 2.7: Random shortcut network topology.

Circulant Network Topology

Circulant network topology is widely used in various network designs. Its example is shown in Figure 2.8. They are used as a ring extension to design and implement local area networks. The networks are also called distributed loop computer-networks[49, 50]. A variant of circulant network topology called Multi-Ring network topology is used to

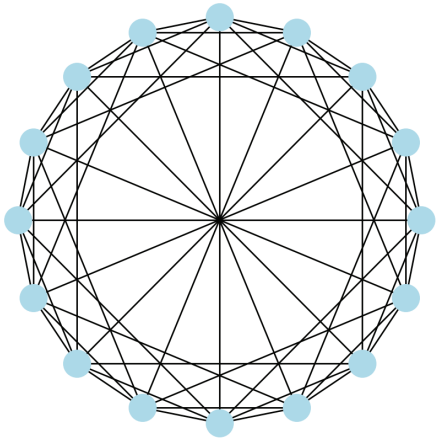


Figure 2.8: Circulant Network topology.

achieve high-performance group communication in peer-to-peer networks[94]. The paper [95] proposed recursive circulant network topology for multicomputer systems. The recursive circulant network topology is also circulant network topology. [96] proposed generalized recursive circulant graphs, which are the extension of recursive circulant graphs. Recently, [53] proposed the optimal circulant network topologies as low latency network topologies.

Figure 2.9 plots diameter and ASPL for different degrees for two 2^{15}-vertex topologies. Our finding is that random shortcuts achieve lower diameter and ASPL than non-random shortcuts (circulant network topology). In this context, we have little motivation to use circulant network topology. Nevertheless, it fits with some collective algorithms. We thus explore the two network topologies for collective communication.

2.2 Collective-Communication Operation

In this dissertation, we described a typical collective communication operation, our target.

2.2.1 Broadcast

A Broadcast operation is a communication pattern used in parallel computers where a single process or node sends the same data or message to all other processes or nodes

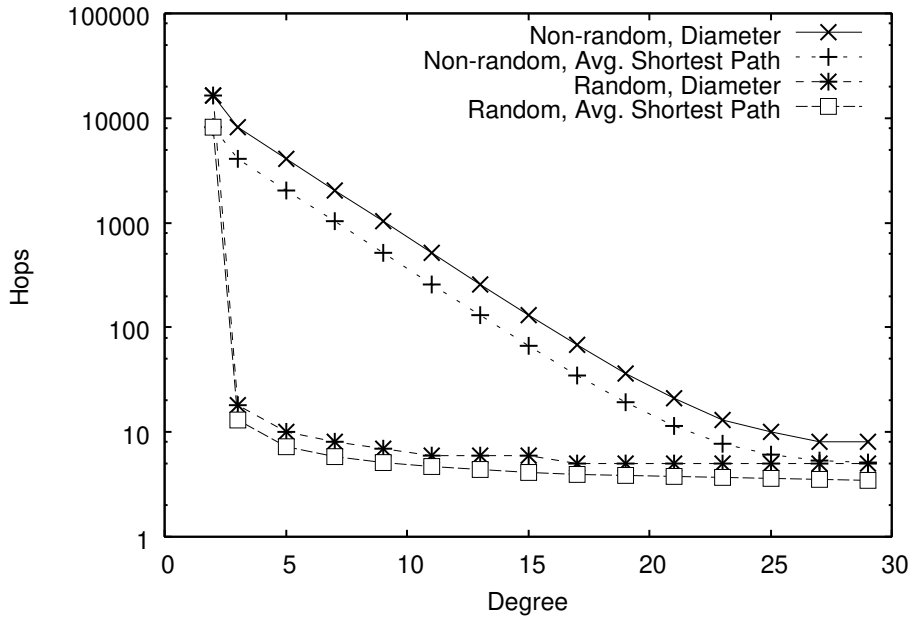


Figure 2.9: Diameter and average shortest path length vs. degree for a 2^{15}-node random shortcut and circulant (non-random shortcut) network topologies (picked up from [2]).

in a specified group. The primary purpose of Broadcast operations is to ensure that all processes receive the same information, which is often necessary for synchronization, distributing configuration data, initializing values, or distributing parts of a problem or dataset. Figure 2.10 is an example of Broadcast operation for four processes. Process #0 sends data 1 to processes #1,2,3.

In the Message Passing Interface (MPI) library context, the Broadcast operation is implemented as the MPI_Bcast function. This function is a collective communication operation involving all processes in a specified group or communicator. A communicator object in MPI defines the group of processes, with MPI_COMM_WORLD being the most common communicator that includes all processes in a parallel program.

Broadcast operations are essential in parallel computing, as they allow for efficient and organized communication between processes, ensuring that all processes have the necessary information to perform their tasks.

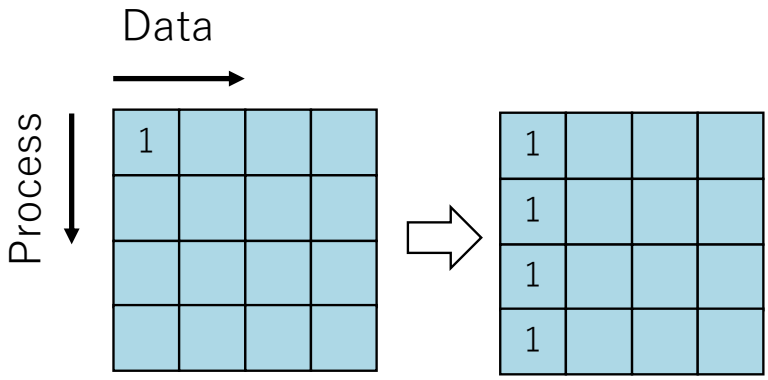


Figure 2.10: Broadcast operation.

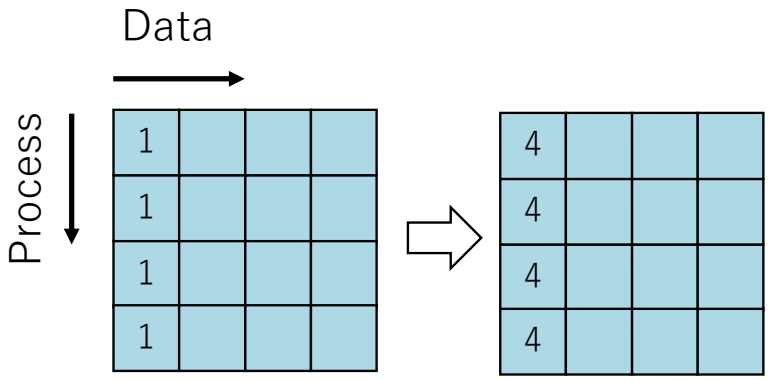


Figure 2.11: Allreduce operation.

2.2.2 Allreduce

The Allreduce operation is a collective communication operation in the Message Passing Interface (MPI) library used in parallel computing. It combines the functionality of the Reduce and Broadcast operations. The primary purpose of Allreduce is to perform a specified reduction operation (e.g., sum, product, maximum, or minimum) on data from all processes in a group or communicator and then distribute the result back to all processes in the group.

The Allreduce operation is functional when all processes need the aggregated result of some operation, such as computing a global sum, maximum or minimum value, or other collective statistics. Some common use cases include:

1. Calculating a global sum or mean in distributed data analysis.

2. Finding the global maximum or minimum value in optimization problems.

3. Computing a global norm in numerical linear algebra operations.

4. Synchronizing model parameters in distributed machine learning applications.

2.2.3 Alltoall

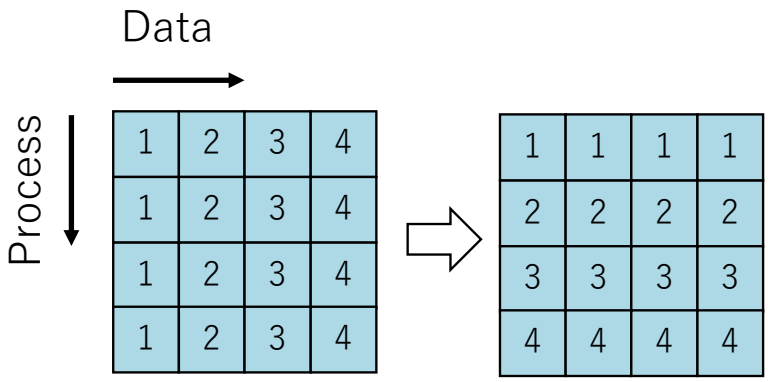


Figure 2.12: Alltoall operation.

The Alltoall operation is a collective communication operation in the Message Passing Interface (MPI) library used in parallel computing. Its primary purpose is to enable each process in a group or communicator to send distinct data segments to every other process in the group. This operation is beneficial when processes must exchange different data with all other processes, such as matrix transpositions, data redistribution, or complex communication patterns. Figure 2.12 illustrates the alltoall operation for four processes. Each process sends different data to different processes.

2.3 Collective-Communication Algorithm

2.3.1 Hardware-, path- and unicast-based Broadcast Techniques

Hardware-, path-, and unicast-based Broadcast techniques are typical methods for collective communications in interconnection networks [97, 98]. Hardware Broadcasts duplicate packets at an intermediate switch.

Since it reduces the total hop counts of packets in Broadcast, it efficiently sends data to multiple destinations. A path-based Broadcast sends data along a path that includes all destinations, and thus requires an efficient broadcast-path search, e.g., the Hamiltonian cycle for Broadcast.

However, current conventional network products, such as InfiniBand and Ethernet does not always support hardware- and path-based broadcast techniques. We target unicast-based collective communication in this dissertation.

2.3.2 Typical Collective-Communication Algorithms

We explain the typical collective communication algorithms, which is mainly discussed in this dissertation.

Binomial Tree Algorithm

Broadcast is a one-to-all communication pattern that a source process Broadcasts message to all processes in the same group. The binomial tree algorithm was widely used to implement the Broadcast operation due to its low steps, such as in MPICH2 and Open MPI [22] [21]. Assume that there are N compute nodes, and each computer node runs only a process. The binomial tree-based Broadcast will be completed by $\lceil \log_2 N \rceil$ steps.

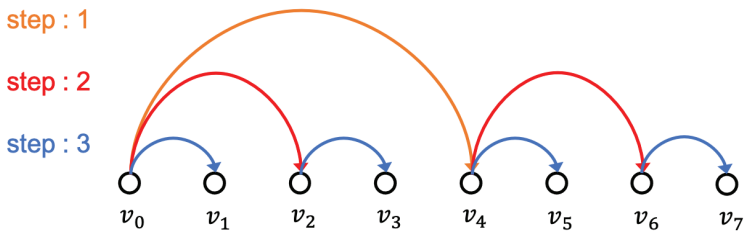


Figure 2.13: Broadcast using binomial tree algorithm.

Figure 2.13 shows an example of a binomial tree Broadcast with eight compute nodes. Assume that each node runs one process and v_0 is the root node; completing the Broadcast will take three steps. Mark $v_i \rightarrow v_j$ as that node v_i sends messages to node v_j. In the first step, only one unicast $v_0 \rightarrow v_4$ exists. In the second step, the v_0 and v_4 will send messages as the source nodes, and there are two unicasts $v_0 \rightarrow v_2$ and

$v_4 \rightarrow v_6$. In the third step, nodes v_0, v_2, v_4, v_6 will be the source node to send messages to the remaining nodes that have not yet received the message, there are four unicasts $v_0 \rightarrow v_1$, $v_2 \rightarrow v_3$, $v_4 \rightarrow v_5$ and $v_5 \rightarrow v_6$.

Recursive Doubling Algorithm

The recursive doubling algorithm is often used to implement the Allreduce operation, and it is easy to execute when the number of process N is a power of two. Here, we explain the case where the number of processes N is to the power of two. The recursive doubling algorithm takes $\log_2 N$ steps to complete the Allreduce operation.

Figure 2.14 shows recursive doubling Allreduce with eight computer nodes. Mark $v_i \leftrightarrow v_j$ as that node v_i and v_j exchange data with each other. In the first step, nodes that are a distance one apart exchange their data, so there are $v_0 \leftrightarrow v_1$, $v_2 \leftrightarrow v_3$, $v_4 \leftrightarrow v_5$ and $v_6 \leftrightarrow v_7$. In the second step, nodes that are a distance two apart exchange their data, so there are $v_0 \leftrightarrow v_2$, $v_1 \leftrightarrow v_3$, $v_4 \leftrightarrow v_6$ and $v_5 \leftrightarrow v_7$. In the third step, nodes that are a distance three apart exchange their data; there are $v_0 \leftrightarrow v_4$, $v_1 \leftrightarrow v_5$, $v_2 \leftrightarrow v_6$ and $v_3 \leftrightarrow v_7$.

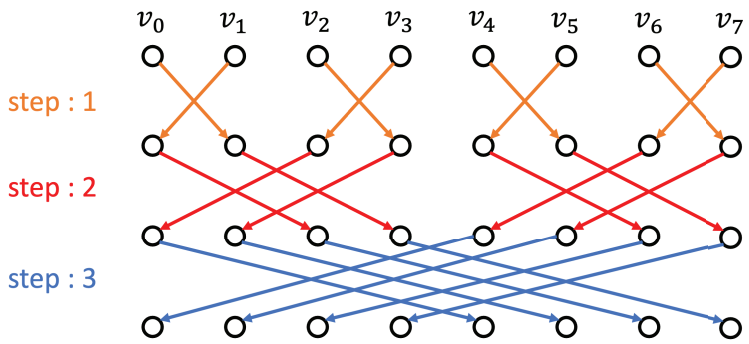


Figure 2.14: Allreduce the using recursive doubling algorithm.

Bruck's Algorithm

Bruck's algorithm is an efficient algorithm for Alltoall operation [99]. The Bruck's algorithm consists of three phases. In the first and the third phases, each process only needs to rearrange the data locally. The second phase performs inter-process communication. Here, we only briefly describe the second phase involving inter-process

communication. Assuming that there are N compute nodes and each compute node is running one process, Bruck's algorithm takes $\lceil \log_2 N \rceil$ steps to complete the Alltoall operation. In kth step($0 \leq k < \lceil \log_2 N \rceil$), compute node v_i sends to $v_{(i+2^k \mod N)}$ and receives from $v_{(i-2^k+N \mod N)}$[17].

Figure 2.15 shows Bruck's algorithm Alltoall with eight computer nodes. Assume that each compute node is running one process for ease of understanding. Mark $v_i \longrightarrow v_j$ as that node v_i sends messages to node v_j. In the first step, the set of unicast is $\{v_0 \longrightarrow v_7, v_1 \longrightarrow v_0\ , v_2 \longrightarrow v_1, v_3 \longrightarrow v_2, v_4 \longrightarrow v_3, v_5 \longrightarrow v_4, v_6 \longrightarrow v_5, v_7 \longrightarrow v_6\}$. In the second step, the set of unicast is $\{v_0 \longrightarrow v_6, v_1 \longrightarrow v_7\ , v_2 \longrightarrow v_0, v_3 \longrightarrow v_1, v_4 \longrightarrow v_2, v_5 \longrightarrow v_3, v_6 \longrightarrow v_4, v_7 \longrightarrow v_5\}$. In the third step, the set of unicast is $\{v_0 \longrightarrow v_4, v_1 \longrightarrow v_5\ , v_2 \longrightarrow v_6, v_3 \longrightarrow v_7, v_4 \longrightarrow v_0, v_5 \longrightarrow v_1, v_6 \longrightarrow v_2, v_7 \longrightarrow v_3\}$.

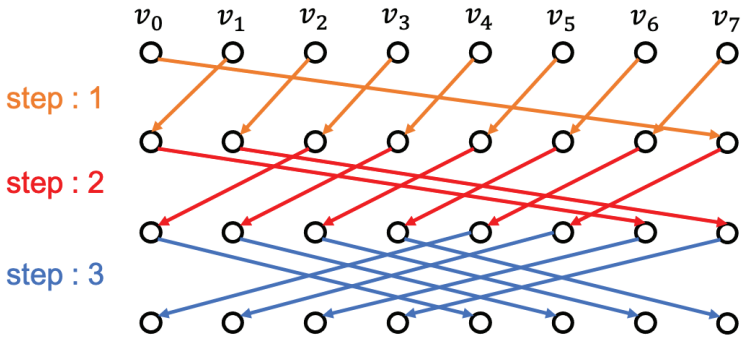


Figure 2.15: Alltoall using Bruck's algorithm.

-

2.4 Process Mapping

2.4.1 Topology-Aware Process Mapping

Process mapping, i.e., the process of placing the message-passing interface (MPI) ranks of a parallel program onto the compute nodes designated by the system software can effectively improve the access locality [44], thus obtaining high communication performance. Our survey is consistent with our prior works on high-radix parallel computers [44]. Application runtime reduction by selecting an appropriate process mapping can result in higher resource utilization.

Some prior process mapping works focused on the 3-D Torus BlueGene/L system [100] [101]. They assumed that nodes allocated for a job must be rectangular and contiguous.

However, systems implementing such approaches may lead to the fragmentation of free compute nodes. This could be low system utilization. The works [102, 103, 104]extended the traditional rectangular mapping approach to use node ordering to make contiguous allocations. These linear approaches have the advantage of fast allocations from their ordered list of free nodes.

Other approaches [105, 101, 106, 107] consider the communication pattern when using topology-embedding techniques [44]

The works [108] [109] studied the impact of simple linear node mapping on the performance of mini-applications on different Fat-tree configuration systems. Similar to that in Torus, node ordering [110] is applied to the fat-tree network topology for job allocation. The works [111] [112] implemented a topology-aware node allocation policy that allocates isolated partitions to jobs to eliminate inter-job interference on a Fat-tree network.

The HyperX network topology was proposed and compared with the fat-tree network topology, where a simple random node mapping is used. This is because topology-aware mapping is impractical in production environments due to the limited availability of idle resources.

Process mapping is also considered for Dragonfly network topology [87]. The work [113] analyzed Cartesian multi-dimensional nearest neighbor exchanges and showed that random process mapping with direct routing is consistently outperformed by Cartesian process mapping with indirect routing. The work [114] showed that the impact of process mapping is minimal by using small-scale experiments (up to 256 nodes) on a Cray XC30 system.

The work [115] compared Torus, Fat-tree, and Dragonfly using linear and random process mapping schemes. It showed that different mapping schemes lead to similar performance for a single job executions on all networks. For multi-job workloads with a few large jobs, the Torus network consistently achieves the best performance, with the Fat-tree network performance slightly worse.

2.4.2 Topology-Agnostic Process Mapping

The study in [97] showed that it is impossible to avoid packet contentions on arbitrary network topologies, and it proposed a multicast algorithm called chain concatenation order (CCO) to reduce the number of packet contentions. The CCO algorithm first constructs a breadth-first search (BFS) tree with a source node as root, and it visits a destination node with an order based on the BFS tree. The CCO is our competitor in multicast on random network topologies. The concept of CCO can be generalized to process mapping to minimize the number of packet contentions between processes.

There are also many optimization methods of collective-communication operations for heterogeneous HPC platforms. The study in [116, 117] proposed a topology-aware algorithm to improve the performance of collective communication operations for large-scale InfiniBand clusters. An optimization of collective-communication operation was proposed by considering message length or number of nodes in widespread MPI implementation MPICH2 [22], or by feeling message length [116]. However, their existing optimization is not well optimized for random network topologies regarding process mapping total path hops of unicasts that form a collective communication operation.

3

Efficient Collective Communication using Random Network Topology

3.1 Introduction

As described in Chapter 2, collective communication operations significantly impact the performance of applications running on parallel computers. Implementing efficient collective communication operations has high implications for improving the performance of applications.

As shown in Figure 2.1 of Chapter 2, the performance of collective communication is affected by several factors, such as the network topology of parallel computers, the job/process mapping strategy for allocating computational resources like compute nodes, and the algorithm for implementing collective communication operations. This chapter will consider job/process mapping and collective algorithms to achieve efficient collective communication on random shortcut network topology.

This chapter uses random shortcut topologies to build an interconnected network

for parallel computers. A low-diameter network topology is critical to the performance of a parallel computer. Figure 2.9 shows that adding random shortcuts to a regular network topology can cause the diameter and ASPL of the network topology to drop rapidly and that a random shortcut network topology with a low diameter and ASPL allows messages to pass through fewer hops from the source node to the destination node, thus satisfying the low latency requirement. In this chapter, we use the ring-based shortcut network topology, where the ring serves as the base network topology. Then, we keep adding shortcuts to the ring to build a low diameter and ASPL network topology [2].

Job/Process mapping allocates the compute nodes and processors to parallel applications. The mapping strategies also affect the performance of parallel applications. The choice of mapping strategy is often related to the topology and application characteristics [105, 106, 118, 119, 120]. We picked up three different process mapping strategies based on the characteristics of ring-based random shortcut network topology: random mapping, hierarchical tree mapping, and ring-based consecutive mapping. In the following section 3.2, we will introduce these three process mapping strategies as the baseline.

In this chapter, we primarily focus on three collective communication operations: Broadcast, Allreduce, and Alltoall. These operations are the most commonly used collective communication patterns [16], extensively employed in various parallel computing scenarios. The Binomial tree algorithm, recursive doubling algorithm, and Bruck's algorithm [99] are the classic implementations for Broadcast, Allreduce, and Alltoall operations. Many MPI libraries adopt these three algorithms to realize the corresponding collective communication operations [22, 21, 23].

To achieve efficient Broadcast, Allreduce, and Alltoall operations in the context of given network topology, process mapping strategy, and collective operation algorithms, we propose the use of the two-opt method [121] to reduce the overall hop count of collective communications [3]. By minimizing the communication overhead in collective communication operations. We aim to enhance the performance of parallel applications.

In the subsequent sections of this chapter, Section 3.2 introduces the three process mapping strategies in random shortcut network topology, including random mapping, hierarchical tree mapping, and ring-based consecutive mapping. Section 3.3 elucidates

the approach of employing the two-opt method to minimize the overall hop count in collective communication operations and examines the extent to which the two-opt method can reduce the number of hops in the rank placement. Section 3.4 shows the evaluation of the performance of collective communication operations and parallel applications with the two-opt rank placement onto the random shortcut network topology. Section 3.5 discusses the results of quality and scalability of the two-opt rank replacement method. Section 3.6 summarizes the contents of this chapter.

3.2 Process Mapping in Random Network Topology

The selection of an appropriate process mapping strategy frequently plays a pivotal role in determining the performance of parallel-computing applications. The selection of a process mapping policy is usually related to many factors, such as the network topology and type of applications. Many process mapping studies are for specific network topologies such as torus, fat-tree, and dragonfly [39, 122, 45, 120]. In the context of random shortcut network topology, we take random mapping, hierarchical-tree mapping, and ring-based consecutive mapping as our base process mapping strategies, considering the unique attributes of the random shortcut network topology. In the subsequent section, we describe these three mapping methodologies.

3.2.1 Random Mapping

In contrast to regular topologies like the torus, mapping a regular embedded topology onto a random topology poses a significant challenge, primarily due to the stochastic nature of the connections. The small-world effect [9] characterizes random shortcut topologies, leading to a low diameter and ASPL (Average Shortest Path Length), signifying that any two points within the network are relatively close. Consequently, a natural approach for random shortcut topologies involves random mapping, indiscriminately assigning tasks to a designated number of computational nodes without accounting for geometric characteristics.

Figure 3.1 illustrates a schematic of random mapping where a task is arbitrarily assigned to 8 compute nodes in a 32-node random shortcut network topology. The random mapping approach does not guarantee the continuity of the physical locations

of the assigned compute nodes. Although random shortcuts interconnect these nodes, their physical locations may be far apart. The low diameter and ASPL features allow these compute nodes to communicate with very few hops, even when they are physically far apart [2]. In addition, the random mapping strategy dramatically improves the scheduling performance of tasks [123].

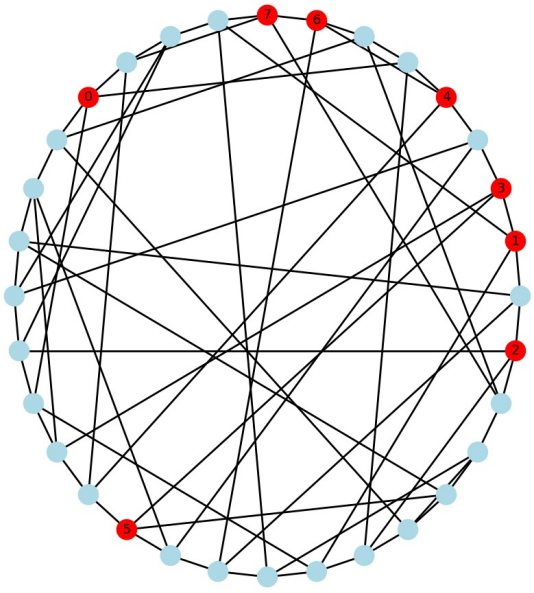
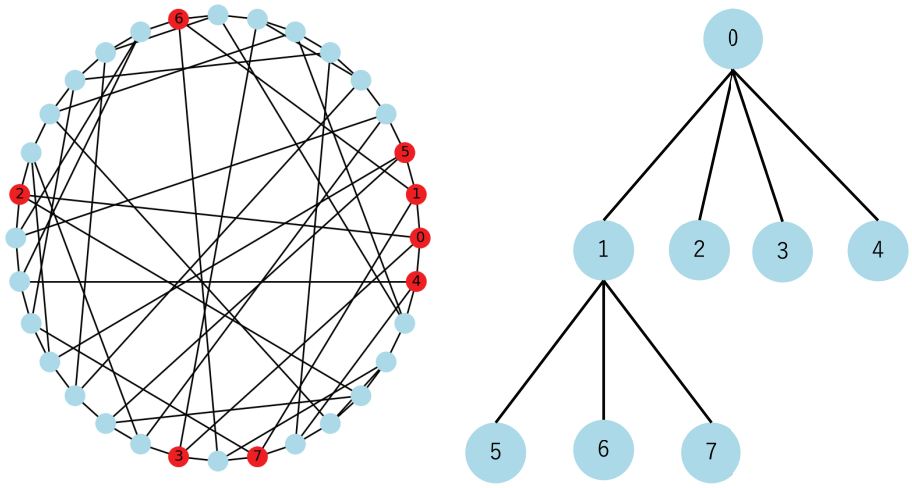


Figure 3.1: An 8-node random process mapping on a 32-node random shortcut network topology.

3.2.2 Hierarchical-Tree Mapping



(a) Hierarchical tree mapping (b) Spanning tree.

Figure 3.2: An 8-node hierarchical tree mapping on a 32-node random shortcut network topology.

Due to the small-world effect, random shortcut network topologies exhibit a reduced diameter and ASPL. Intuitively, one might expect an embedded topology employing random mapping to demonstrate a lower diameter and ASPL similarly. Nevertheless, the inherent randomness of the random mapping approach precludes any guarantee of consistently achieving the lowest possible diameter and ASPL in the resulting embedded topology. To produce embedded topologies that exhibit low diameter and ASPL, hierarchical-tree mapping approaches were proposed for random topologies [124].

Figure 3.2(a) illustrates a schematic of hierarchical-tree mapping where a task is assigned to 8 compute nodes in a 32-node random shortcut network topology. In the case of figure 3.2(a), first, select an available compute node as the tree's root. Then, a breadth-first search is initiated from the root, during which all available neighboring nodes of the root are appended as its leaf nodes. This process is repeated for each leaf node until the tree reaches the requisite number of compute nodes for the task. Finally, as figure 3.2(b) shows, a spanning tree was generated. Obviously, the diameter of the embedded tree topology is the depth of the spanning tree.

Generally speaking, when considering collective communication, tree structure is

efficient. Indeed, the BlueGene/L supercomputer provides a tree network for Broadcast and a torus network for stencil communication [125]. In this context, we pick up hierarchical tree process mapping embedded in a random network topology. In contrast to random mapping in Section 3.2.1, hierarchical tree mapping guarantees an upper bound on the distance between any two nodes in the task-assigned compute nodes.

3.2.3 Ring-Based Consecutive Mapping

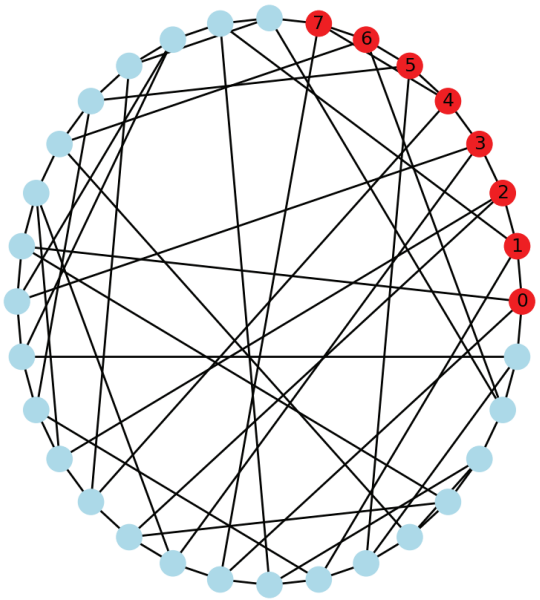
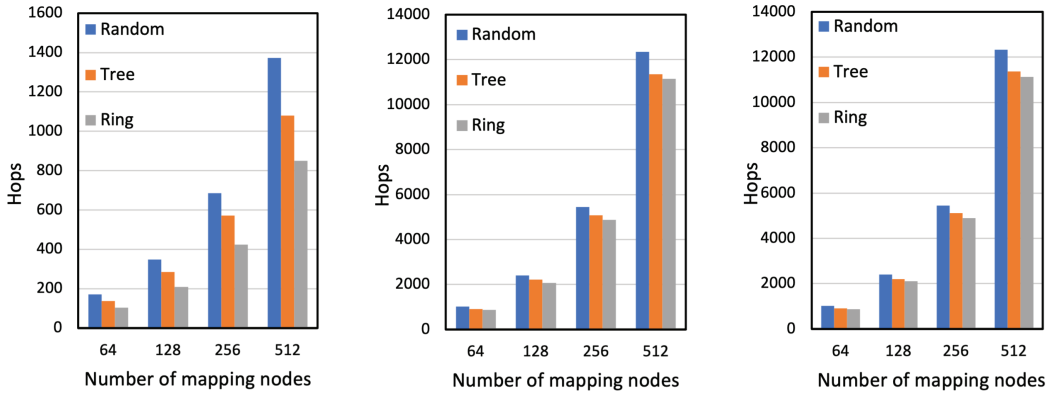


Figure 3.3: An 8-node ring-based consecutive mapping on a 32-node random shortcut network topology.

In some regular topologies, such as mesh and torus, continuous mapping strategies are employed to map tasks into regular topological structures. For instance, in mesh and torus topologies, tasks are organized into sub-meshes and sub-torus [126, 127, 128]. In an entirely random network topology, it is challenging to map tasks to a regular topology due to the uncertainty of node connections. In this study, the random topology we use is generated by randomly adding shortcuts to a ring. Based on this feature, we proposed using a ring-based continuous mapping strategy that arranges the mapped nodes along the ring, and nodes form a line along the ring. Figure 3.3 illustrates a schematic of ring-based consecutive mapping where a task is assigned to 8

compute nodes in a 32-node random shortcut network topology. This mapping is particularly suitable for applications where nodes are neighbors. In addition, shortcuts ensure that even mapped nodes far apart along a line on the ring, such as the head and tail nodes of the line, can communicate quickly via shortcuts. For example, in figure 3.3, the shortest distance between node 0 and node 7 is 2.

3.2.4 Hop Count Analysis of Process Mapping Strategies



(a) Hops of Broadcast. (b) Hops of Allreduce. (c) Hops of Alltoall.

Figure 3.4: Collective communication hops of three mapping strategies on random network topology.

We evaluated and compared the hop counts of different collective communication operations under three mapping strategies: random, hierarchical-tree, and ring-based mapping, which will help us understand the hop counts of varying mapping strategies without any optimization.

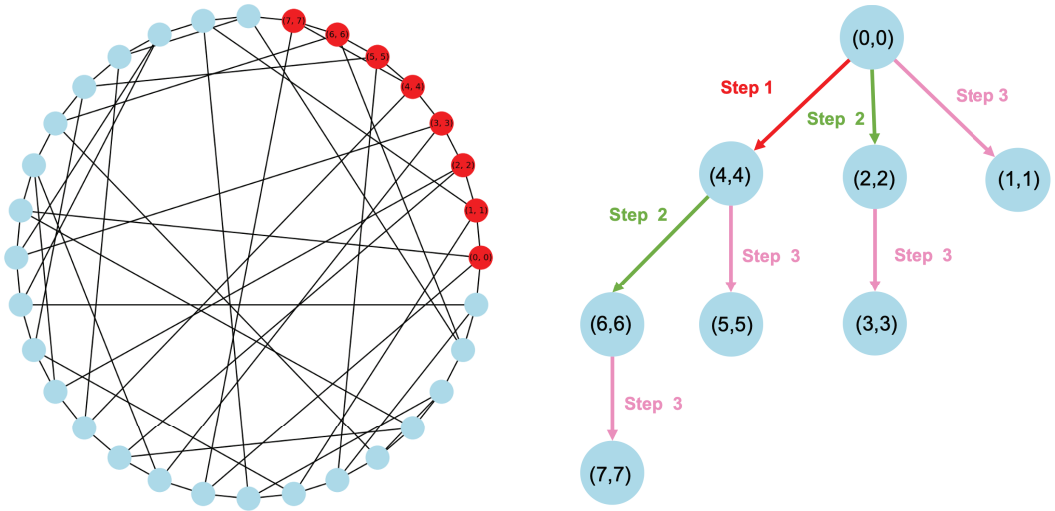
Figures 3.4(a), 3.4(b) and 3.4(c) represent the hop counts of random, hierarchical-tree and ring-based mapping strategies under Broadcast, Allreduce and Alltoall operations, respectively. The X-axis represents the number of mapped nodes, and the Y-axis represents the number of hops. In this evaluation, the total number of nodes in the interconnection network is 1,024. As shown in the figures, the ring-based mapping has the most minor hops. Hierarchical tree mapping is the next.

The number of hops for the Broadcast operation is small compared to that for Allreduce and Alltoall operations.

3.3 Rank Placement

3.3.1 Ascending-Order Rank Placement

Once a process mapping is completed, each process is mapped onto a compute node. There is room to improve the performance of the collective communication operations by further updating the mapping between the process and compute node. In MPI, each process is identified by rank. We call process rank later. The most straightforward manner is ascending-order rank placement, in which a lower-rank process is mapped onto the lowest ID compute node among free compute nodes of the task.



(a) Ring-based consecutive mapping.

(b) Binomial tree Broadcast.

Figure 3.5: Ascending-order rank placement in Broadcast operation.

We let (a, b) to denote a pair of a computer node's ID and the process rank running on it, a denotes the ID of the compute node, and b denotes the process rank. Figure 3.5(a) represents an example of the ascending-order rank placement of ring-based consecutive process mapping of broadcast operation on a 32-node random shortcut topology. Compute nodes with IDs 0 to 7 participate in the Broadcast operation. In this example, the rank of the process on each compute node is the same as the ID of that compute node.

Figure 3.5(b) shows a binomial tree of Broadcast operations containing the eight

Step	Communication	Hops per commun.	Hops per step	Total hops
1	(0,0) → (4,4)	3	3	11
2	(0,0) → (2,2)	2	4	
	(4,4) → (6,6)	2		
3	(2,2) → (3,3)	1	4	
	(4,4) → (5,5)	1		
	(5,5) → (6,6)	1		
	(6,6) → (7,7)	1		

Table 3.1: Communication steps and hop counts of ascending-order rank placement in Broadcast.

processes. The binomial tree of Broadcast is generated based on the order of process rank. There is an ordered list of process's rank $< 0, 1, 2, 3, 4, 5, 6, 7 >$; this ordered list generates the binomial tree. Process 0 sends a message to process 4 in step 1. Processes 0 and 4 send a message to processes 2 and 6 in step 2, respectively. Finally, processes 0, 2, 4, and 6 send a message to processes 1, 3, 5, and 7, respectively.

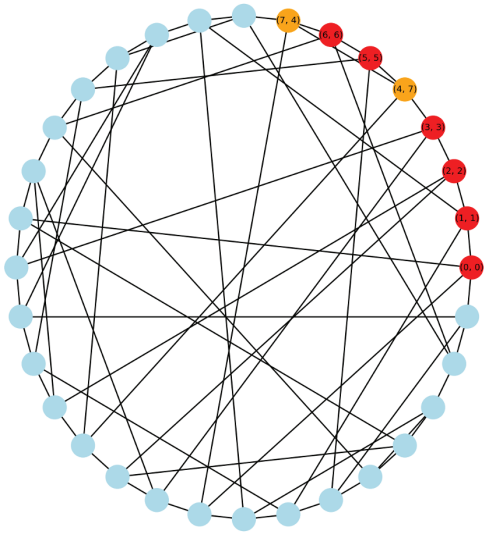
Table 3.1 shows the hops of each communication, the step and total hops of the Broadcast operation of the above case.

3.3.2 Two-opt Rank Placement

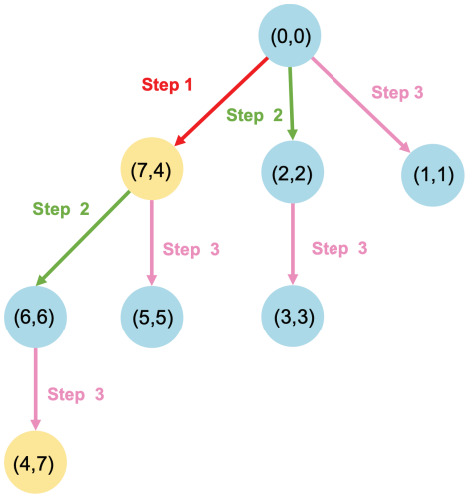
We attempt to improve the rank placement regarding hop counts or several contentions. The two-opt algorithm was first proposed in 1958 to solve the traveling salesman problem [121]. We apply the two-opt algorithm for rank placement of collective communication on random network topologies.

Step	Communication	Hops per commun.	Hops per step	Total hops
1	(0,0) → (7,4)	2	2	9
2	(0,0) → (2,2)	2	3	
	(7,4) → (6,6)	1		
3	(0,0) → (1,1)	1	4	
	(2,2) → (3,3)	1		
	(7,4) → (5,5)	1		
	(6,6) → (4,7)	1		

Table 3.2: Communication steps and hop counts of the two-opt rank placement in Broadcast.



(a) Swap operation: swap two nodes's ranks.



(b) Binomial tree Broadcast.

Figure 3.6: The two-opt Broadcast.

We previously introduced the ascending-order rank placement to assign the process's rank. We optimize the ascending-order rank placement by the two-opt method to reduce the total hops of the collective communication operations. We expect that minimizing the total hop counts leads to the goal of improving the performance of collective communication operations. Figure 3.6 shows how the two-opt method reduces the total number of broadcast operations. Like Figure 3.5, the random shortcut network topology has 32 compute nodes. Compute nodes with IDs 0 to 7 participate in the Broadcast operation.

The application of the two-opt method to the Broadcast is as follows. First, assign ranks to all processes using the ascending-order rank-placement strategy. In the initialization, each process has the same rank number as the compute node ID, then calculates the total hops for the Broadcast. The total hops are 11, as illustrated in Table 3.1. Next, perform the two-opt optimization process. Figure 3.6 shows how the two-opt method reduces the total number of Broadcast operations; (1) randomly select two compute nodes among all compute nodes involved in the broadcast operation, then (2) swap the process ranks running in these two compute nodes. In Figure 3.6, the compute nodes with ID 4 and 7 were selected. After this swap operation, the process

rank of compute node 4 becomes 7 and the process rank of compute node 7 becomes 4. Figure 3.6(b) shows the binomial tree after the swap of process ranks. Then, the total hops of Broadcast were re-calculated.

Table 3.2 shows the hops of each communication, the step, and total hops of the Broadcast operation of the new binomial tree broadcast. After the process-rank swap operation, the total hops decrease. That means the two-opt swap reduces the average shortest path length (ASPL) of the Broadcast operation. Then, repeat the swap of process ranks of two randomly selected compute nodes. If the total hops are reduced, accept the swap operation. Otherwise, cancel the swap operation. Repeat the procedure until the total path hops for the Broadcast operation cannot be smaller in the moderate number of attempts.

There is a variation of the two-opt Broadcast. Although it is designed to reduce the total path hops of point-to-point communications, the two-opt rank placement can be updated to reduce the number of possible contentions rather than reducing the total path hops. That is, instead of minimizing the total path hops, an alternative objective function to optimize the Broadcast can be to reduce the total number of possible contentions of the corresponding point-to-point communications on a link.

The two-opt algorithm can be applied for the other collective communication operations, i.e., Allreduce and Allgather. The ordered list of visiting nodes can express the other collective communication operations based on the recursive doubling algorithm. Since the two-opt method creates an ordered list of visiting nodes for its optimization, we can generate the two-opt collective communication operations with a similar procedure to that described above.

3.3.3 Hop Count Analysis of Two-Opt Rank Placement

We evaluated and compared the hop counts of collective communication operations using the two-opt rank replacement with different process mapping strategies, such as ring, random, and hierarchical mapping on the random network topology. In this evaluation, the degree of random shortcut network is 19; the network has 1024 compute nodes, 512 compute nodes were involved in the collective communication. Each compute node runs one process. The collective-communication algorithms for Broadcast, Allreduce, and Alltoall operations are binomial tree, recursive doubling, and

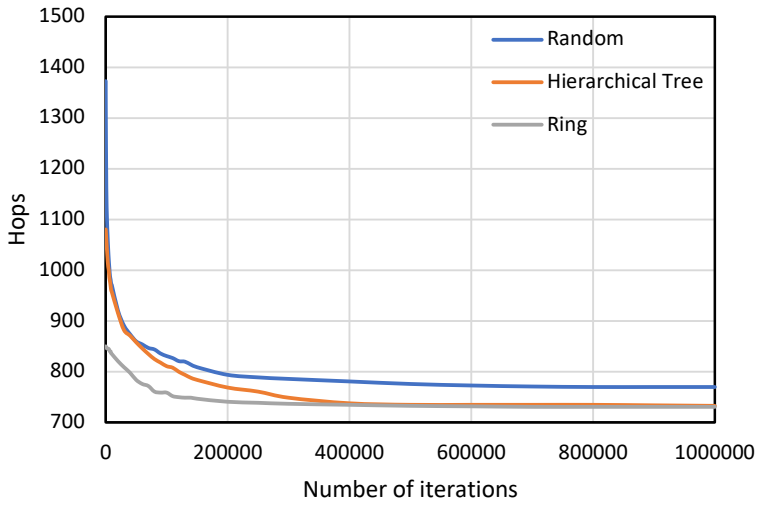


Figure 3.7: Hops of the two-opt rank placement for Broadcast in random network topology.

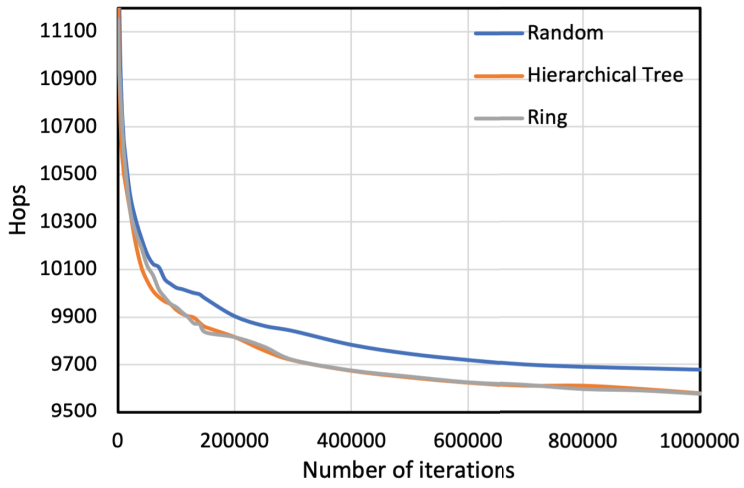


Figure 3.8: Hops of the two-opt rank placement for Allreduce in random network topology.

Bruck's ones, respectively.

Figures 3.7, 3.8, and 3.9 represent the variation of the total hops for the Broadcast, Allreduce, and Alltoall operations with the number of iterations of the two-opt method,

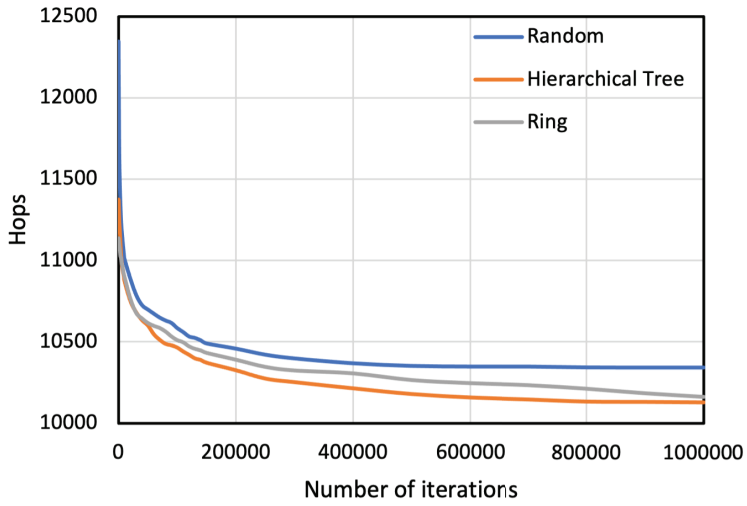


Figure 3.9: Hops of the two-opt rank placement for Alltoall in random network topology.

respectively. The X-axis represents the number of iterations of the two-opt optimization. The Y-axis represents the total hops of the collective communication operations. The results show a significant decrease in the total hops of the collective communication operations as the number of iterations of the two-opt operation increases. The total hop counts of the collective communication operations tend to stabilize gradually when the number of iterations increases.

After a sufficiently large number of iterations, up to 1,000,000 in this evaluation, Figure 3.7 shows that the two-opt rank placement reduced up to 32%, 44%, and 14% hops for random, hierarchical tree, and ring process mapping strategies, respectively. For the Allreduce operation, as illustrated in Figure 3.8, the two-opt rank placement can reduce 21%, 16%, and 11% hops for random, hierarchical tree, and ring process mapping strategies, respectively. For Alltoall operation, as illustrated in Figure 3.9, the two-opt rank placement can reduce about 16%, 11%, and 9% hops for random, hierarchical tree, and ring process mapping strategies, respectively.

The initial hops of different process mapping strategies affect the optimization space of two-opt replacements.

3.4 Evaluation

3.4.1 Methodology

We employ the SimGrid simulation framework (v3.28) [129] to analyze the performance of collective communication operations and parallel applications on large-scale parallel computers.

When generating a ring-based random shortcut network topology, it is essential to ensure all nodes have the same degree when adding shortcuts. In this study, we try to make all nodes have the same degree whenever possible. In our evaluation, the number of total compute nodes is 1024, and the degree of each compute node is 19.

Table 3.3 shows the interconnection network parameters in SimGrid. Switches are interconnected to form an interconnected network, with one compute node per switch. In this study, we leverage the built-in routing algorithm in the SimGrid simulation framework, which employs the Floyd algorithm to determine the shortest path between nodes in terms of hops.

Table 3.3: Parameters of the interconnection network.

Power of compute node	100GFLOPS
Switch latency	100ns
Link bandwidth	100Gbps
Switch capacity	3.6Tbps
Routing algorithm	Floyd (Minimal Paths)

The ultimate evaluation is to attempt the combination of job mapping and rank placement for better performance of collective communications on random network topology. We picked up the random, hierarchical tree, and ring task mappings while we selected ascending order (default), CCO(chain concatenation ordering) [97], two-opt for hop counts (two-opt-hops), and two opt for contentions (two-opt-contention). The ascending order and CCO are our competitors.

3.4.2 Execution Time of Collective Communication Operations

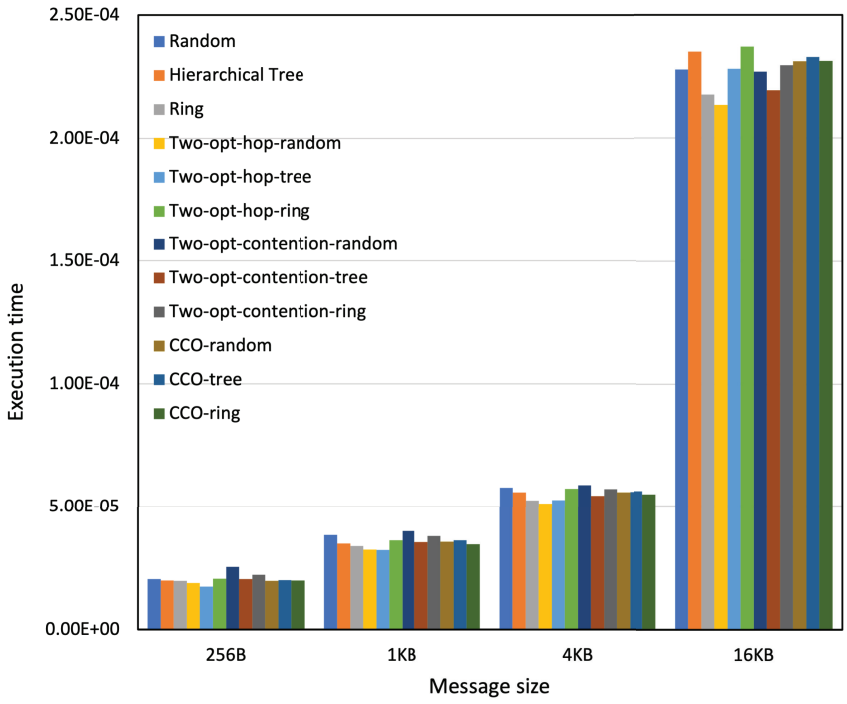
We preliminarily illustrated the hop counts of different mapping strategies with rank placement optimizations. Collective communication operations are engaged across 512

processes, corresponding to an equivalent number of compute nodes. The iteration times for the two-opt optimization are set to 200,000. Execution time is the time it takes to repeat the collective operation ten times.

Table 3.4 illustrates their hop counts of Broadcast operation. The Broadcast hop counts can be reduced to 42%, 29%, and 13% for the random, hierarchical tree, and ring mapping strategies by the two-opt rank placement.

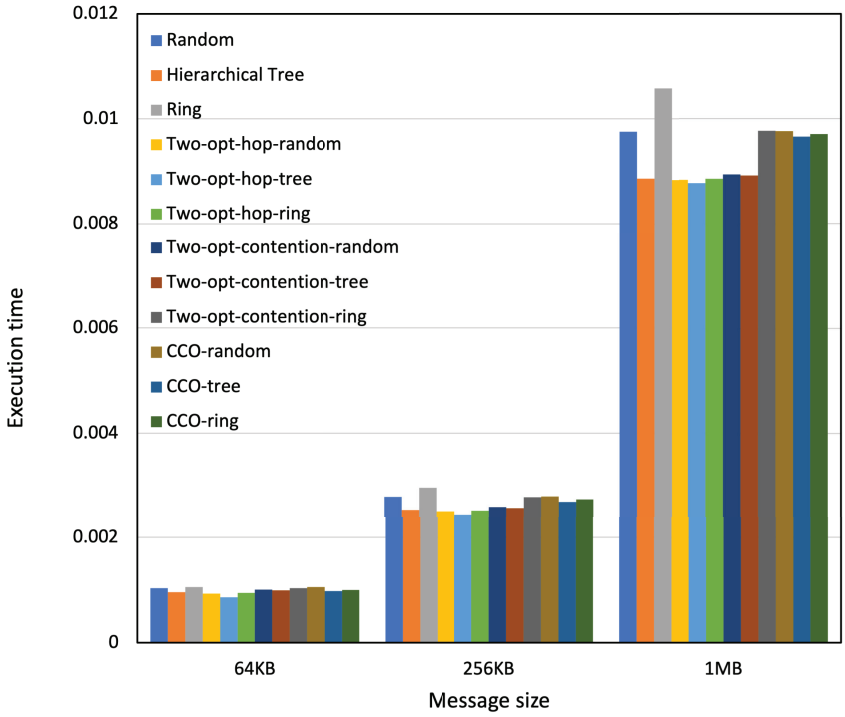
Table 3.4: Hop counts of different rank placements on different process mapping strategies for Broadcast (1,024 nodes, 512 processes).

	Ascending Order	Two-opt
Random	1,373	794
Hierarchical Tree	1,080	769
Ring	850	741



(a) Execution time of Broadcast with small message size.

Figures 3.10(a) and 3.10(b) present the execution time for the Broadcast operation in two different scenarios: small message sizes ranging from 256B to 16KB and large



(b) Execution time of Broadcast with large message size.

Figure 3.10: Execution time of Broadcast operation in random network topology (1,024 nodes, 512 processes).

message sizes ranging from 64KB to 1MB, respectively. These results enable us to analyze the performance of the Broadcast operation under varying message size conditions. The lower values are better on the Y-axis (the unit is second). Our main finding is that the two-opt method for minimizing the hop count with random process mapping is usually best. This is because the total hop counts highly affect the execution time, especially for short messages. Since random network topologies have many shortcuts, the impact of process mapping on the execution time is not significant.

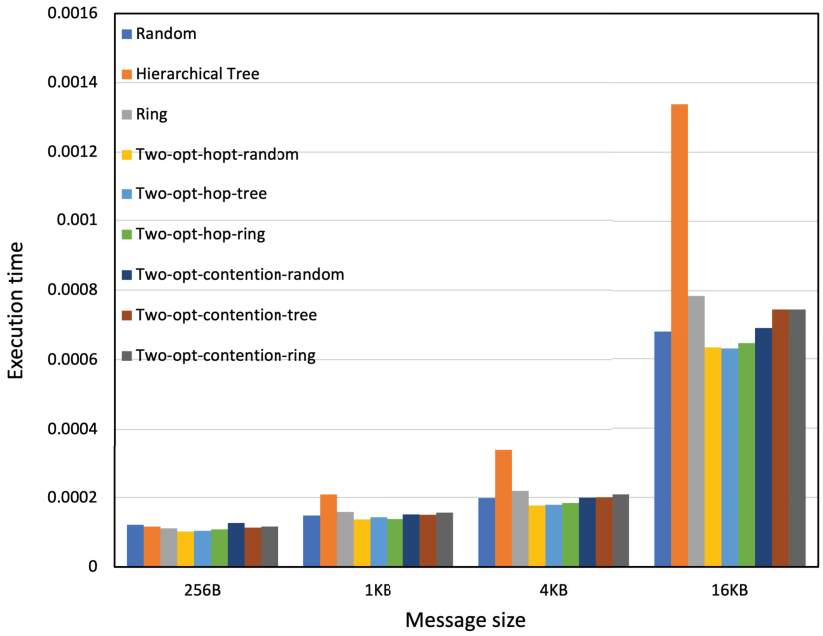
The CCO algorithm is expected to have a better performance due to the small possibility of the massage contentions. However, the two-opt methods usually outperform them. We can say that the optimization only to the messages generated in the collective operation, the two-opt method, works efficiently.

Analytical results of Allreduce operations are illustrated in Table 3.5. The hop

counts of the Allreduce can be reduced to 20%, 14%, and 12% for the random, hierarchical tree, and ring mapping strategies by the two-opt rank placement.

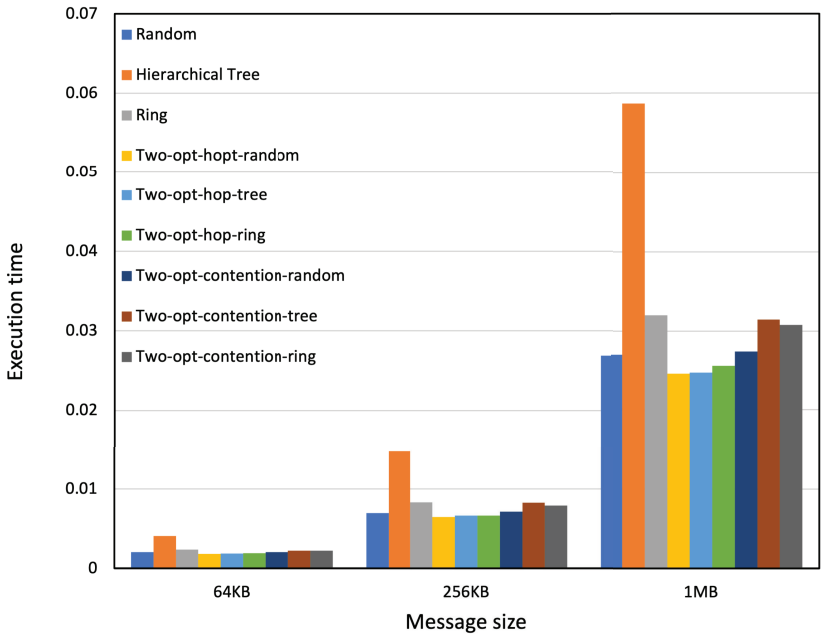
Table 3.5: Hop counts of different rank placements on different process mapping strategies for Allreduce (1,024 nodes, 512 processes).

	Ascending-Order	Two-opt
Random	12,308	9,904
Hierarchical Tree	11,362	9,816
Ring	11,152	9,816



(a) Execution time of Allreduce with small message size.

Figures 3.11(a) and 3.11(b) present the execution time for the Allreduce operation in two different scenarios: small message sizes ranging from 256B to 16KB and large message sizes ranging from 64KB to 1MB, respectively. These results enable us to analyze the performance of the Allreduce operation under varying message size conditions.



(b) Execution time of Allreduce with large message size.

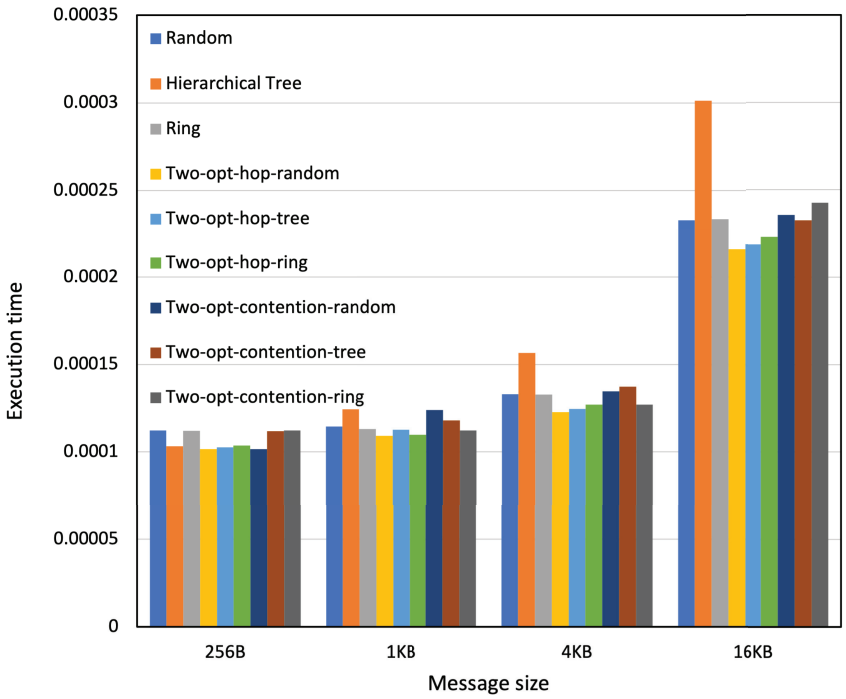
Figure 3.11: Execution time of an Allreduce operation in random network topology (1,024 nodes, 512 processes).

Figures 3.11(a) and 3.11(b) illustrate that the same performance trend to the Broadcast is obtained; the two-opt method for minimizing hop counts is better with random process mapping. The reason why the two-opt method works well is tuning only for the messages generated in the collective operation.

Table 3.6: Hop counts of different rank placements on different process mapping strategies for Alltoall (1,024 nodes, 512 processes).

	Ascending-Order	Two-opt
Random	12347	10459
Hierarchical Tree	11373	10326
Ring	11135	10389

Table 3.6 illustrates their hop counts of Alltoall operation. The hop counts of the Alltoall can be reduced to 15%, 9%, and 7% for the random, hierarchical tree, and ring



(a) Execution time of Alltoall with small message size.

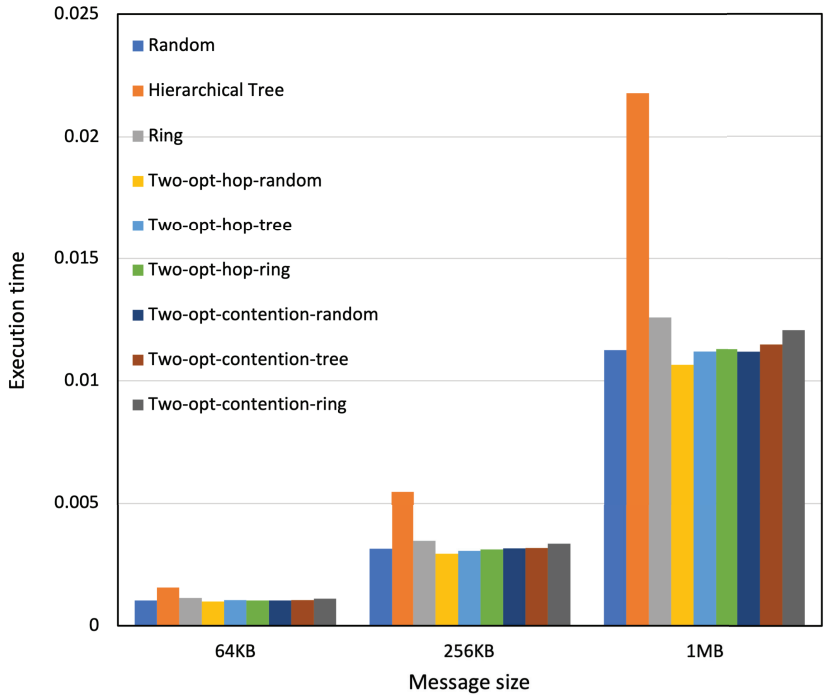
mapping strategies by the two-opt rank placement.

Figures 3.12(a) and 3.12(b) present the execution time for the Alltoall operation in two different scenarios: small message sizes ranging from 256B to 16KB and large message sizes ranging from 64KB to 1MB, respectively. The performance tendency is similar to those in Allreduce and Broadcast; the two-opt method for minimizing hop counts in random process mapping is usually better in various message sizes.

3.4.3 Performance Evaluation of Parallel Applications

In Section 3.4.2, we evaluate the performance of collective communication operations, and the evaluation results show that the two-opt rank placement optimization dramatically reduces the hop counts and improves the performance of collective communication operations.

In this section, we would like to evaluate the performance gain of two-opt rank placement optimization for parallel applications. We evaluated some of the applications



(b) Execution time of Alltoall with large message size.

Figure 3.12: Execution time of Alltoall operation in random network topology (1,024 nodes, 512 processes).

in the NAS parallel benchmark [130]: FT, IS, MG, and LU. We set the problem size of these applications as Class A and execute applications with 128 processes, i.e., 128 compute nodes. We choose a specific collective communication operation for rank placement two-opt optimization based on the communication characteristics of the applications. Each application's most used collective communication operation is chosen, based on the demonstrated in [131]. Since the Broadcast operation has less running time than Allreduce and Alltoall, we optimize two-opt rank placement by considering only Allreduce and Alltoall operations. The iteration times of two-opt are 20,0000, and the algorithms for Allreduce and Alltoall are recursive doubling and Bruck's [99], respectively. The elemental mapping strategies are random, hierarchical tree, and ring-based consecutive mapping.

Figures 3.13 to 3.16 display box plots of parallel applications FT, IS, MG, and

Table 3.7: Applications Description and Collective Communication Operations for two-opt optimization

Application	Description	Collective Operation
FT	discrete 3D fast Fourier Transform	Alltoall
IS	Integer Sort	Alltoall, Alltoallv
MG	Multigrid	Allreduce
LU	Lower-Upper Gauss-Seidel solver	Allreduce

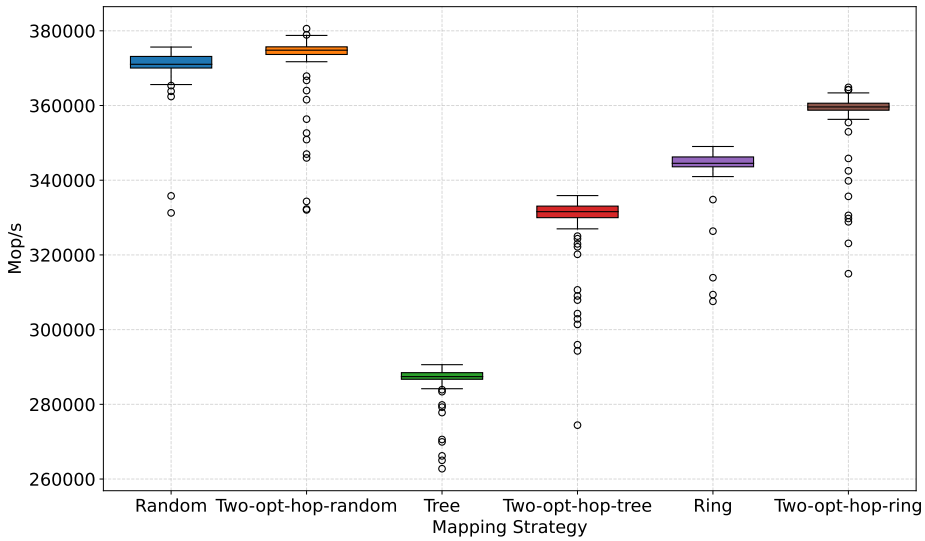


Figure 3.13: Performance evaluation of application FT in random shortcut network topology.

LU performance. The x-axis shows Mop/s, with higher values indicating better performance, and the y-axis shows different mapping strategies. Each figure consists of six data sets for random, hierarchical tree, and ring-based consecutive mapping, each in versions with and without two-opt optimization, and each set includes 100 data points. In these box plots, the median appears as a horizontal line inside each box, while the lower and upper quartiles form the bottom and top of the box. The "whiskers" extend from each box to illustrate the data's range, marking the minimum and maximum values. Points outside the whiskers represent outliers.

Figure 3.13 illustrates the performance evaluation of FT, and the figure shows that for any mapping strategy, the two-opt rank placement optimization for collective communication operation improves overall performance. The performance improvement

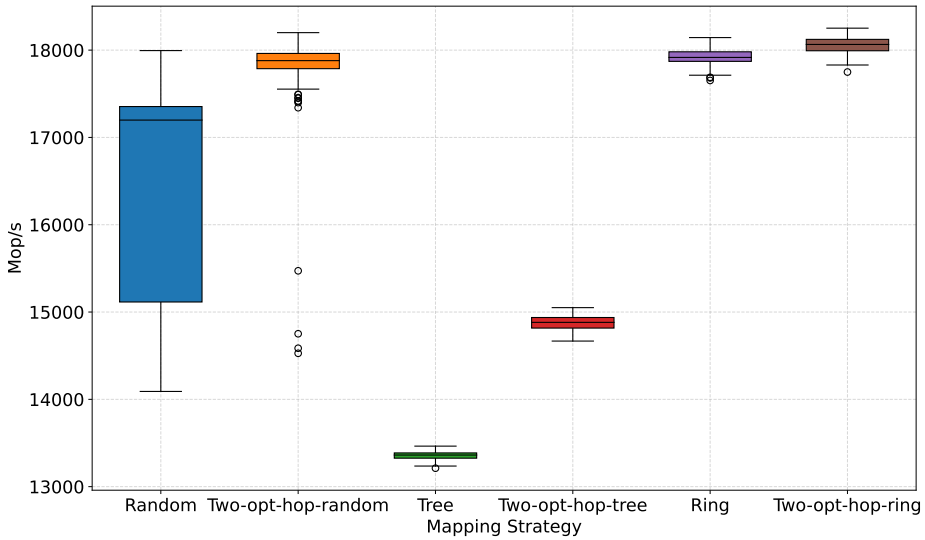


Figure 3.14: Performance evaluation of application IS in random shortcut network topology.

is especially noticeable for hierarchical tree and ring-based consecutive mapping strategies.

Figure 3.14 illustrates the performance evaluation of IS, and the figure shows that for any mapping strategy, the two-opt rank placement optimization for collective communication operation improves overall performance. The performance improvement is especially noticeable for random and hierarchical tree mapping strategies.

Figure 3.15 presents a performance evaluation for the MG application. The hierarchical tree mapping strategy benefits from the two-opt rank placement optimization for the Allreduce operation, showing a marked enhancement in overall performance. In contrast, the performance gains are more modest for the random and ring-based consecutive mapping strategies. This variation in performance improvement can be attributed to the communication characteristics of MG, where the bytes of messages sent by collective communication represent only a tiny fraction of the total communication volume [131]. Consequently, enhancing the efficiency of collective communication operations has a somewhat limited impact on the overall performance. The evaluation results of collective communication in Section 3.4.2 show that Allreduce operations under original hierarchical tree mapping perform poorly and that applying a two-opt optimization can dramatically improve Allreduce performance. Therefore,

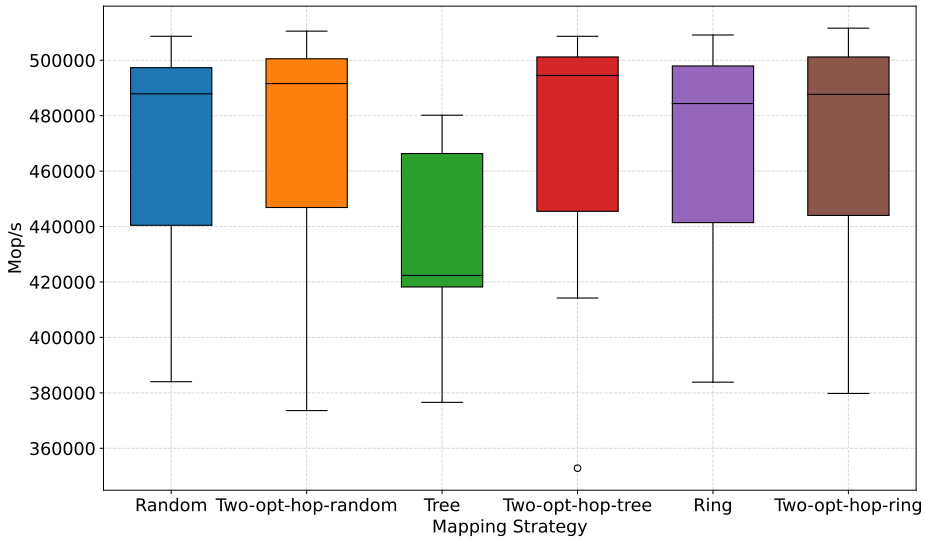


Figure 3.15: Performance evaluation of application MG in random shortcut network topology.

two-optimization can enhance the performance of MG with hierarchical tree mapping.

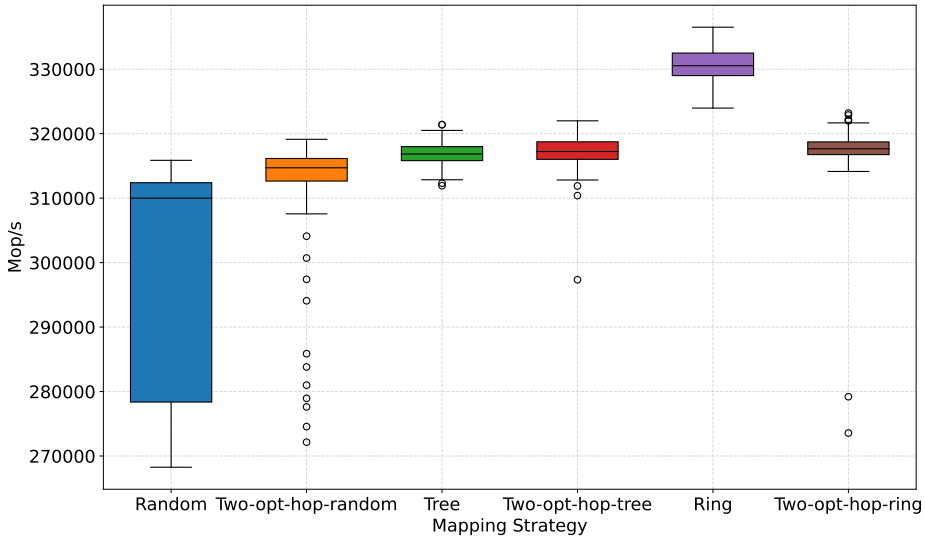


Figure 3.16: Performance evaluation of application LU in random shortcut network topology.

Figure 3.16 displays the performance evaluation results for the LU application. Similar to the communication characteristics of MG, the volume of bytes sent through

collective communication in LU constitutes a minimal portion of the overall communication volume [131]. Consequently, enhancements in collective communication performance only minimally impact the overall performance improvement. Interestingly, in ring-based consecutive mapping, the two-opt optimization degrades the performance of LU. This result may be because nodes in LU applications often communicate with their neighbors. Original ring-based consecutive mapping is conducive to this communication pattern, and two-opt rank placement increases the ASPL.

3.5 Discussions

3.5.1 Quality of Results

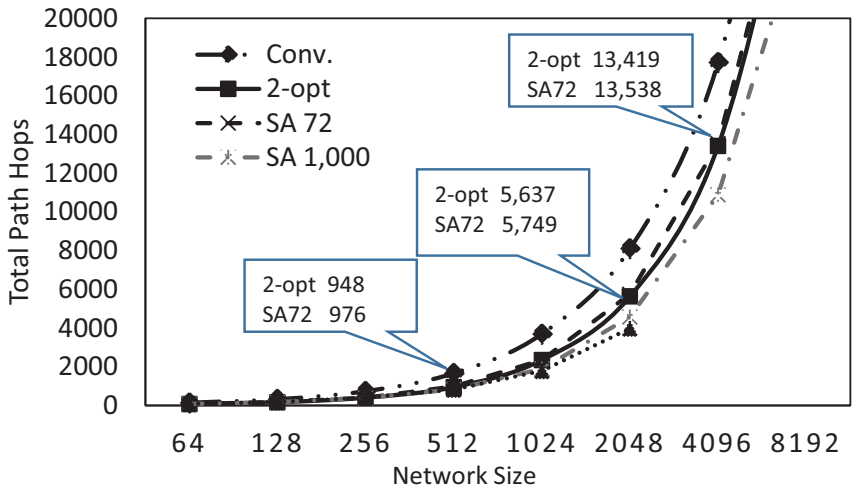


Figure 3.17: Total path hops of the two-opt, SA and MPICH2 Broadcast [3].

The quality of results of the two-opt algorithm would affect the performance of collective communication operations on the random network topologies. In this study, we select a two-opt algorithm to instantly compute the ordered list of visiting nodes in collective communication operations. However, the two-opt algorithm is simple and thus may result in a local minimum value in the optimization. Here, we compare the two-opt Broadcast that swaps the ordered list of visiting nodes 5,000 times to the other

sophisticated heuristics, i.e., simulated annealing (SA).

Generally speaking, the implementation of SA should execute many attempts, i.e., the node swap in the ordered list, to obtain a better solution. In Broadcast, 1,000 or larger iterations obtain a better solution.

Figure 3.17 illustrates the total path hops of the two-opt and the competitor's Broadcast for various network sizes. The lower value is better in the figure. The "two-opt" and "Conv" plots the total path hops of the two-opt Broadcast and that in MPICH2, respectively. The optimization is done to reduce the total path hops of point-to-point communications. The "SA 1,000" represents the SA with 1,000 iterations. SA provides shorter total path hops in Broadcast, and the two-opt Broadcast is marginally inferior to the SA with 1,000 and 10,000 iterations. However, both SAs can not provide the solution, i.e., the ordered list of visiting nodes, as the network size becomes large, e.g., 2,048 for SA with 10,000 iterations, due to its high computation cost. By contrast, the two-opt Broadcast, SA with 72 iterations and that in MPICH2 provide the solution even if the network size becomes large, e.g., the 8,192-node random network topology.

The SA Broadcast with 72 iterations has almost the same or slightly longer computation time than the two-opt Broadcast in Figure 3.18. The SA Broadcast with 72 iterations is thus a competitor to the two-opt Broadcast. The critical finding is that the two-opt Broadcast provides comparable and often slightly better total path hops (up to 3%) than the SA Broadcast with 72 iterations.

We thus conclude that the two-opt Broadcast has a good trade-off between the scalability and quality of results.

3.5.2 Scalability

A potential scalability issue could be the calculation time for Broadcast on a random network topology. We detail the scalability. Figure 3.18 shows the computation time vs. network size for 8-degree random network topologies when the two-opt and SA Broadcast executes as a single-threaded program on a 3.20 GHz Intel Xeon CPU E5-2667 v4 with 528 GiB of RAM. The two-opt and SA Broadcasts are optimized in terms of total path hops of point-to-point communications that form a Broadcast. The two-opt algorithm swaps 5,000 times to attempt to update Broadcast's ordered list

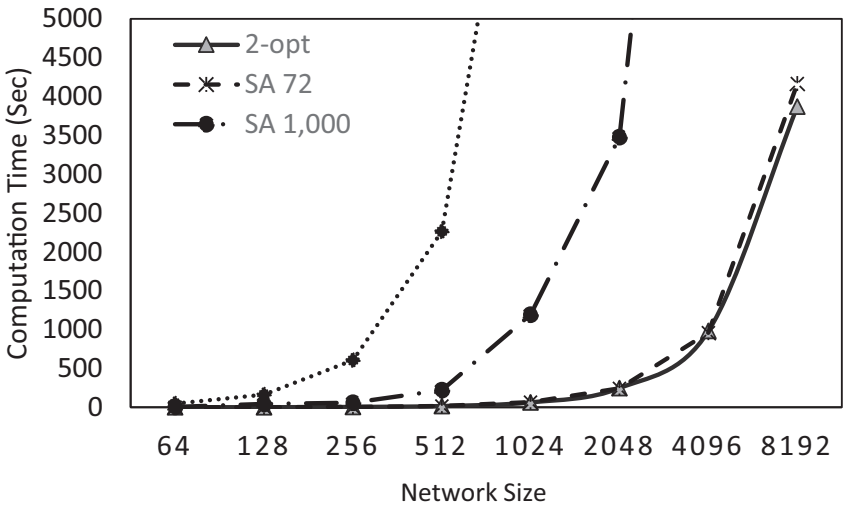


Figure 3.18: Computation time vs. network size for two-opt Broadcast [3].

of nodes. As expected, the results show that it is feasible to instantly compute the two-opt Broadcast and the SA Broadcast with 72 iterations by up to 2,048-node random network topologies, e.g., 4.32 seconds for 256 nodes. By contrast, the computation time of SAs is saturated at 1,024 nodes and 4,096 nodes for 1,000 iterations and 10,000 iterations, respectively.

3.6 Summary

In this chapter, we applied the two-opt approach to replace processes' rank for building efficient collective communication in random shortcut network topologies.

The two-opt approach replaces the processes's rank, changing the point-to-point communication that makes up the collective communication, which affects the overall performance of the collective communication.

The discrete-event simulation results significantly enhance collective communication performance using the two-opt approach. This improvement is achieved by reducing the hops of collective communication operations and boosting the overall performance of parallel applications, particularly in scenarios where collective communication plays a dominant role in application performance.

We further explore the performance factors. For a given random network topology,

the collective communication operations should be instantly computed for each source-destination pair before executing the target parallel programs.

Regarding total path hops, the two-opt approach is comparable to the competitor's simulated annealing (SA) and has a lower overhead.

4

Efficient Collective Communication using Circulant Network Topology

4.1 Introduction

Chapter 2 describes that collective communication operations significantly impact the performance of parallel applications. Chapter 3 investigates using the two-opt approach in random shortcut network topology to reduce the number of hops in collective communication and thus improve communication efficiency and shows that in applications where collective communication operations are predominant (such as FT in NAS parallel benchmark [130, 131]), reducing the total hop count through the two-opt method can significantly improve overall application performance. Although the two-opt approach in Chapter 3 can dramatically reduce the hop count of collective communication, as shown in Figures 3.7, 3.8 and 3.9, the hops of collective communication stabilizes with increasing number of iterations.

This chapter focuses on the collective communication on a particular circulant

network topology. Cooperative communication has a minimal number of hops on this network topology, even down to the theoretical minimum - i.e., the number of hops between any two nodes communicating in collective communication is one. Typically, it is costly to implement a network topology with a hop count of one between nodes. For instance, a fully connected network topology ensures that any two nodes are adjacent. Fully connected topology could be applied to a small interconnection network for efficient collective communication [8]. However, as the size of fully connected topology increases, the explosion in the number of links is unacceptable. Many low-diameter network topologies, such as Dragonfly [12] and Slim Fly [13], try to achieve low communication hops and cost-effectiveness.

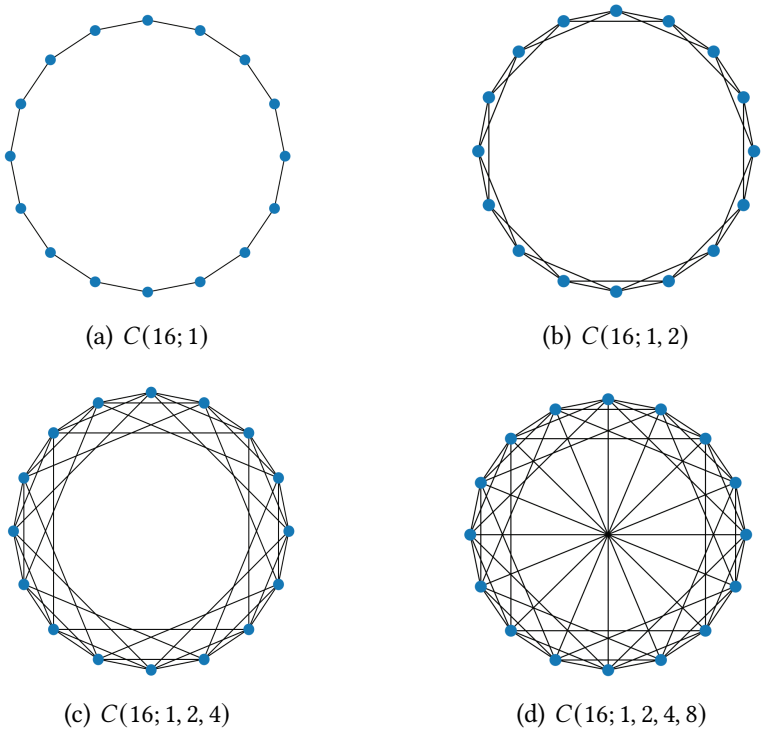
The circulant network topology used in this chapter has a similar construction process to the random shortcut topology in Chapter 3, i.e., augmenting shortcut links to a basic ring topology. The circulant network topology takes a higher diameter and ASPL than the counterpart random shortcut network topology with the same degree [2]. However, the circulant network topology interestingly enables collective communication operations with lower aggregate path hops than the random shortcut network topology. More precisely, for Broadcast, Allreduce, and Alltoall operations with a specific algorithm, such as binomial tree Broadcast, recursive-doubling Allreduce, Bruck's Alltoall[99], the aggregate path hops can reach a theoretical minimum. Theoretically minimum hop counts mean that all the point-to-point communications that form a collective-communication operation take only one hop.

Section 4.2 introduces circulant topology and the generation of the target circulant topology discussed in this chapter. Section 4.3 introduces the behavior patterns of three collective communication operations: Broadcast, Allreduce, and Alltoall in Circulant Topology. Section 4.4 presents a circulant mapping strategy to achieve the minimum collective communication hops count. Section 4.5 evaluates collective communication operations and parallel applications on circulant network topology. Section 4.6 summarizes the contents of this chapter.

4.2 Target Circulant Network Topology

4.2.1 Circulant Topology

A circulant graph $C(n; c_1, c_2, ..., c_k)$ has n vertices $v_0, v_1, ...v_{n-1}$, in which $c_1 = 1$ and $c_i < c_{i+1}$, v_a is connected to v_b if and only if $a \equiv b \pm c_i(\mod n)$. The sequence $< c_1, c_2, ..., c_k >$ is called jump sequence, $c_1, c_2, ..., c_k$ are called jumps, the maximum jump $c_k \leq \lfloor n/2 \rfloor$ [96, 132]. Figure 4.1 illustrates the examples of circulant graphs with 16 vertices.



(a) $C(16; 1)$ (b) $C(16; 1, 2)$

(c) $C(16; 1, 2, 4)$ (d) $C(16; 1, 2, 4, 8)$

Figure 4.1: Circulant graphs with 16 vertices.

Circulant network topology is widely used in various network designs. They are used as a ring extension to design and implement local area networks. The networks are also called distributed loop computer-networks[49, 50]. A variant of circulant network topology called Multi-Ring network topology is used to achieve high-performance group communication in peer-to-peer networks[94]. The paper [95] proposed recursive circulant network topology for multicomputer systems. The

recursive circulant network topology is also circulant network topology. The paper [96] proposed generalized recursive circulant graphs, which are the extension of recursive circulant graphs. Recently, the research[53] presented the optimal circulant network topologies as low latency network topologies.

4.2.2 High-Radix Circulant Network Topology

In this work, we focus on the circulant graphs $C(n; c_1, c_2, ..., c_k)$ with the following properties:

- The number of vertices n is powers of two.
- The jump $c_i = 2^{i-1}$, c_i is in jump sequence $< c_1, c_2, ..., c_k >$, $1 \leq i \leq k = \log_2 n$.

We find that circulant topology that satisfies the above conditions is well suited for collective communication, the binomial tree Broadcast, recursive doubling Allreduce and Bruck's Alltoall on this type of circulant network topology can achieve the minimum number of total hop counts. In Section 4.3, the communication behavior of these collective communication operations is analyzed in this circulant network topology. We can use $G(n, k)$ to represent this class of circulant topology. n is the number of vertices, k is the number of elements of sequence $< c_1, c_2, ..., c_k >$, $n = 2^k$ and the degree of each vertex is $2k - 1$. The circulant graph $C(16; 1, 2, 4, 8)$ in figure 4.1(d) can also be represented as $G(16, 4)$.

We apply the circulant graph $G(N, \log_2 N)$ to the network topology of parallel computers; N is a power of two. Each vertex in the graph represents a switch, and multiple compute nodes are attached to each switch. The degree of a vertex in circulant graph $G(N, \log_2 N)$ is $2 \log_2 N - 1$, and the switch has $2 \log_2 N - 1$ cables linked to other switches. In this dissertation, we focus on collective communication operations on network topology, assuming only one compute node on each switch.

4.3 Collective Communication in Target Circulant Network Topology

We apply the circulant graphs $G(n, \log_2 n)$ to the network topology of parallel computers, assuming that each vertex represents a compute node. Then, we detail the behavior of

Broadcast, Allreduce, and Alltoall on the circulant network topology $G(16, 4)$. The binomial tree Broadcast, recursive doubling Allreduce and Bruck's Alltoall in circulant $G(n, \log_2 n)$can achieve the minimum number of total hops.

4.3.1 Broadcast in Circulant Network Topology

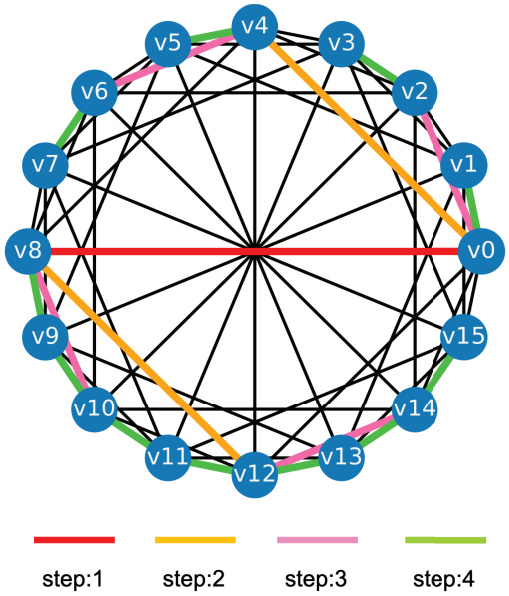


Figure 4.2: Binomial tree Broadcast in circulant network topology.

Figure 4.2 illustrates a Broadcast operation using the binomial tree algorithm on a $G(16, 4)$ network topology. This operation is completed in four steps. In the first step, we perform the point-to-point communication $v_0 \rightarrow v_8$. Fortunately, there is a direct link between v_0 and v_8. In the second step, we archive the point-to-point communications from $v_0 \rightarrow v_4$ and $v_8 \rightarrow v_{12}$, there are also links between nodes v_0, v_4 and v_8, v_{12}. In the last two steps, each point-to-point communication only requires one hop, similar to the preceding two steps. As Figure 4.2 illustrates, each point-to-point communication in the Broadcast operation on the circulant network topology involves only one hop. Therefore, the total hop count for the binomial tree Broadcast reaches the ideal minimum number of hops.

4.3.2 Allreduce in Circulant Network Topology

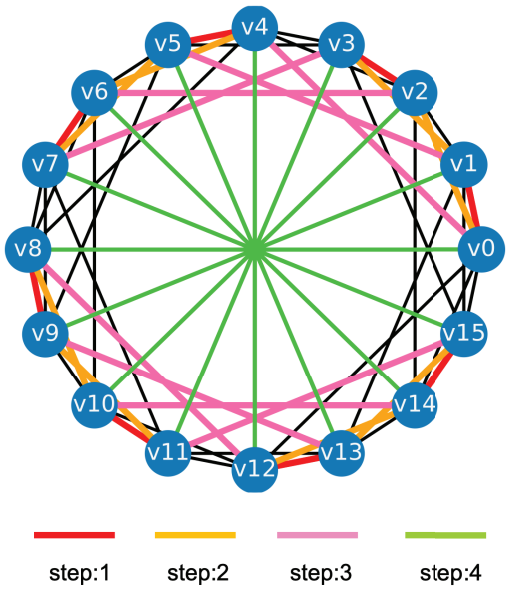


Figure 4.3: Recursive doubling Allreduce in circulant network topology.

Figure 4.4 shows an example Allreduce operation with a recursive doubling algorithm on $G(16, 4)$ circulant network topology. It takes four steps to finish the Allreduce operation. In the first step, eight pairs nodes exchange there messages $v_0 \leftrightarrow v_1$, $v_2 \leftrightarrow v_3$, $v_4 \leftrightarrow v_5$, $v_6 \leftrightarrow v_7$, $v_8 \leftrightarrow v_9$, $v_{10} \leftrightarrow v_{11}$, $v_{12} \leftrightarrow v_{13}$, $v_{14} \leftrightarrow v_{15}$, we can clearly see that each pair of nodes are connected. In the last three steps, each step has eight pairs of nodes exchanging their messages, and each pair is connected. The total hops of the Allreduce operation can also reach the ideal minimum hops.

4.3.3 Alltoall in Circulant Network Topology

Figure 4.4 provides an example of Alltoall operation using Bruck's algorithm on a $G(16, 4)$ network topology. It takes four steps to finish the Alltoall operation. In this Alltoall operation, all the network topology links are fully utilized. In Bruck's Alltoall operation, every point-to-point communication has both the source and destination nodes directly connected, resulting in a single hop for each communication. Therefore, the total hop count of the Alltoall operation also achieves the ideal minimum hop count.

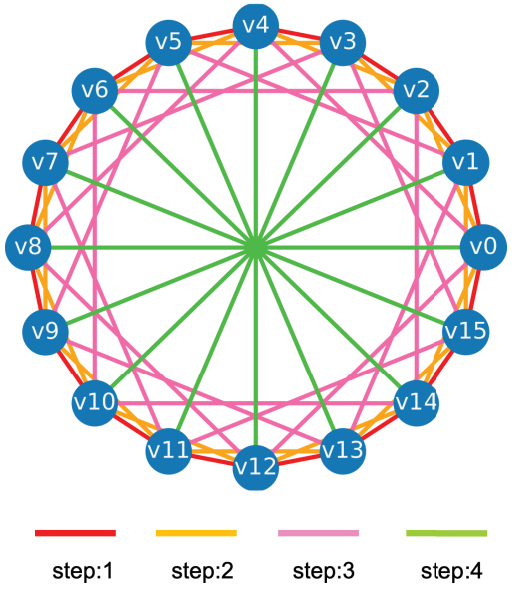


Figure 4.4: Bruck's algorithm Alltoall in circulant network topology.

4.3.4 Hop Count of Collective Communication Operations in Circulant Network Topology

In the above, we explained the collective operations on $G(16, 4)$ circulant network topology. It can be generalized as follows.

For any node v_a in the $G(n, \log_2 n)$ network topology, it has $2k-1$ neighboring nodes $\{v_b | b \equiv a \pm 2^{i-1} \mod n, 1 \leq i \leq k\}$. For the sake of convenience, in the Broadcast operation, we select v_0 as the root node. Due to the symmetry of the above circulant network topology, choosing any node as the root node has the same effect. In the first step, v_0 as the source node, there is point-to-point communication $v_0 \rightarrow v_{\frac{n}{2}}$. In the second step, nodes v_0 and $v_{\frac{n}{2}}$ are the source nodes, there are point-to-point communications $v_0 \rightarrow v_{\frac{n}{4}}$ and $v_{\frac{n}{2}} \rightarrow v_{\frac{3n}{4}}$. More generally, in the ith step, there are 2^{i-1} source nodes $\{v_{\frac{jn}{2^{i-1}}} | 0 \leq j \leq 2^{i-1}-1\}$, the point-to-point communication from source node $v_{\frac{jn}{2^{i-1}}}$ is $v_{\frac{jn}{2^{i-1}}} \rightarrow v_{\frac{(2j+1)n}{2^i}}$. Let $a = \frac{jn}{2^{i-1}}$, the point-to-point communication $v_{\frac{jn}{2^{i-1}}} \rightarrow v_{\frac{(2j+1)n}{2^i}}$ can be converted to $v_a \rightarrow v_{a+\frac{n}{2^i}}$. Since $n = 2^k$, $\frac{n}{2^i}$ can be represented as 2^{k-i}. The destination node of point-to-point communication $v_{a+2^{k-i}}$ is the neighboring node of source node v_a. For any point-to-point communication of Broadcast operation, the source node and destination node are connected, and the hop counts of the

point-to-point communication are one, so the Broadcast operation on the $G(n, \log_2 n)$ circulant network topology can reach the ideal minimum hops.

In the Allreduce operation using recursive doubling algorithm, at the ith step, the two nodes have a distance 2^{i-1} to exchange their data. Assume that nodes v_a and $v_b (b > a)$ exchange their data at ith step, the equation $b = a + 2^{i-1}$ holds. It's obvious that v_a and v_b are connected, the total hops of two point-to-point communication $v_a \rightarrow v_b$ and $v_b \rightarrow v_a$ are two. Therefore, the Allreduce operation on $G(n, \log_2 n)$ network topology can reach the ideally minimum hops.

In the Alltoall operation using Bruck's algorithm, at the ith step, node v_a sends message to node $v_b, b = a - 2^i + n \mod n$, so the node v_b is the neighboring node of v_a, the hop count of point-to-point communication $v_a \rightarrow v_b$ is one. Thus, the Alltoall operation on $G(n, \log_2 n)$ circulant network topology can also reach the ideal minimum hops.

Collective Communication	Algorithm	Hop count
Broadcast	Binomial tree	$n - 1$
Allreduce	Recursive doubling	$n \log_2 n$
Alltoall	Bruck's algotirhm	$n \log_2 n$

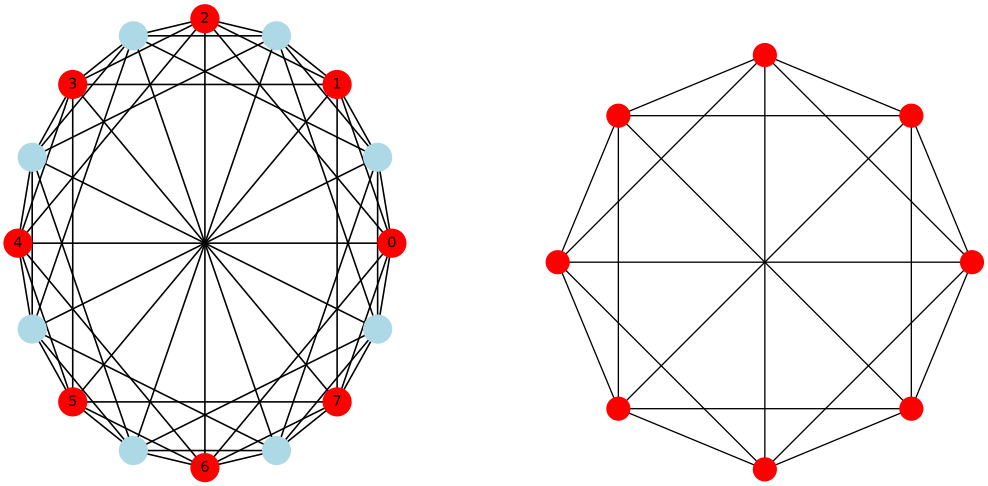
Table 4.1: Hop count of collective communication operations in circulant network topology

We can derive the number of hops performing a collective communication operation in a circulant network topology $G(n, \log_2 n)$. Assume that all nodes in $G(n, \log_2 n)$ are involved in collective communication and use the shortest path routing. Since any point-to-point communication of collective communication operation has a hop count of 1, the total hop count of collective communication equals the number of point-to-point communications. Table 4.1 shows the hop count of collective communication operations using a specific algorithm.

4.4 Circulant Mapping Strategy for Circulant Network Topology

In Chapter 3, we introduced random, hierarchical-tree, and ring-based consecutive mapping strategies for random shortcut network topology, and these mapping strategies can be applied to circulant network topology as well.

In our previous assumption, we considered all nodes in $G(n, \log_2 n)$ participating in collective communication, enabling it to achieve the minimum number of hops. However, applying the mapping strategies introduced in Chapter 3 to circulant network topology, where a part of nodes participate in collective communication, may prevent achieving the minimum hop count. We propose a mapping strategy for the circulant network topology that enables collective communication to reach the minimum hop count even when a part of the nodes participates in communication.



(a) Mapping 8 nodes in circulant network topology $G(16, 4)$

(b) The mapped topology is $G(8, 3)$.

Figure 4.5: An 8-node mapping on a 16-node circulant network topology.

Figure 4.5(a) provides an example of mapping eight nodes in circulant network topology $G(16, 4)$. Figure 4.5(b) shows that the topology of the mapped nodes is still a circular topology $G(8, 3)$. Since the mapped topology remains a circulant topology, the aggregate communication operations that the mapped nodes participate in still achieve a minimum number of hops. This mapping strategy allows the topology consisting of

mapped nodes to remain a circulant topology, which we call the circulant mapping strategy.

We assume that m nodes are mapped in circulant network topology $G(n, \log_2 n)$. Since the mapped topology is circulant topology $G(m, \log_2 m)$, m should be satisfied as a power of 2, and $m \leq n$. There are $\frac{n}{m}$ sub-circulant topologies $G(m, \log_2 m)$ in the circulant network topology $G(n, \log_2 n)$; the mapping strategy needs to map m nodes into any sub-circulant topology $G(m, \log_2 m)$.

Algorithm 4.1 Mapping m nodes into circulant network topology $G(n, \log_2 n)$

Require: Host topology $G(n, \log_2 n)$, nodes in $G(n, \log_2 n)$ $V = \{v_0, v_1, ..., v_{n-1}\}$, mapped nodes $M = \{\}$, number of mapped nodes m
Ensure: The topology of mapped nodes M is circulant topology $G(m, \log_2 m)$

```
stride ← n/m
i ← rand() mod n
M ← M ∪ {v_i}
while |M| < m do
    i ← i + stride mod n
    M ← M ∪ {v_i}
end while
```

Algorithm 4.1 describes a method for mapping m nodes in a circulant topology $G(n, \log_2 n)$ to make a circulant $G(m, \log_2 m)$. Here, we are only concerned with how to construct a sub-circulant topology $G(m, \log_2 m)$ in circulant topology $G(n, \log_2 n)$, not the efficiency of topology mapping, assuming that all nodes are idle and all nodes can be mapped immediately. With $\frac{n}{m}$ as the stride, starting from randomly selected node v_i, the mapped nodes are continuously added according to stride. When the mapping is completed, the mapped nodes form a circulant topology $G(m, \log_2 m)$.

4.5 Evaluation

4.5.1 Methodology

We use the discrete-event simulator SimGrid (v3.28) [133] to evaluate the performance of collective operations. In this evaluation, we focus on the impact of latency on performance, assuming that there is one compute node on each switch as Sec-

tion 4.2.2 described, and only one processor per compute node. The parameters of the interconnection network are illustrated in Table 4.2.

Table 4.2: Parameters of the interconnection network.

Power of compute node	100GFLOPS
Switch latency	100ns
Link bandwidth	100Gbps
Switch capacity	3.6Tbps
Routing algorithm	Floyd (Minimal Paths)

The number of compute nodes or switches is 1024, and the degree of each switch is 19 based on the properties of the target circulant network topology described in Section 4.2.2.

4.5.2 Execution Time of Collective Communication Operations

We compare the performance of four different mapping approaches, namely, circulant, ring-based consecutive, hierarchical-tree, and random mapping. For hierarchical tree and random mapping, we also apply the two-opt approach to optimize the rank placement to reduce the hops and contention for collective communication operation; the iteration times of the two-opt approach is 200000. We also applied the CCO algorithm on random and hierarchical tree mapping for the Broadcast operation.

Figures 4.6(a) and 4.6(b) present the execution time of the Broadcast operation in two different scenarios: small message sizes ranging from 256B to 16KB and large message sizes ranging from 64KB to 1MB, respectively. These results enable us to analyze the performance of the Broadcast operation under varying message size conditions. The lower values are better on the Y-axis (the unit is second). The results show that when the message size is small, after the message size becomes large (64KB-1MB), although circulant mapping and Ring-based consecutive mapping have the least number of hops, the performance improvement is not significant, and after the message size becomes large (64KB-1MB), circulant and ring-Ring-based consecutive mapping have the best performance.

Figures 4.7(a) and 4.7(b) present the execution time of the Allreduce operation. Allreduce has a much larger hop count than broadcast, and performance gains from reducing hops become apparent. Since circulant mapping for Allreduce has the same

number of hops as ring-based consecutive mapping, their Allreduce runtimes are almost identical. The results show that both circulant and ring-based consecutive mapping achieve the best results and significantly better performance when the message size of the paper is small or large. In the case of large message sizes, such as 1MB, compared to unoptimized random and hierarchical tree mapping, performance is improved by 51% and 80%, respectively. The performance is also improved by 29% and 42% compared to random and hierarchical tree mapping optimized with two-opt hop reduction, respectively.

Figures 4.8(a) and 4.8(b) present the execution time of the Allreduce operation. The evaluation results of the Alltoall operation follow a similar trend to the results of the Allrduce operation. Compared to other mapping strategies, circulant and ring-based consecutive mapping perform best. In the case of large message sizes, such as 1MB, compared to unoptimized random and hierarchical tree mapping, performance is improved by 60% and 72%, respectively. The performance is also improved by 21% and 34% compared to random and hierarchical tree mapping optimized with two-opt hop reduction, respectively.

4.5.3 Performance Evaluation of Parallel Applications

In this section, we evaluate the performance of parallel applications. Same as the evaluation in Section 3.4.3, we evaluate the applications in NAS parallel benchmark [130], which are FT, IS, MG, and LU. The problem size is Class A, the number of processes is 128, and the iteration times of the two-opt approach are 20,0000. Broadcast, Allreduce, and Alltoall algorithms are binomial trees, recursive doubling, and Bruck's ones [99]. Table 4.5.3 shows the description of applications and the collective communication operation to be optimized for each application.

Table 4.3: Applications description and its frequently used collective communication operations.

Application	Description	Collective Operation
FT	discrete 3D fast Fourier Transform	Alltoall
IS	Integer Sort	Alltoall, Alltoallv
MG	Multigrid	Allreduce
LU	Lower-Upper Gauss-Seidel solver	Allreduce

Figures 4.9 to 4.12 display box plots of parallel applications FT, IS, MG, and LU performance. The x-axis shows Mop/s, with higher values indicating better performance, and the y-axis shows different mapping strategies. Each figure consists of six data sets for random, hierarchical tree, ring-based consecutive, and circulant mapping, each in versions with and without two-opt optimization except for circulant mapping, and each set includes 100 data points. In these box plots, the median appears as a horizontal line inside each box, while the lower and upper quartiles form the bottom and top of the box. The "whiskers" extend from each box to illustrate the data's range, marking the minimum and maximum values. Points outside the whiskers represent outliers.

Figure 4.9 illustrates the performance evaluation of FT, and the figure shows that the performance circulant and ring-based consecutive mappings are better than the others. In contrast, circulant mappings are slightly better than ring-based consecutive mappings. Figure 4.10 illustrates the performance evaluation of IS, and the figure shows that the ring-based consecutive mapping has the best performance. Figure 4.11 presents a performance evaluation for the MG application. The evaluation results show that the performance of the four mapping strategies does not have noticeable differences.

Figure 4.12 displays the performance evaluation results for the LU application. Circulant mapping and ring-based consecutive mapping are significantly better than random and hierarchical tree mapping. Circulant mapping has the best performance and is considerably better than ring-based consecutive mapping.

4.6 Summary

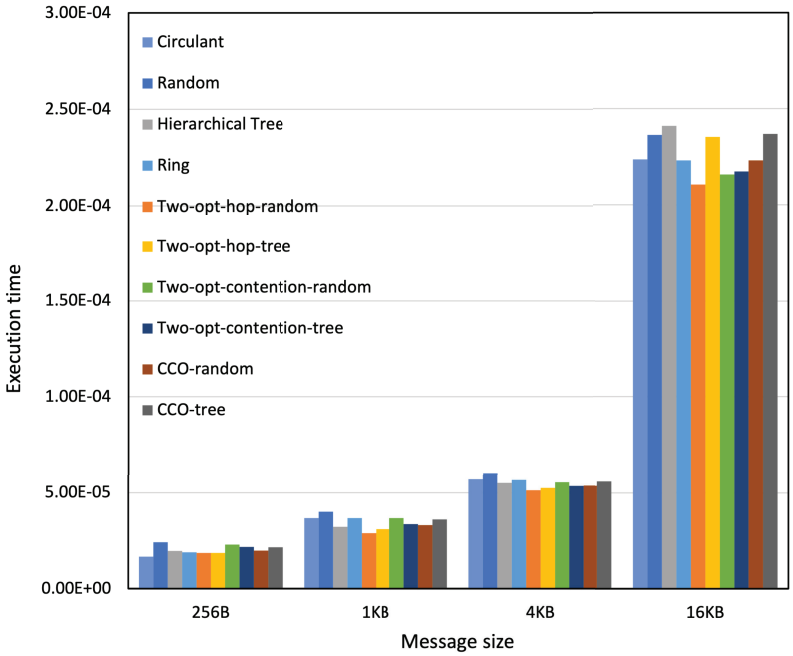
In this chapter, we apply a particular circulant graph to build a network topology network. This circulant network topology is highly suitable for collective communication operations. We analyze the behavior of three collective communication operations: Broadcast, All reduce, and Alltoall with the binomial tree, recursive doubling, and Bruck's algorithm, respectively. These three collective communication operations can achieve ideal hop counts, which means each communication between two nodes only has one hop. For the circulant network topology, we proposed using ring-base consecutive and circulant mapping; these two mapping strategies can implement

extremely low hops collective communication operations. In particular, circulant mapping can achieve ideal hops collective communication.

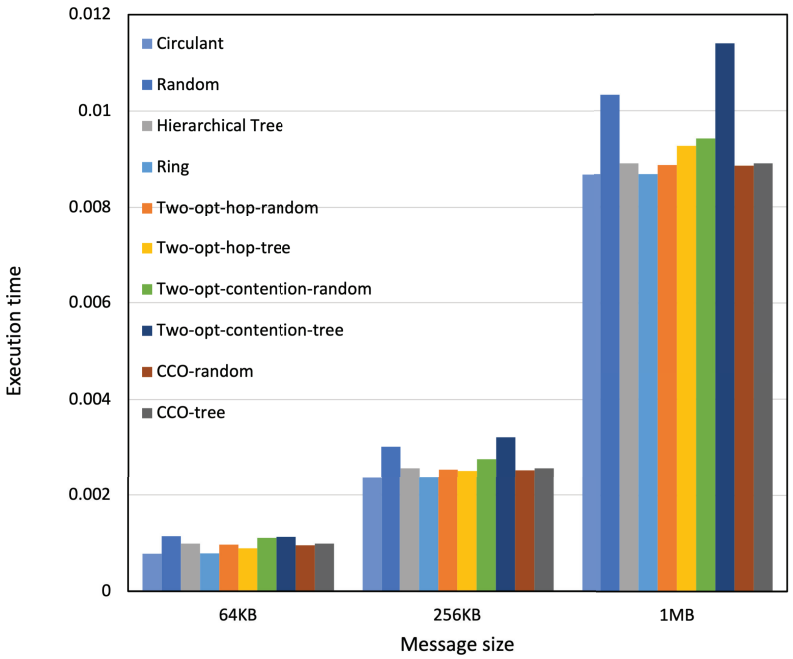
We also propose a process mapping strategy for ring network topologies called circulant mapping, which enables the partially mapped compute nodes involved in collective communication to achieve the theoretically lowest number of hops still.

The discrete-event simulation results in SimGrid show that our proposed mapping strategy, which aims to minimize the number of hops in collective communication, significantly improves the collective communication performance compared to other mapping strategies, such as random and hierarchical-tree mapping.

The results of the evaluation of collective This efficient collective communication also reflects the performance of parallel applications. The evaluation of the application shows that the efficient collective communication on circulant network topology leads to the high performance of parallel applications.

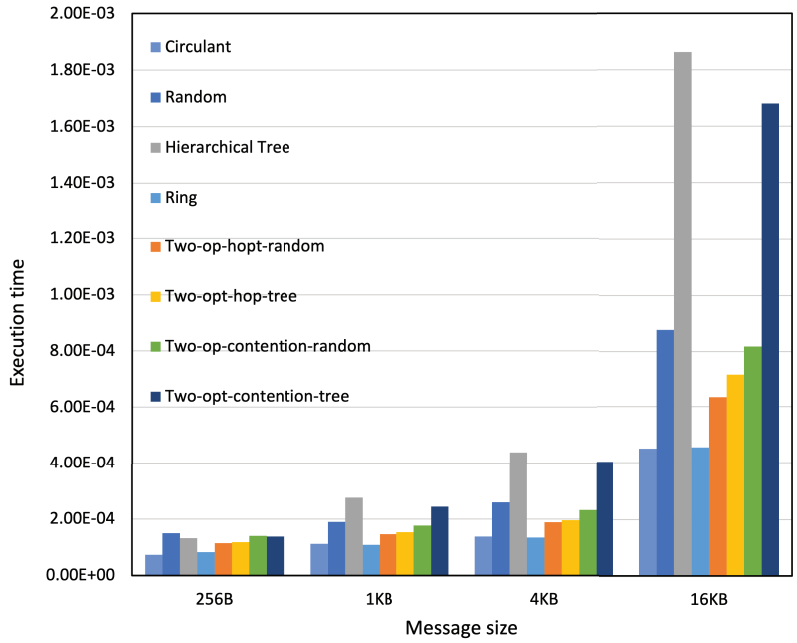


(a) Execution time of Broadcast with small message size.

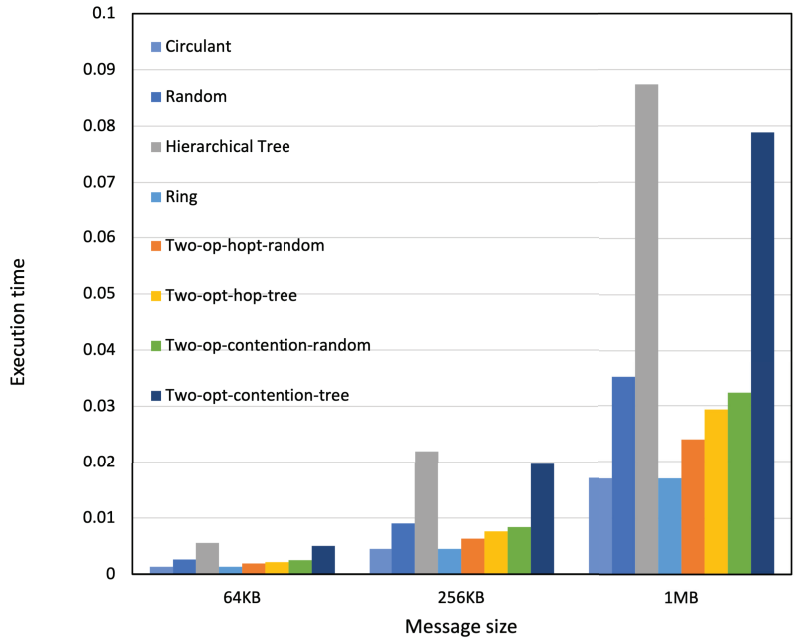


(b) Execution time of Broadcast with large message size.

Figure 4.6: Execution time of Broadcast operation on circulant network topology (1,024 nodes, 512 processes).

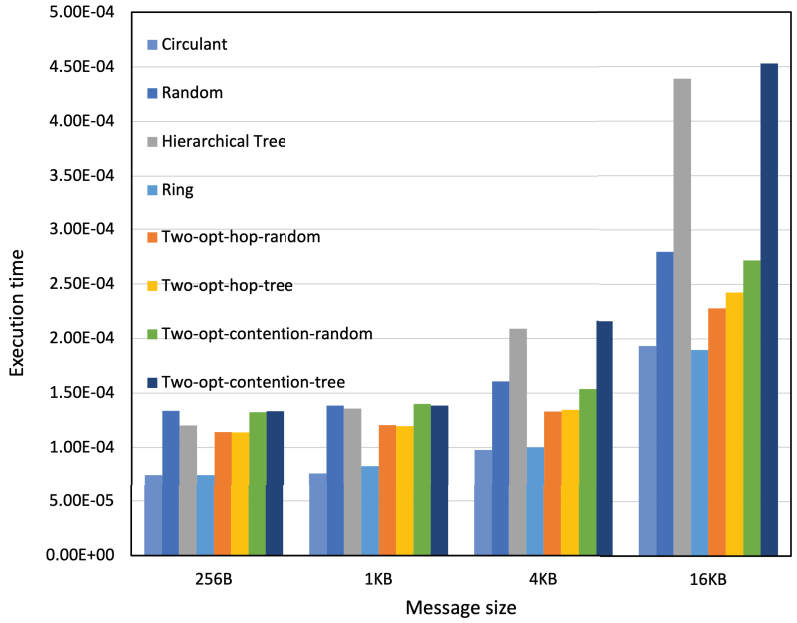


(a) Execution time of Allreduce with small message size.

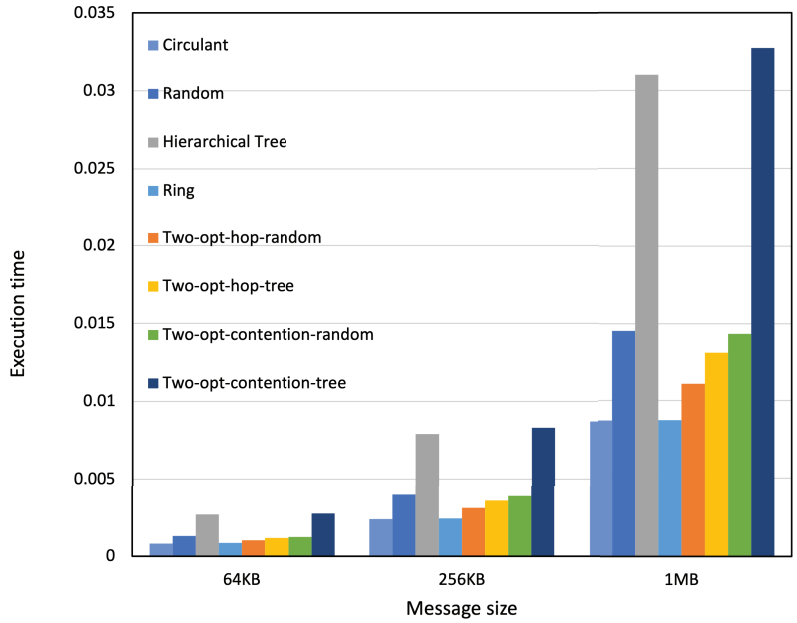


(b) Execution time of Allreduce with large message size.

Figure 4.7: Execution time of Allreduce operation on circulant network topology (1,024 nodes, 512 processes).



(a) Execution time of Alltoall with small message size.



(b) Execution time of Alltoall with large message size.

Figure 4.8: Execution time of Alltoall operation on circulant network topology (1,024 nodes, 512 processes).

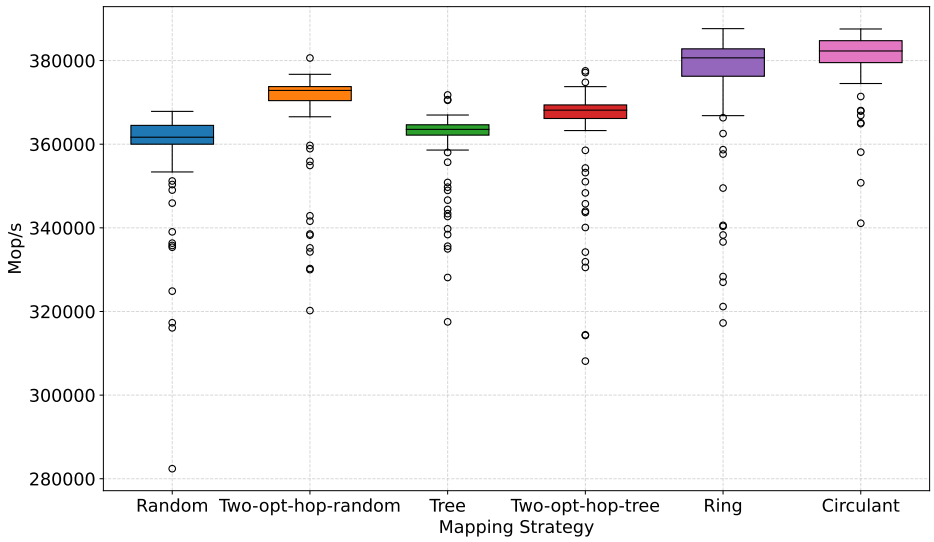


Figure 4.9: Performance evaluation of application FT in circulant network topology.

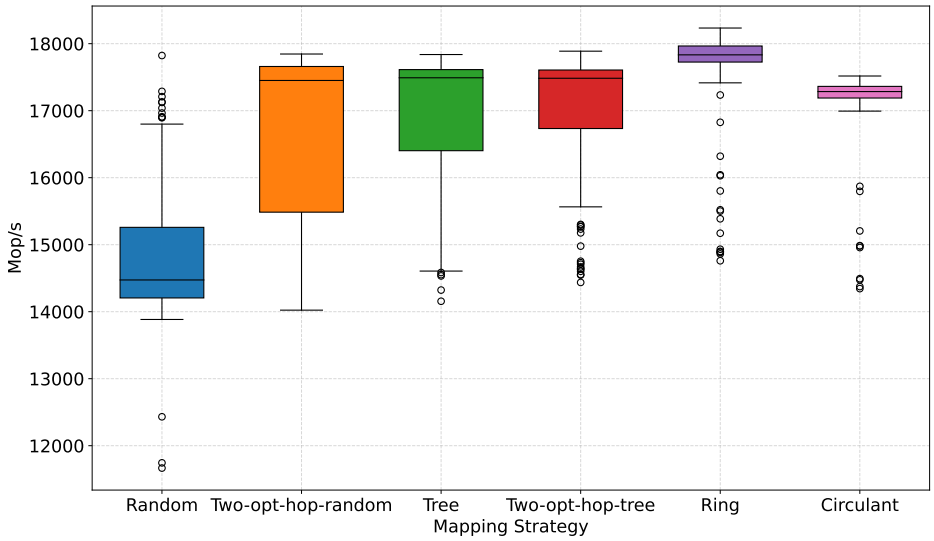


Figure 4.10: Performance evaluation of application IS in circulant network topology.

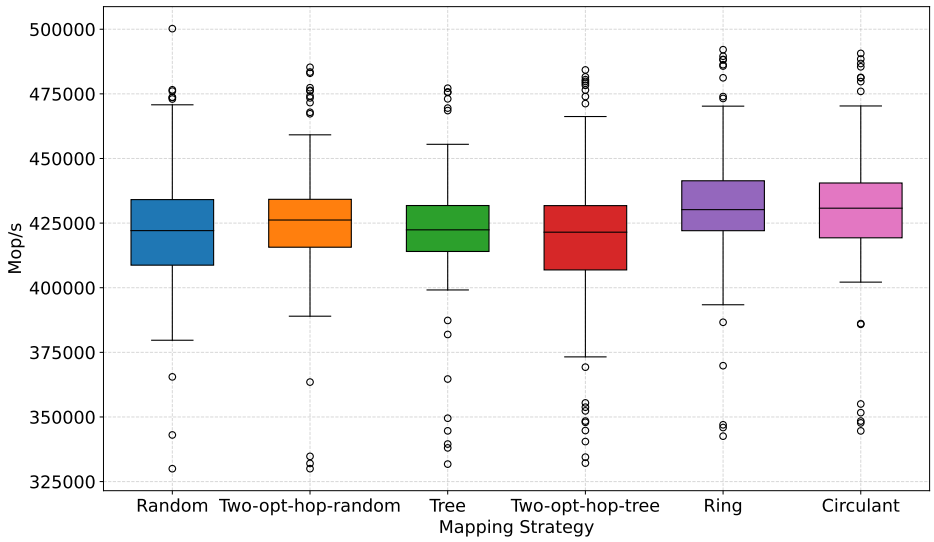


Figure 4.11: Performance evaluation of application MG in circulant network topology.

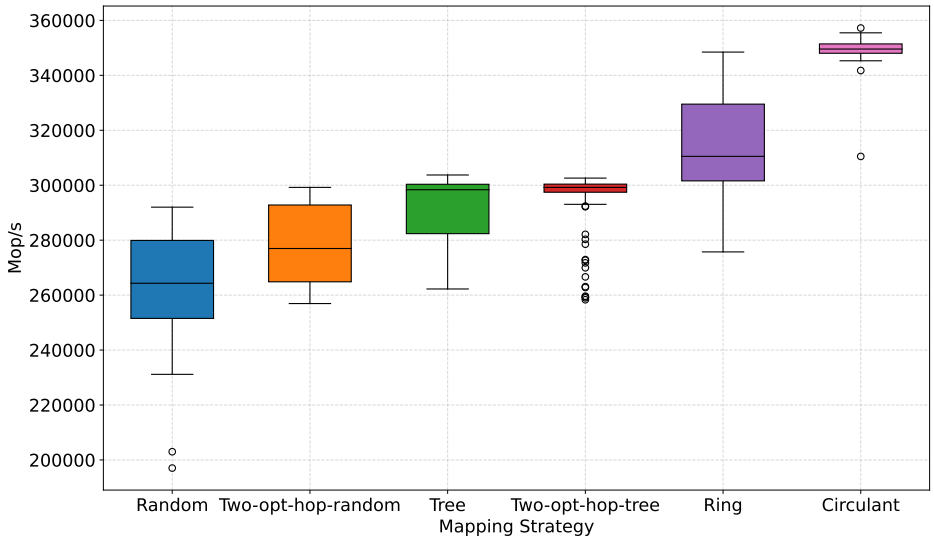


Figure 4.12: Performance evaluation of application LU in circulant network topology.

5

Comparison of Random and Non-Random Network Topologies

We proposed efficient collective communication using two different network topologies: random shortcut network topology and circulant network topology. These two types of network topologies are constructed by augmenting shortcut links to a basic ring topology. The difference is that random shortcut network topology is obtained by randomly increasing shortcut links, while non-randomly adding shortcut links receives circulant. Then, one might think, "Which is better?". This chapter attempts to answer by comparing these two types of topologies.

This chapter first compares the properties of network topologies such as diameter and ASPL. It secondly illustrates the comparison of the hop counts and performance of collective communication operations. Thirdly, it compares the hop counts and performance of point-to-point communication on these two network topologies. It also analyzes the total communication hop count by varying the point-to-point and collective communications ratio. Then, the performance of parallel applications is

compared. Finally, we compare the cost of these two network topologies.

5.1 Diameter and Average Shortest Path Length

Diameter and ASPL are essential properties to measure the performance of network topology. We compared the diameter and ASPL of these shortcut network topologies and the Dragonfly network topology. The Dragonfly is a low latency, low diameter, and cost-effective network topology that has been deployed on real large-scale parallel computers [89, 90, 91, 92], and we chose it as our baseline.

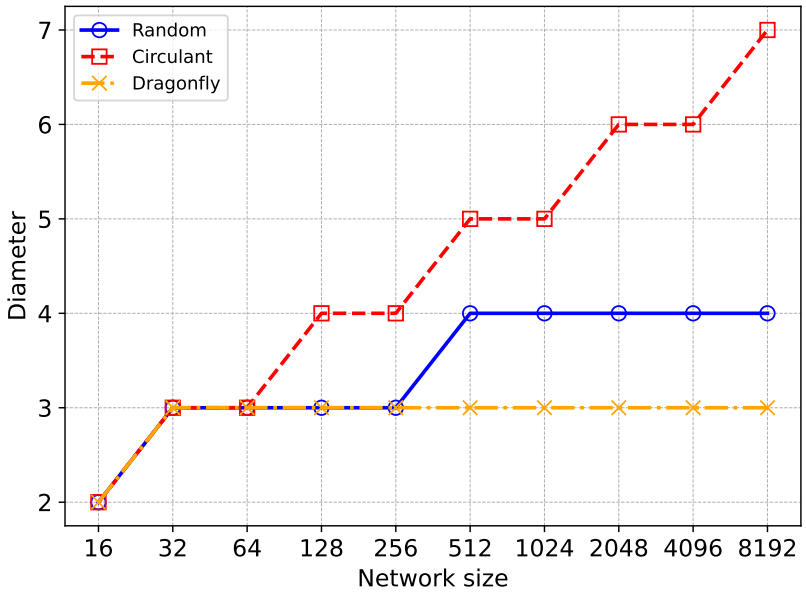


Figure 5.1: Comparison of diameter of three network topologies.

Figures 5.1 and 5.2 illustrated the diameter and average shortest path length (ASPL) of random shortcut, circulant, and Dragonfly network topologies, respectively. The x-axis in both figures represents the network size, i.e., the number of switches. Each network topology has the same degree, the number of degree d satisfies the condition in Chapter 4 : $d = 2\log_2 N - 1$. The y-axis in Figure 5.1 is diameter, and that in Figure 5.2 represents their ASPL.

As shown in Figure 5.1, the diameter of the Dragonfly network topology is always 3, and the diameter of the random shortcut network topology is more significant

than that of the Dragonfly. Still, it is relatively stable, with a diameter of 4, when the network size is 8194. The diameter of the circulant network topology increases as the network size increases, and when the network size is 8192, the diameter reaches 7. When the network is more significant than 64, the circulant network topology has a larger diameter than the random shortcut network topology.

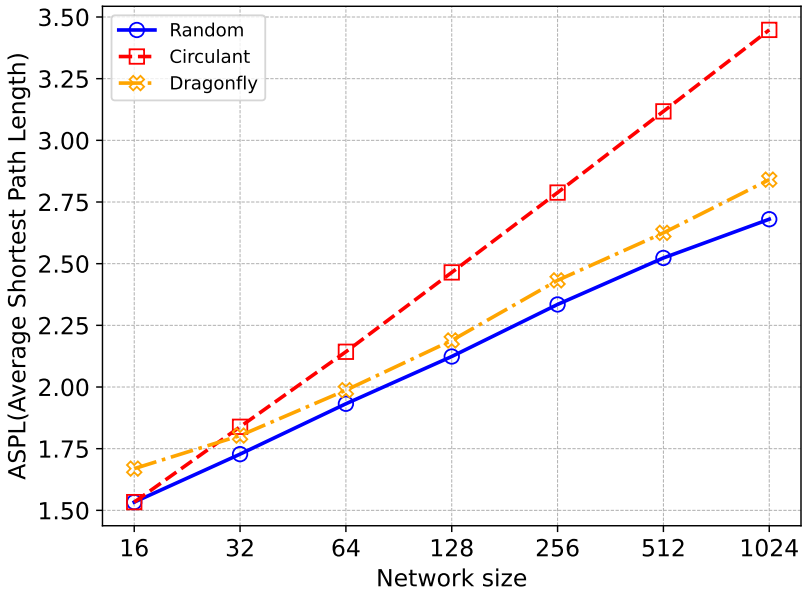


Figure 5.2: Comparison of average shortest path length of three network topologies.

Figure 5.2 illustrates the ASPL comparison of these three network topologies. Although the Dragonfly network topology has a smaller diameter than the random shortcut network topology, the random shortcut network topology has a smaller ASPL, and the circulant network topology has the largest ASPL.

5.2 Collective Communication in Shortcut Network Topologies

5.2.1 Hop Count of Collective Communication Operations

Figures 5.3, 5.4 and 5.5 illustrated the hop count of Broadcast, Allreduce and Alltoall operations on circulant, random shortcut and Dragonfly network topologies,

respectively. The X-axis represents the network size, and the Y-axis represents the hop count.

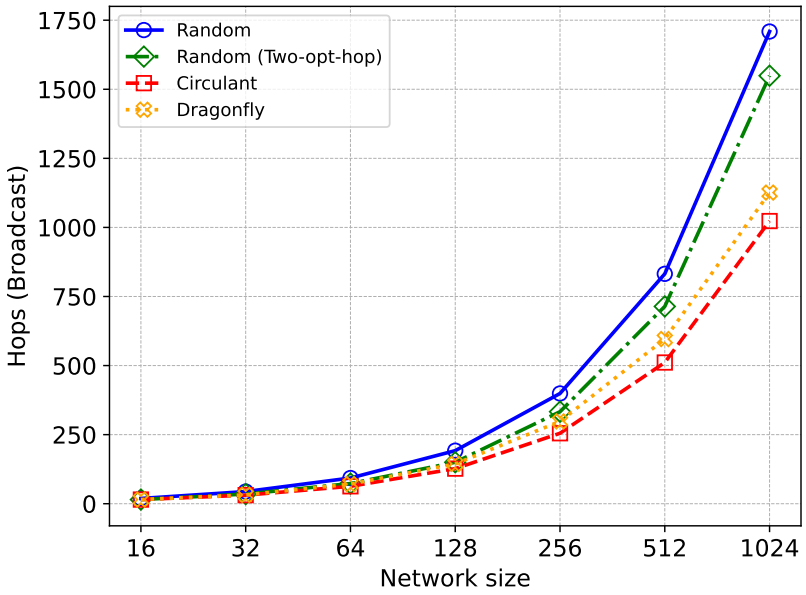


Figure 5.3: Hop counts of Broadcast operations for three network topologies.

Similar to the results in Section 5.1, all network topologies have the same degree, and the degree $d = 2\log_2 N - 1$. We assume that one compute node is connected to each switch, and all compute nodes involve the collective communication operations. We attempted ring-based consecutive mapping. There is also the case of two-opt rank placement optimization for random shortcut network topology. The iteration time is 200,000i n two-opt rank placement optimization.

We previously illustrated that the binomial-tree Broadcast, recursive doubling Allreduce, and Bruck's Alltoall operations in the circulant network topologies can achieve the minimum collective communication hop count. Figures 5.3, 5.4 and 5.5 show that the hop count of these three collective communication operations on circulant network topology is the lowest. Although circulant network topology has a larger diameter and ASPL than random shortcut and Dragonfly network topologies, as illustrated in Section 4, the circulant network topology is highly suitable for collective communication operations. The hop counts of collective communication on circulant network topology are significantly lower than random shortcut and Dragonfly network

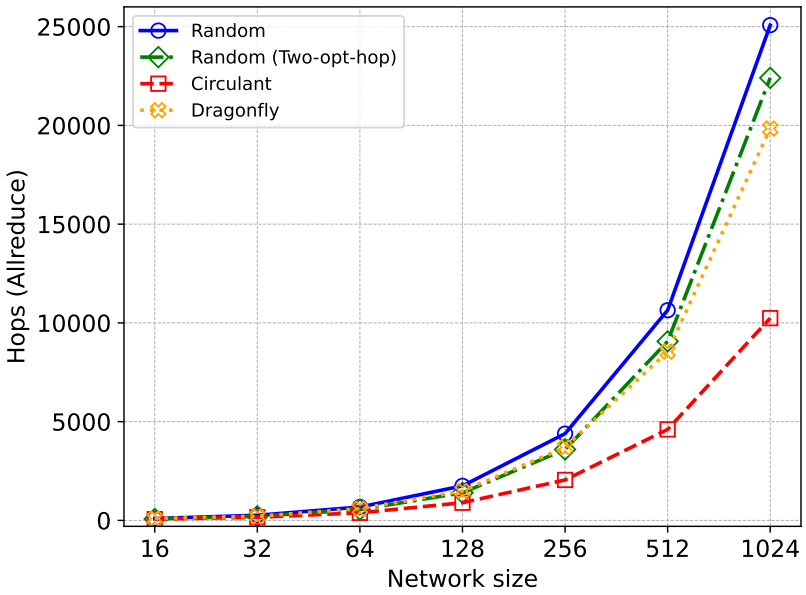


Figure 5.4: Hop counts of Allreduce operations for three network topologies.

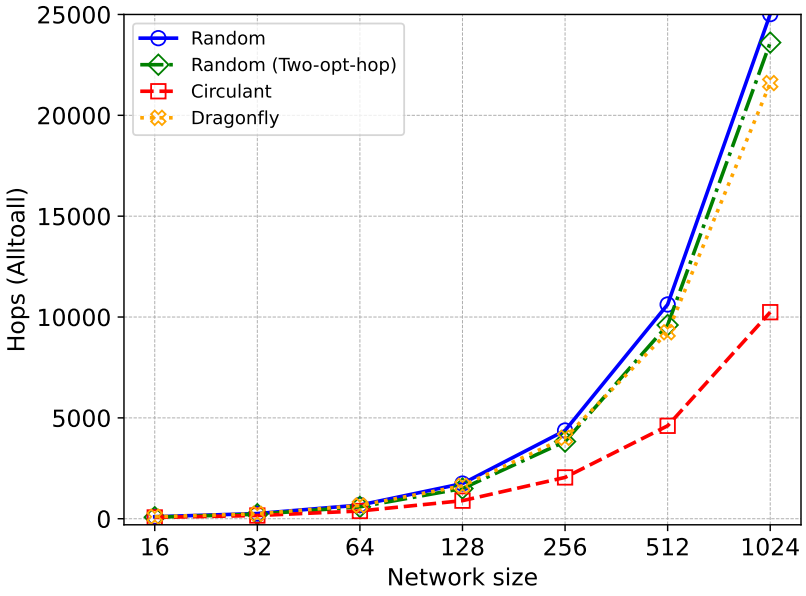


Figure 5.5: Hop counts of Alltoall operations for three network topologies.

topologies.

When the network size is 1,024, for the binomial-tree Broadcast operation, the hop

counts on circulant are 9% and 40% less than Dragonfly and random network topology. For the recursive doubling Allreduce operation, the hop count on circulant network topology is 48% and 59% less than Dragonfly and random network topologies. For Bruck's Alltoall operation, the hop count on circulant network topologies is 52% and 59% less than Dragonfly and random network topologies.

5.2.2 Performance of Collective Communication Operations

We evaluate the execution time of Broadcast, Allreduce, and Alltoall with different message sizes on the random shortcut, circulant, and Dragonfly network topologies. The parameters' setting of the network is the same as Chapters 3 and 4. The network size is set to 1,024, and 512 compute nodes are involved in the collective communication operations. We divide the message size into small and large groups. The small size includes 256B, 1KB, 4KB, and 16 KB. The large size has 64KB, 256KB, and 1MB. Figures 5.6 to 5.11 illustrates the evaluation results of all collective communication operations; the X-axis represents message size, and the Y-axis represents the execution time of collective communication operation.

Figures 5.6 and 5.7 illustrate the execution time of Broadcast operation with small and large message sizes. The results show that the Broadcast performance of ring-based consecutive mapping in Dragonfly is the worst. The random mapping in Dragonfly is better than the original random and ring-based consecutive mapping in random shortcut network topology but worse than them with two-opt hop rank placement optimization. When the message size becomes large, the Broadcast performance on circulant network topology using ring-based consecutive and circulant mapping is the best.

Figures 5.8 and 5.9 illustrate the execution time of the Allreduce operation with small and large message sizes. Similar to the Broadcast performance evaluation results, the Allreduce performance of ring-based consecutive mapping in Dragonfly is the worst. The random mapping on Dragonfly is better than the ring-based consecutive mapping on random shortcut network topology but worse than random mapping. With two-opt rank optimization, the random and ring-based consecutive mapping Allreduce operation on random shortcut network topology performs better than Dragonfly. The ring-based mapping Allreduce in circulant network topology is also the best.

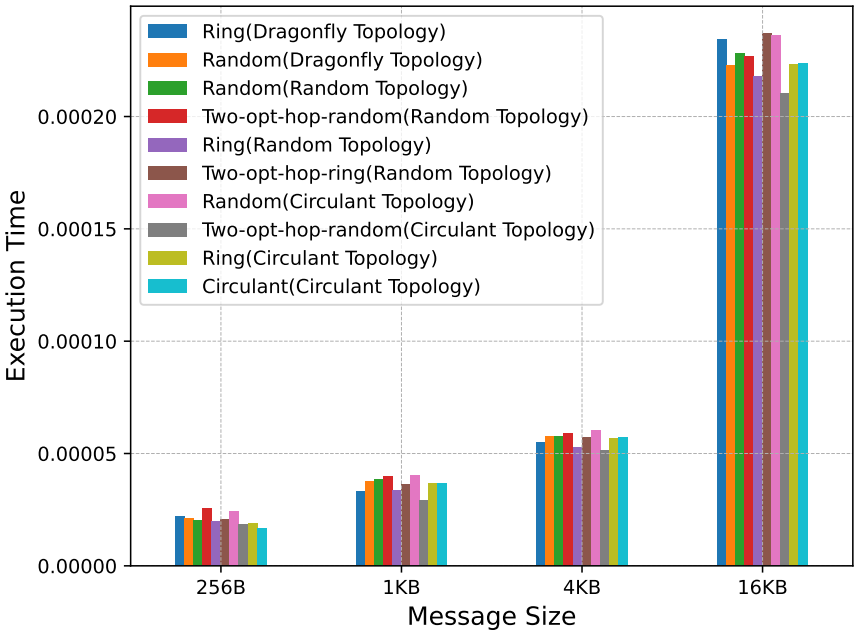


Figure 5.6: Execution time of Broadcast operations for different network topologies and mappings with small message size.

Figures 5.10 and 5.11 illustrate the execution time of the Alltoall operation with different message sizes. Similar to the Allreduce performance evaluation results, the Alltoall performance of ring-based consecutive mapping in Dragonfly is the worst. The random mapping on Dragonfly is better than the ring-based consecutive mapping on random shortcut network topology but worse than random mapping. With two-opt rank optimization, the random and ring-based consecutive mapping Alltoall operation on random shortcut network topology performs better than Dragonfly. The ring-based mapping Alltoall in circulant network topology is also the best.

The performance evaluations of Broadcast, Allreduce, and Alltoall operations show that the ring-based consecutive mapping in Dragonfly network topology is highly worse—the congestion of the global links in Dragonfly topology may be one of the reasons.

For a fair comparison, all topologies have the same degree. The degree d of random shortcut, circulant, and Dragonfly satisfies the condition: $d = 2\log_2 N - 1$. When the network size is 1024, the degree d is $d = 19$. We generate the 1024 network size

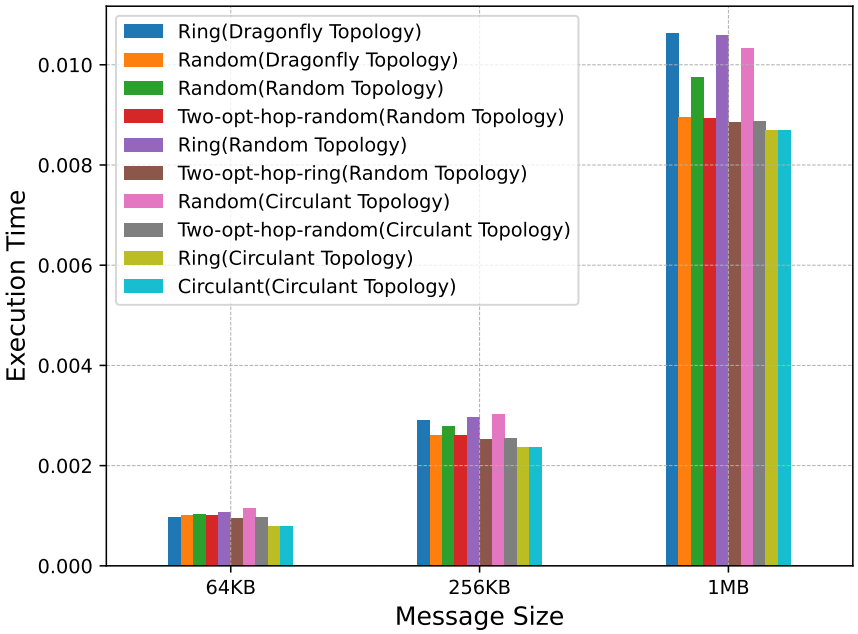


Figure 5.7: Execution time of Broadcast operations for different network topologies and mappings with large message size.

Dragonfly network topology this way, dividing the 1024 switches into 64 groups; each group has 16 switches, with each switch having 4 global links to switches in other groups; this case does not satisfy the load-balance [87] of Dragonfly network topology. The low number of global links may be responsible for the performance degradation of collective communication operations.

5.3 Point-to-Point Communication in Shortcut Network Topologies

This section compares hop counts and communication time of point-to-point communication in shortcut topologies with uniform random traffic patterns. Then, we analyze the total communication hop count by varying the point-to-point and collective communications ratio.

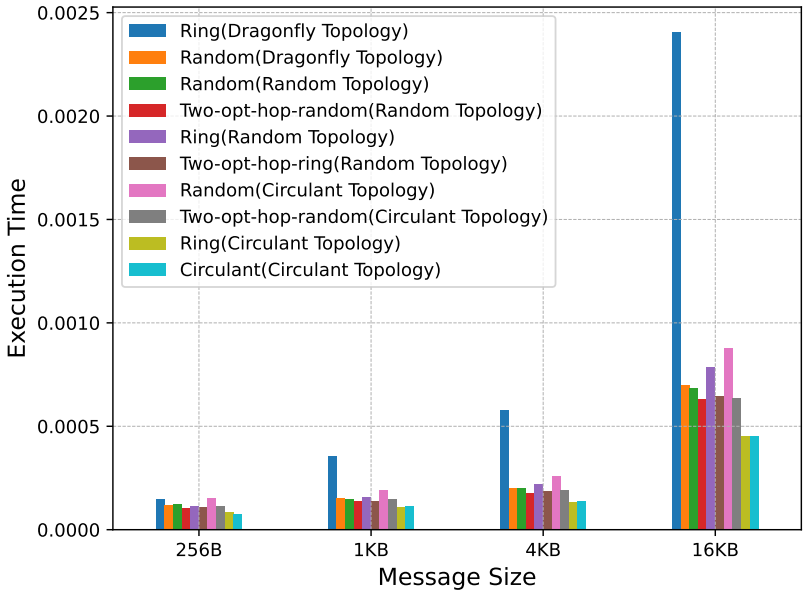


Figure 5.8: Execution time of Allreduce operations for different network topologies and mappings with small message size.

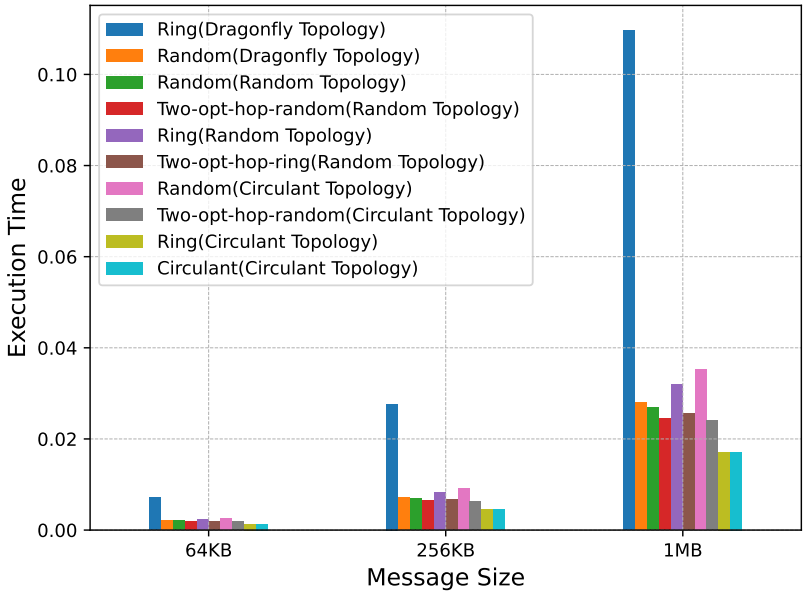


Figure 5.9: Execution time of Allreduce operations for different network topologies and mappings with large message size.

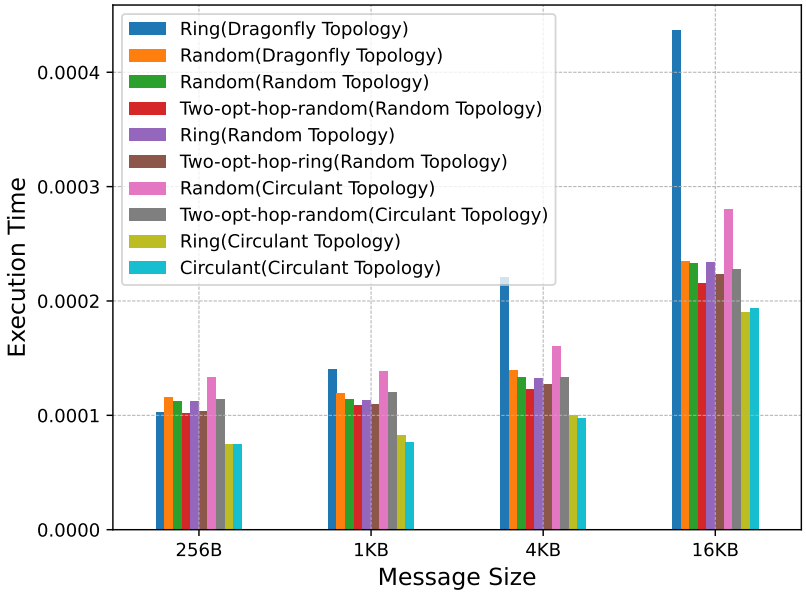


Figure 5.10: Execution time of Alltoall operations for different network topologies and mappings with small message size.

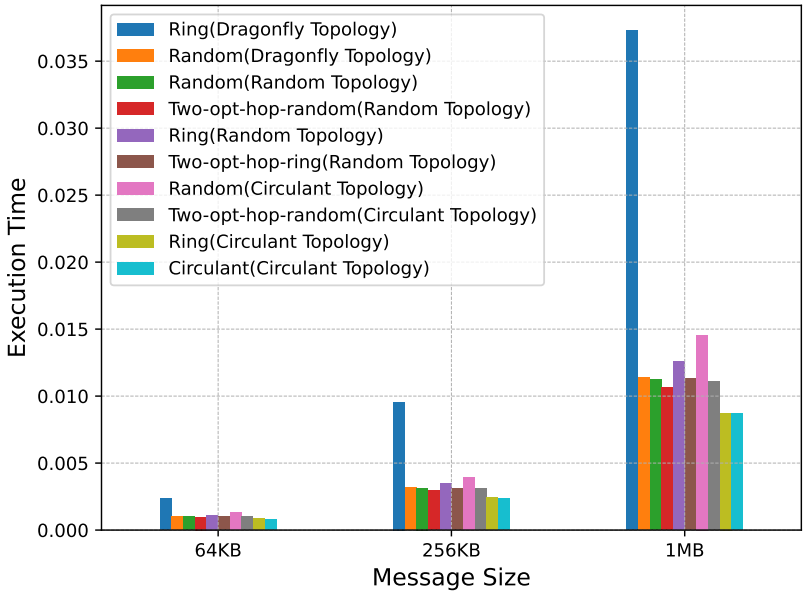


Figure 5.11: Execution time of Alltoall operations for different network topologies and mappings with large message size.

5.3.1 Hop count of Point-to-Point Communication

Figure 5.12 illustrates the average hops of point-to-point communication on random shortcut and circulant network topologies with uniform random traffic patterns. The number of switches is 1024, and the number of compute nodes is also 1024, assuming that one compute node is connected to each switch. The X-axis represents the number of mapped compute nodes; mapping strategies are used, which are introduced in Chapters 3 and 4. The Y-axis represents the average hop count. The average hop count is the mean number of hops taken by 100000 individual point-to-point communications.

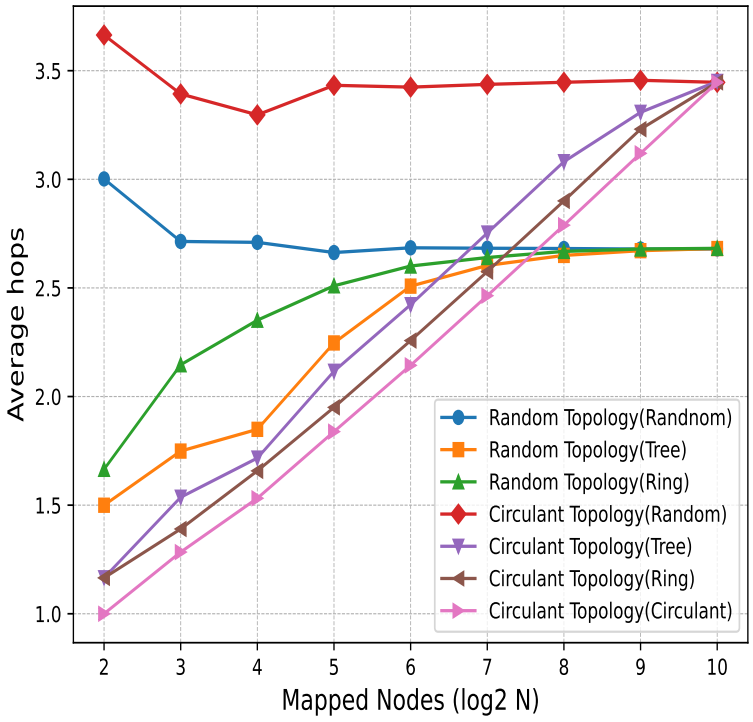


Figure 5.12: Average hop count of point-to-point communication

Section 5.1 illustrates that the circulant network topology has a higher diameter and ASPL than random shortcut network topology for the same degree. Intuitively, the point-to-point communication in circulant network topology may have a higher hop count. However, Figure 5.12 shows that even though circulant network topology has higher diameter and ASPL, the point-to-point communication in circulant network topology still archives lower hop count than random shortcut network topology in

many cases. As shown in Figure 5.12, when the network size is 1024, the average hop count of point-to-point communication in the circulant network topology is less than that of the random shortcut network topology when the mapped compute nodes are less than or equal to 128, using both circulant mapping and ring-based consecutive mapping.

5.3.2 Performance of Point-to-Point Communication

Section 5.1 describes using the circulant and ring-based consecutive mappings in circulant networks, where point-to-point communication has fewer hop counts than in random shortcut network topology when the number of mapped computational nodes is large, e.g., when the network size is 1024. The mapped nodes are less than 128.

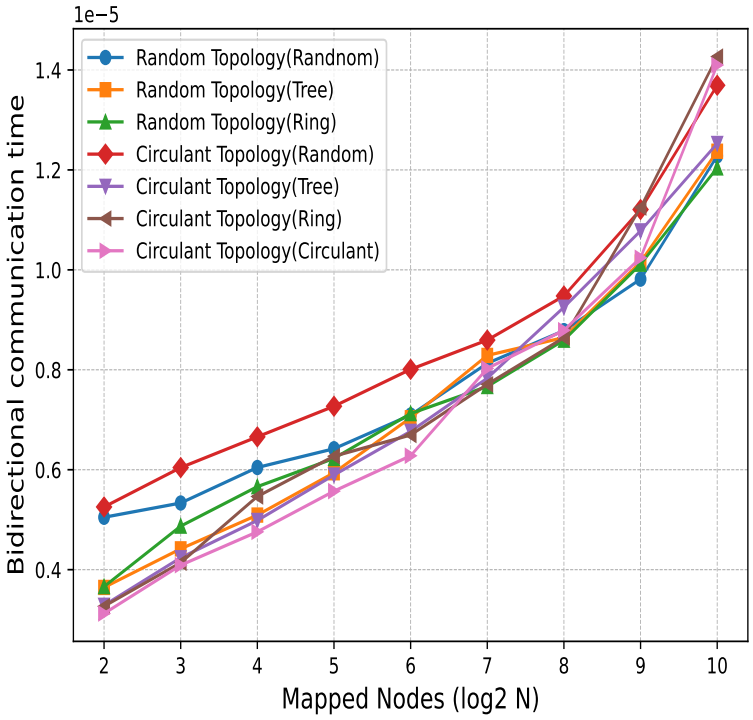


Figure 5.13: Bidirectional communication time of point-to-point communication.

In this section, we evaluate the bidirectional communication time of point-to-point communication in circulant and random shortcut network topologies by using the discrete-event simulator SimGrid (v3.28) [133]. The network is configured with the

same parameters as in Sections 3.4.3 and 4.5. The number of switches and compute nodes are 1024, the message size of point-to-point communication is 128 bytes, and the traffic pattern is random. The bidirectional time is the mean value of 10000 bidirectional point-to-point communications.

Figure 5.13 shows the bidirectional point-to-point communication evaluation result. The X-axis represents the bidirectional communication time; the Y-axis represents the number of mapped compute nodes. As Figure 5.13 shows, when the number of compute nodes is less than 128, point-to-point communication with circulant mapping in a circulant network topology outperforms point-to-point communication in random shortcut network topology. This result is consistent with the finding in Section 5.3.1 that point-to-point communication requires fewer hops than a random network topology in a circulant network topology that employs circulant mapping when there are not many mapped compute nodes. In the circulant network, point-to-point communication with random mapping experiences the lowest performance.

Figures 5.12 and 5.13 illustrate that, despite the circulant network topology's larger diameter and average shortest path length (ASPL) compared to the random shortcut network topology, the utilization of effective mapping strategies like ring-based consecutive mapping and circulant mapping can decrease the number of hops in point-to-point communication, leading to improved performance.

5.4 Total Hop Count of Communication in Shortcut Network Topologies

Sections 5.2 and 5.3 compared the hop counts of collective and point-to-point communication in circulant network and random shortcut network topologies, respectively. In this section, we would like to analyze the total communication hops of point-to-point communication and collective communication in circulant and random shortcut network topologies.

The number of switches and compute nodes are both 1024. We vary the number of compute nodes involved in communication from 8 to 1024 and the ratio of collective communication from 10%

Software-based collective communication operations are realized through a series

of point-to-point communications; we use m to represent the number of point-to-point communications within one collective communication operation. For a collective communication operation, its point-to-point communication count can be expressed like this:

$$m = \begin{cases} N-1, & \text{Broadcast} \\ N \log_2 N, & \text{Allreduce or Alltoall} \end{cases}$$

N is the number of compute nodes involved in the communication; assume that N is a power of 2. Considering that the Broadcast uses a binomial tree and that Allreduce and Alltoall use Bruck's algorithm, respectively. We use n to represent the number of collective communication operations and p to represent the number of point-to-point communications outside of collective communication operations. Then, the ratio of collective communication can be represented as:

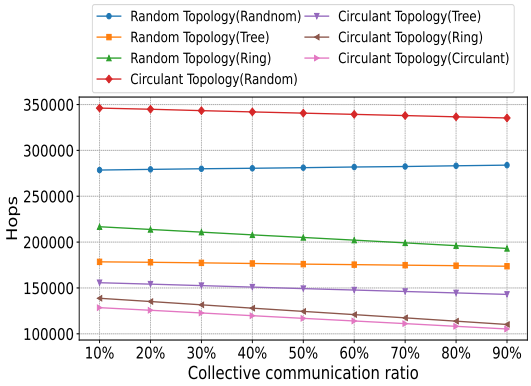
$$ratio = \frac{mn}{mn+p}$$

We control the total number of communications to be 102400, i.e., $mn + p = 102400$, changing the ratio of collective communications by adjusting the number of collective communication operations n.

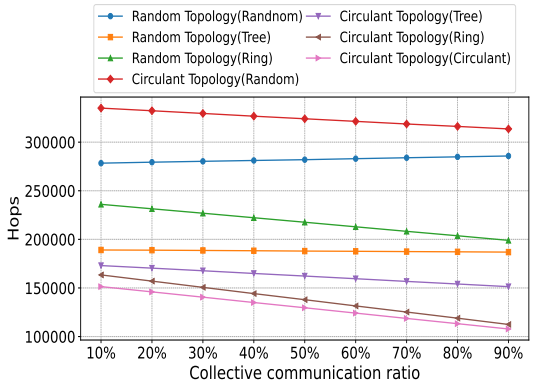
Figure 5.14 illustrates how the total number of hops varies with the collective communication ratio. The X-axis represents the collective communication ratio, and the Y-axis represents the total communication hop count. The collective communication operation is Alltoall; the mapping strategies use the one described in the previous chapter, and a random traffic pattern is used for point-to-point communication.

The evaluation results in Figure 5.14 show that varying the collective communication ratio on circulant or random shortcut network topology does not change the total number of communication hops when using random mapping. Other mapping strategies, such as a hierarchical tree, ring-based consecutive mapping, and circulant mapping, can reduce the total number of communication hops when the ratio of collective communication increases. In particular, ring-based consecutive mapping and circulant mapping on circulant network topology drastically reduce the number of hops of communication when the ratio of collective communication increases.

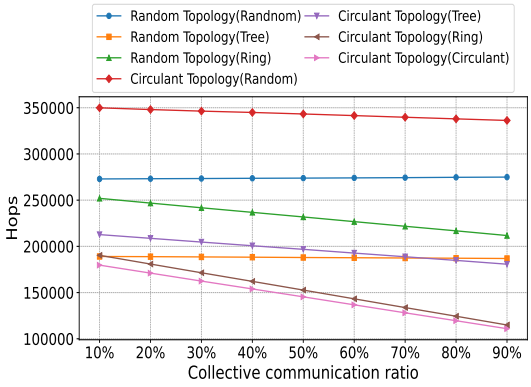
Figure 5.14(a) to Figure 5.14(e) show that when the number of compute nodes



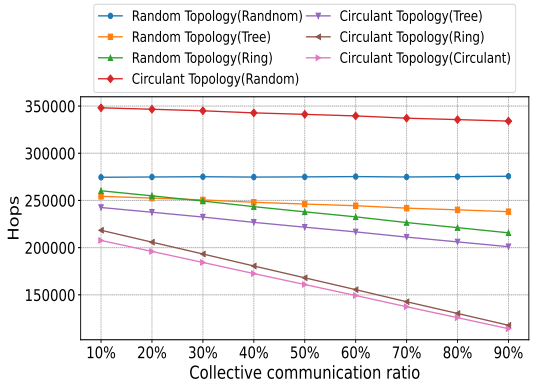
(a) 8 compute nodes involved in communication



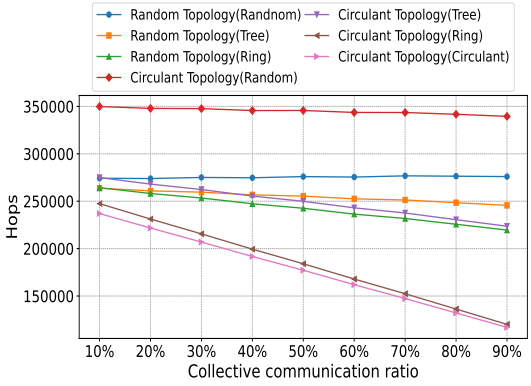
(b) 16 compute nodes involved in communication



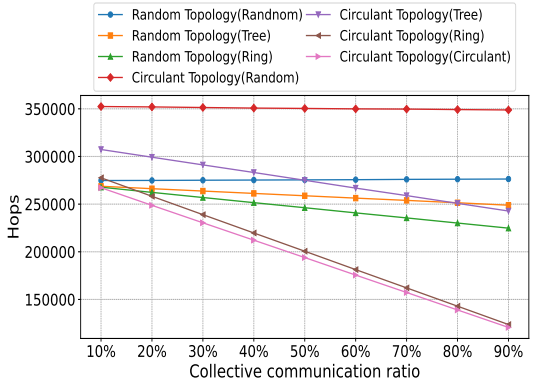
(c) 32 compute nodes involved in communication



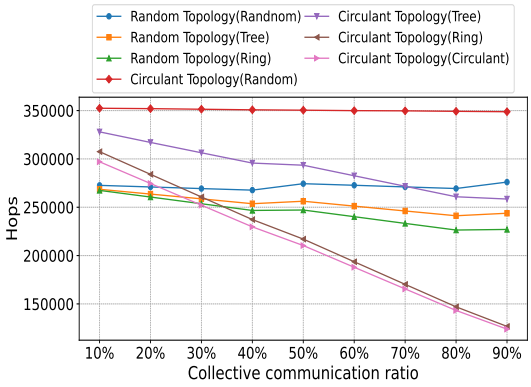
(d) 64 compute nodes involved in communication



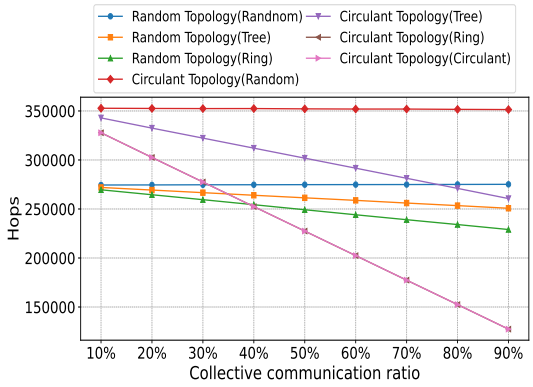
(e) 128 compute nodes involved in communication



(f) 256 compute nodes involved in communication



(g) 512 compute nodes involved in communication



(h) 1024 compute nodes involved in communication

Figure 5.14: Communication hop count vs. Collective Communication ratio.

involved in communication is small, e.g., less than or equal to 128, the number of communication hops using ring-based consecutive mapping and circulant mapping in a circulant network is always smaller than that in random shortcut network topology, no matter how the ratio of collective communication is changed.

Figure 5.14(f) to Figure 5.14(h) show that as the number of computing nodes involved in communication increases, e.g., greater than or equal to 256, the total number of hops in the random shortcut network topology is less than that in the circulant network topology when the ratio of collective communication is small. As the ratio of collective communication increases, the total number of hops in the circulant network topology starts to be less than that in the random shortcut topology. When all the compute nodes are involved in the communication, i.e., the number of compute nodes involved in the communication is 1024, when the ratio of collective communication is more than 50%, the total number of hops in the circulant network topology starts to be less than that in random shortcut topology.

5.5 Performance of Parallel Applications

In this section, we compare the performance of applications in circulant and random shortcut network topologies. The network parameters are the same as the evaluation in Sections 3.4.3 and 4.5.3. We evaluate the applications in NAS parallel benchmark [130], which are FT, IS, MG, and LU. The problem size is Class A, the number of processes is 128, using the mapping strategies described in the previous chapters, and the iteration times of the two-opt approach is 20,0000. Broadcast, Allreduce, and Alltoall algorithms are binomial trees, recursive doubling, and Bruck's ones [99]. For random shortcut topologies, use random, ring-based consecutive mapping strategies and corresponding two-opt rank placement optimization approach, and for circulant network topology, use ring-based and circulant mapping strategies.

Figure 5.15 illustrates the performance comparison of applications FT. The performance of FT using ring-based consecutive and circulant mapping strategies in a circulant network topology outperforms the performance in a random shortcut network topology. The work [134] demonstrates that random topologies with low ASPL are suitable for HPC applications, especially FT. The application of FT extensively uses Alltoall operations, and the low diameter and ASPL properties allow random

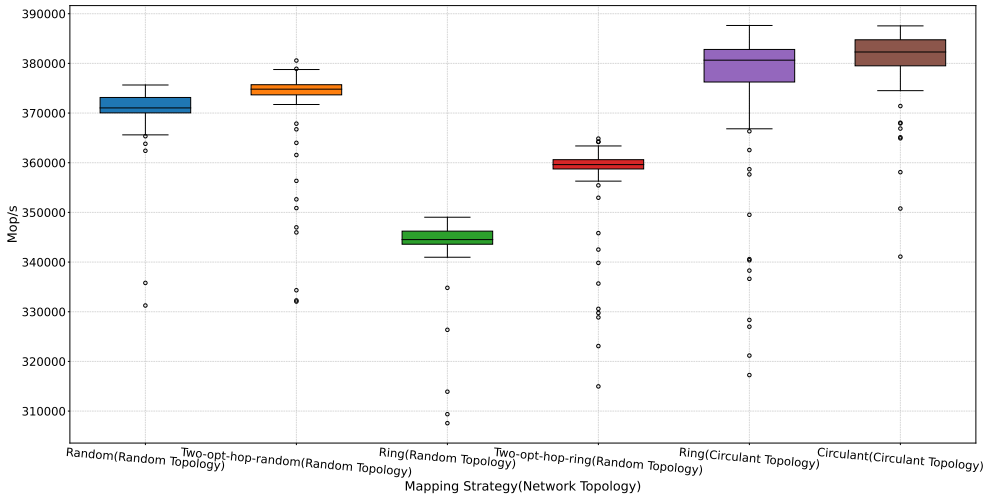


Figure 5.15: Performance evaluation of application FT in shortcut network topologies.

shortcut network topology to achieve efficient FT execution. However, FT performs even better in circulant network topology than random shortcut network topology. The Alltoall operations using Bruck's algorithm can achieve low hop counts in circulant network topology, resulting in higher performance.

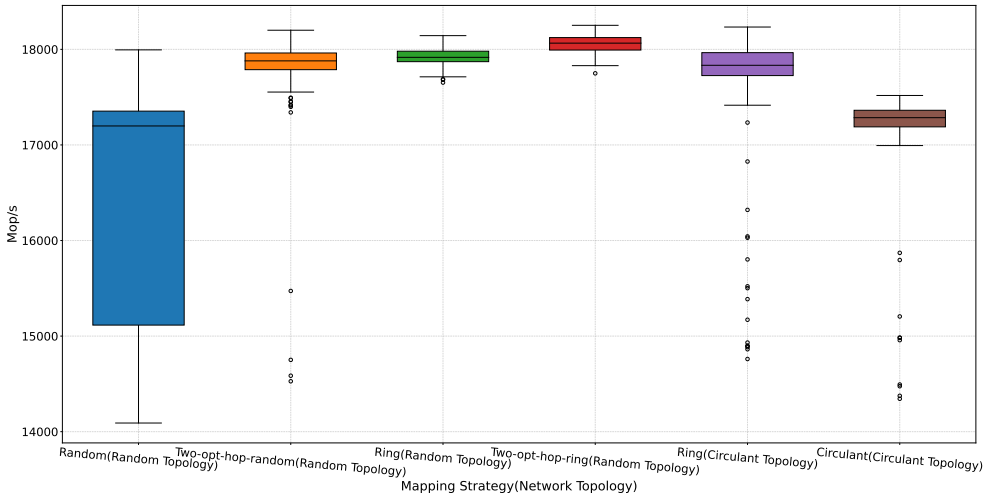


Figure 5.16: Performance evaluation of application IS in shortcut network topologies.

Figure 5.16 illustrates the performance comparison of applications IS. The ring-based consecutive mapping with two-opt rank placement optimization in random

shortcut network topology performs best. Using a ring-based consecutive mapping strategy is slightly better than circulant network topology.

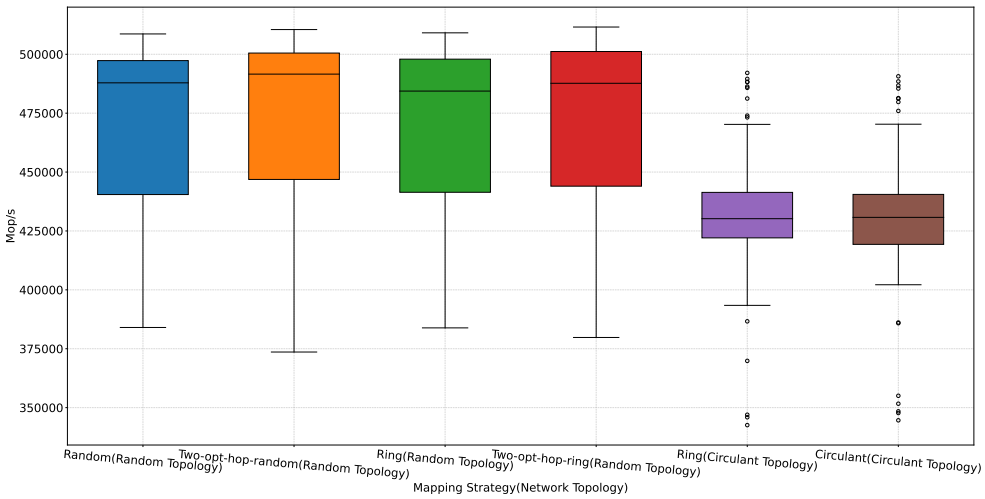


Figure 5.17: Performance evaluation of application MG in shortcut network topologies.

Figure 5.17 illustrates the performance comparison of applications MG. The MG makes extensive use of point-to-point communication. The evaluation results show in Figure 5.17 that MG performs better in a random shortcut network topology than in a circulant network topology.

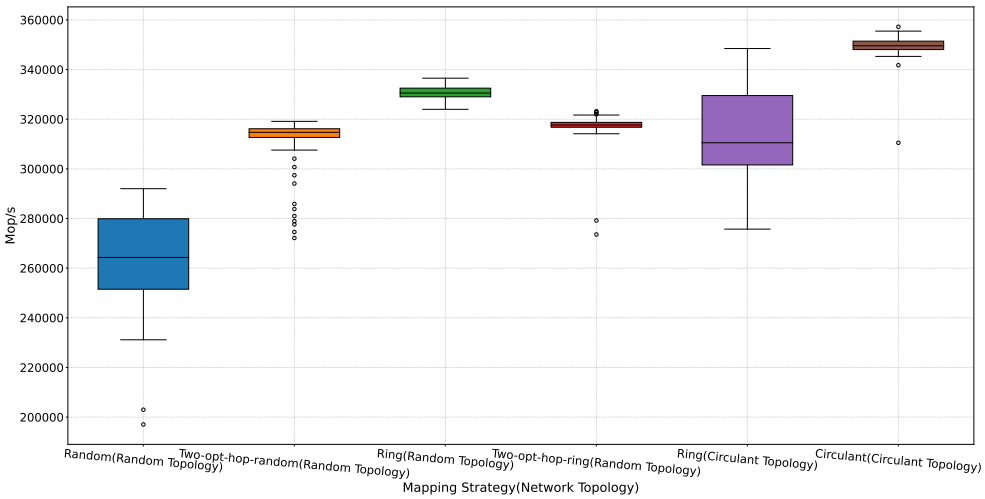


Figure 5.18: Performance evaluation of application LU in shortcut network topologies.

Figure 5.18 presents a performance comparison of the LU application. LU involves numerous neighboring communications, and the ring-based consecutive mapping ensures that any two neighboring compute nodes are just one hop away, making this mapping strategy highly efficient for LU. Additionally, the circulant mapping in the embedded circulant topology also ensures that any two neighboring compute nodes are just one hop apart; the evaluation results in Figure 5.18 show that the circulant mapping in circulant network topology has the best performance.

5.6 Cost Analysis of Shortcut Network Topologies

In previous sections, we compared the performance of collective communication, point-to-point communication, and parallel applications in circulant and random shortcut network topologies. In this section, we would like to compare the cable length and cost of constructing an interconnection network with these two shortcut network topologies.

Modern supercomputers typically employ a TOR (Top of Rack) network architecture, where multiple switches and compute nodes are placed within a single cabinet. Then, the cabinets are interconnected using cables. We assume each cabinet is 0.6 meters wide and 2.1 meters deep [135]. Each cabinet has 8 switches, and the physical layout of cabinets is the 2-D grid. The 2-D grid has r rows and c columns. The number of rows r and columns satisfy $r = \lceil \sqrt{n} \rceil$ and $c = \lceil n/r \rceil$. The intra-cabinet cable overhead is 100cm, and the inter-cabinet cable overhead is 200cm, using the Manhattan distance to represent the distance between two cabinets. We ignore the cable overhead between the compute nodes and the connecting switch. We use the method in [136] to calculate the total cable overhead.

Figure 5.19 illustrates the comparison of average cable length between the circulant network and random shortcut network. The x-axis represents the network size, and the y-axis represents the cable length. As the network size increases, the average cable length of the random shortcut topology rises rapidly. Circular network topology has a much smaller average cable length than random shortcut network topology for the same network size. When the network size is 1024, i.e., the number of switches is 1024, the cable length of circulant topology is 46% less than random shortcut topology.

Then, we compare the cost of the circulant network topology and random shortcut

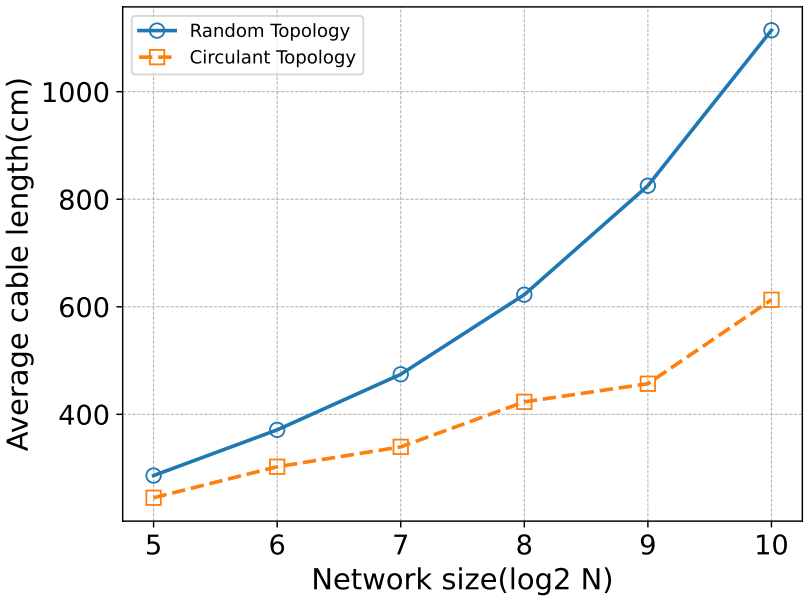


Figure 5.19: Comparison of average cable length between circulant and random shortcut network topologies.

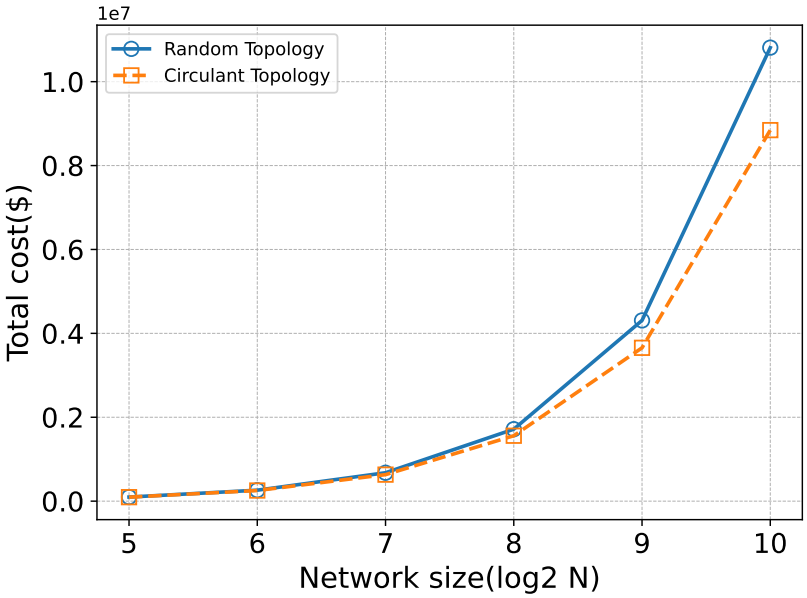


Figure 5.20: Comparison of total cost between circulant and random shortcut network topologies.

network topology. We use the cost model in [13], where the total cost includes the switches' and cable costs. The switch's cost is related to the number of switches and the unit price of the switch. The cable cost is related to cable length and bandwidth; we assume the cable bandwidth is 100Gbps.

Figure 5.20 illustrates the total cost comparison between the circulant and random shortcut network topologies. The x-axis represents the network size, and the y-axis represents the total cost. We assume that for a circulant topology and random topology with the same network size, the number of switches and the unit price are the same, i.e., the cost of the switches is the same, and the difference in the total cost is determined by cable length. Figure 5.20 illustrates that as the network size grows, the effect of the cost of cable length on the total cost becomes significant; the cable length highly affects the network cost. When the network size is 1024, i.e., the number of switches is 1024, the total cost of circulant topology is 18% less than random shortcut topology. In this context, the circulant network topology is better than the random shortcut network topology, especially in the case of large-scale networks.

6 Conclusions and Future Direction

6.1 Conclusions

Applications on large parallel computers face the challenge of solving challenging problems, such as weather forecasting and finding drugs. In parallel applications, compute nodes communicate with each other. Then, collective communication is widely used in these massively parallel applications and affects their performance.

Improving the performance of collective communication is an essential way to shorten the execution time of the application of parallel computers. Both hardware and software levels influence the performance of collective communication. At the hardware level, an essential factor is the network interconnection. In a low-diameter network topology network interconnection, the delay of messages from one computing node to another becomes low. Reducing the communication delay between nodes can improve the overall performance of collective communication. Implementing algorithms and process/job mapping affect collective communication performance at the software level.

In this dissertation, we co-design network topology, process mapping, and processes' rank placement strategies to improve collective communication performance on random shortcut and circulant network topology.

Firstly, we applied the two-opt approach for random shortcut network topology to re-place processes' rank for building efficient collective communication. The discrete-event simulation results significantly enhance collective communication performance using the two-opt approach. This improvement is achieved by reducing the hops of collective communication operations. We also evaluate the performance of parallel applications in random shortcut network topology. In applications where collective communication dominates performance, the two-opt approach dramatically improves application performance through efficient collective communication.

Secondly, we proposed circulant process mapping to implement efficient collective communication in circulant network topology. Although compared to the random shortcut network topology, the circulant network topology has a larger diameter and average shortest path length, the circulant network topology is more suitable for collective communication. The collective communication operations such as Broadcast, Allreduce, and Alltoall can achieve a minimum number of hops in circulant network topology. Moreover, we proposed a process mapping strategy named circulant mapping for circulant network topology. Combining the circulant network topology and the circulant process mapping strategy can achieve significantly efficient collective communication.

Finally, we compared these two shortcut network topologies in depth. The random shortcut network topology has a lower diameter and average shortest path length(ASPL). Intuitively, communication within this topology involves fewer hops. However, combining a suitable mapping strategy with a circulant network topology results in lower hop counts and higher efficiency for collective communication. In some cases, such as fewer nodes involved in the communication, the number of hops for point-to-point communication in a circulant network is even less. We also analyze the cost of these two network topologies for building an interconnection network for parallel computers; the results show that using a circuit network topology to make an interconnection network has less cost.

In scenarios where collective communication is pivotal in performance, opting for a circulant network yields outstanding results. Conversely, when point-to-point

communication holds sway, a shortcut network with a lower diameter and average shortest path length (ASPL) offers a distinct advantage. We recommend using the second approach when target applications heavily rely on collective communications. In the other cases, we recommend the first approach for a fast execution of parallel applications.

6.2 Future Directions

The dissertation illustrates that efficient collective communication efficiently combines network topology and process mapping. We analyze the performance of collective communications in various environments. Our future work is their implementation on real parallel computers and measuring the performance of parallel applications using collective communications.

More generally, network topology, routing, switching technique, and flow control are the heart of interconnection-network design on a parallel computer. Thus, they affect the router architecture [98], and there are a large number of their prior works. However, we expect that the next-generation interconnection network should care about the design of collective communication. For example, it is reported that 15 main applications out of 64 consume 60% or more execution time by MPI communication in a production supercomputer [16]. We believe that our findings in this study will contribute to the network design efficiently in such a collective-communication-centric era.

Bibliography

[1] HPCI Roadmap. http://www.open-supercomputer.org/workshop/sdhpc/.

[2] Michihiro Koibuchi, Hiroki Matsutani, Hideharu Amano, D. Frank Hsu, and Henri Casanova. A Case for Random Shortcut Topologies for HPC Interconnects. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 177–188, 2012.

[3] Ke Cui and Michihiro Koibuchi. Efficient two-opt collective-communication operations on low-latency random network topologies. *IEICE Transactions on Information and Systems*, 103(12):2435–2443, 2020.

[4] Luca Fedeli, Axel Huebl, France Boillod-Cerneux, Thomas Clark, Kevin Gott, Conrad Hillairet, Stephan Jaure, Adrien Leblanc, Rémi Lehe, Andrew Myers, et al. Pushing the frontier in the design of laser-based electron accelerators with groundbreaking mesh-refined particle-in-cell simulations on exascale-class supercomputers. In *2022 SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 25–36. IEEE Computer Society, 2022.

[5] Craig B Stunkel, Richard L Graham, Gilad Shainer, Michael Kagan, SS Sharkawi, B Rosenburg, and GA Chochia. The high-speed networks of the summit and sierra supercomputers. *IBM Journal of Research and Development*, 64(3/4):3–1, 2020.

[6] NVIDIA InfiniBand Switches . https://www.nvidia.com/en-us/networking/infiniband-switching/.

[7] William James Dally and Brian Patrick Towles. *Principles and practices of interconnection networks*. Elsevier, 2004.

[8] Kenji Mizutani, Hiroshi Yamaguchi, Yutaka Urino, and Michihiro Koibuchi. Optweb: A lightweight fully connected inter-fpga network for efficient collectives. *IEEE Transactions on Computers*, 70(6):849–862, 2021.

[9] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440, 1998.

[10] Michihiro Koibuchi, Ikki Fujiwara, Kiyo Ishii, Shu Namiki, Fabien Chaix, Hiroki Matsutani, Hideharu Amano, and Tomohiro Kudoh. Optical network technologies for hpc: computer-architects point of view. *IEICE Electronics Express*, 13(6):20152007–20152007, 2016.

[11] GraphGolf: The Order/degree Problem Competition. GraphGolf: The Order/degree Problem Competition. http://research.nii.ac.jp/graphgolf/.

[12] John Kim, Wiliam J. Dally, Steve Scott, and Dennis Abts. Technology-Driven, Highly-Scalable Dragonfly Topology. In *ISCA*, pages 77–88, 2008.

[13] Maciej Besta and Torsten Hoefler. Slim fly: A cost effective low-diameter network topology. In *SC'14: proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 348–359. IEEE, 2014.

[14] Top 500, November 2022. https://www.top500.org/lists/top500/2022/11/.

[15] https://spectrum.ieee.org/frontier-exascale-supercomputer. https://spectrum.ieee.org/frontier-exascale-supercomputer.

[16] Sudheer Chunduri, Scott Parker, Pavan Balaji, Kevin Harms, and Kalyan Kumaran. Characterization of mpi usage on a production supercomputer. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 386–400. IEEE, 2018.

[17] Rajeev Thakur, Rolf Rabenseifner, and William Gropp. Optimization of collective communication operations in mpich. *The International Journal of High Performance Computing Applications*, 19(1):49–66, 2005.

[18] Amith R Mamidala, Rahul Kumar, Debraj De, and Dhabaleswar K Panda. Mpi collectives on modern multicore clusters: Performance optimizations and communication characteristics. In *2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pages 130–137. IEEE, 2008.

[19] Ernie Chan, Marcel Heimlich, Avi Purkayastha, and Robert Van De Geijn. Collective communication: theory, practice, and experience. *Concurrency and Computation: Practice and Experience*, 19(13):1749–1783, 2007.

[20] MPI Forum. http://mpi-forum.org/.

[21] Open MPI: Open Source High Performance Computing. https://www.open-mpi.org/.

[22] MPICH | High-Performance Portable MPI. https://www.mpich.org/,.

[23] MVAPICH :: Home. http://mvapich.cse.ohio-state.edu/.

[24] Intel MPI Library. http://software.intel.com/en-us/intel-mpi-library/.

[25] Fabrizio Petrini, Salvador Coll, Eitan Frachtenberg, and Adolfy Hoisie. Hardware- and software-based collective communication on the quadrics network. In *Proceedings IEEE International Symposium on Network Computing and Applications. NCA 2001*, pages 24–35. IEEE, 2001.

[26] Philip K McKinley, Yih-jia Tsai, and David F Robinson. Collective communication in wormhole-routed massively parallel computers. *Computer*, 28(12):39–50, 1995.

[27] Jiayi Sheng, Qingqing Xiong, Chen Yang, and Martin C Herbordt. Collective communication on fpga clusters with static scheduling. *ACM SIGARCH Computer Architecture News*, 44(4):2–7, 2017.

[28] Richard L Graham, Devendar Bureddy, Pak Lui, Hal Rosenstock, Gilad Shainer, Gil Bloch, Dror Goldenerg, Mike Dubman, Sasha Kotchubievsky, Vladimir

Koushnir, et al. Scalable hierarchical aggregation protocol (sharp): a hardware architecture for efficient data reduction. In *2016 First International Workshop on Communication Optimizations in HPC (COMHPC)*, pages 1–10. IEEE, 2016.

[29] Bharath Ramesh, Kaushik Kandadi Suresh, Nick Sarkauskas, Mohammadreza Bayatpour, Jahanzeb Maqbool Hashmi, Hari Subramoni, and Dhabaleswar K. Panda. Scalable mpi collectives using sharp: Large scale performance evaluation on the tacc frontera system. In *2020 Workshop on Exascale MPI (ExaMPI)*, pages 11–20, 2020.

[30] Robert Alverson, Duncan Roweth, and Larry Kaplan. The gemini system interconnect. In *2010 18th IEEE Symposium on High Performance Interconnects*, pages 83–87. IEEE, 2010.

[31] Bob Alverson, Edwin Froese, Larry Kaplan, and Duncan Roweth. Cray xc series network. *Cray Inc., White Paper WP-Aries01-1112*, 2012.

[32] W Michael Brown, Trung D Nguyen, Miguel Fuentes-Cabrera, Jason D Fowlkes, Philip D Rack, Mark Berger, and Arthur S Bland. An evaluation of molecular dynamics performance on the hybrid cray xk6 supercomputer. *Procedia Computer Science*, 9:186–195, 2012.

[33] Abhinav Bhatelé, Laxmikant V Kalé, and Sameer Kumar. Dynamic topology aware load balancing algorithms for molecular dynamics applications. In *Proceedings of the 23rd international conference on Supercomputing*, pages 110–116, 2009.

[34] James C Phillips, Yanhua Sun, Nikhil Jain, Eric J Bohm, and Laxmikant V Kalé. Mapping to irregular torus topologies and other techniques for petascale biomolecular simulation. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 81–91. IEEE, 2014.

[35] Tarun Agarwal, Amit Sharma, A Laxmikant, and Laxmikant V Kalé. Topology-aware task mapping for reducing communication contention on large parallel machines. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, pages 10–pp. IEEE, 2006.

[36] Mehmet Deveci, Kamer Kaya, Bora Uçar, and Ümit V Çatalyürek. Fast and high quality topology-aware task mapping. In *2015 IEEE International Parallel and Distributed Processing Symposium*, pages 197–206. IEEE, 2015.

[37] Yiannis Georgiou, Emmanuel Jeannot, Guillaume Mercier, and Adèle Villiermet. Topology-aware job mapping. *The International Journal of High Performance Computing Applications*, 32(1):14–27, 2018.

[38] Hao Yu, I-Hsin Chung, and Jose Moreira. Topology mapping for blue gene/l supercomputer. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, pages 116–es, 2006.

[39] Abhinav Bhatele, Nikhil Jain, Katherine E Isaacs, Ronak Buch, Todd Gamblin, Steven H Langer, and Laxmikant V Kale. Optimizing the performance of parallel applications on a 5d torus via task mapping. In *2014 21st International Conference on High Performance Computing (HiPC)*, pages 1–10. IEEE, 2014.

[40] Jingjin Wu, Xuanxing Xiong, Eduardo Berrocal, Jia Wang, and Zhiling Lan. Topology mapping of irregular parallel applications on torus-connected supercomputers. *The Journal of Supercomputing*, 73:1691–1714, 2017.

[41] Abhinav Bhatele, Todd Gamblin, Steven H Langer, Peer-Timo Bremer, Erik W Draeger, Bernd Hamann, Katherine E Isaacs, Aaditya G Landge, Joshua A Levine, Valerio Pascucci, et al. Mapping applications with collectives over sub-communicators on torus networks. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE, 2012.

[42] Samuel D Pollard, Nikhil Jain, Stephen Herbein, and Abhinav Bhatele. Evaluation of an interference-free node allocation policy on fat-tree clusters. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 333–345. IEEE, 2018.

[43] Yi-shui Li, Xin-hai Chen, Jie Liu, Bo Yang, Chun-ye Gong, Xin-biao Gan, Shengguo Li, and Han Xu. Ohtma: an optimized heuristic topology-aware mapping algorithm on the tianhe-3 exascale supercomputer prototype. *Frontiers of Information Technology & Electronic Engineering*, 21:939–949, 2020.

[44] Yao Hu and Michihiro Koibuchi. The case for disjoint job mapping on high-radix networked parallel computers. In *Algorithms and Architectures for Parallel Processing: 21st International Conference, ICA3PP 2021, Virtual Event, December 3–5, 2021, Proceedings, Part II*, pages 123–143. Springer, 2022.

[45] Bogdan Prisacari, German Rodriguez, Philip Heidelberger, Dong Chen, Cyriel Minkenberg, and Torsten Hoefler. Efficient task placement and routing of nearest neighbor exchanges in dragonfly networks. In *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*, pages 129–140, 2014.

[46] Ozan Tuncer, Yijia Zhang, Vitus J Leung, and Ayse K Coskun. Task mapping on a dragonfly supercomputer. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2017.

[47] Xin Wang, Misbah Mubarak, Xu Yang, Robert B Ross, and Zhiling Lan. Trade-off study of localizing communication and balancing network traffic on a dragonfly system. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1113–1122. IEEE, 2018.

[48] Ke Cui and Michihiro Koibuchi. A high-radix circulant network topology for efficient collective communication. *the 23rd International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT' 22)*, 12 2022.

[49] Ming T Liu. Distributed loop computer networks. In *Advances in computers*, volume 17, pages 163–221. Elsevier, 1978.

[50] Jean-Claude Bermond, Francesc Comellas, and D. Frank Hsu. Distributed loop computer-networks: a survey. *Journal of parallel and distributed computing*, 24(1):2–10, 1995.

[51] W. D. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2003.

[52] Shuai Wang, Dan Li, Jinkun Geng, Yue Gu, and Yang Cheng. Impact of network topology on the performance of dml: Theoretical analysis and practical factors.

In *IEEE INFOCOM 2019-IEEE conference on computer communications*, pages 1729–1737. IEEE, 2019.

[53] Xiaolong Huang, Alexandre F Ramos, and Yuefan Deng. Optimal circulant graphs as low-latency network topologies. *arXiv preprint arXiv:2201.01342*, 2022.

[54] Muriel Médard and Steven S Lumetta. Network reliability and fault tolerance. *Encyclopedia of Telecommunications*, 2003.

[55] Yang Liu, Jogesh K Muppala, Malathi Veeraraghavan, Dong Lin, and Mounir Hamdi. *Data center networks: Topologies, architectures and fault-tolerance characteristics*. Springer Science & Business Media, 2013.

[56] Cristóbal Camarero, Carmen Martínez, Enrique Vallejo, and Ramón Beivide. Projective networks: Topologies for large parallel computer systems. *IEEE Transactions on Parallel and Distributed Systems*, 28(7):2003–2016, 2016.

[57] Charles Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32(2):406–424, 1953.

[58] Steve Scott, Dennis Abts, John Kim, and William J Dally. The blackwidow high-radix clos network. *ACM SIGARCH Computer Architecture News*, 34(2):16–28, 2006.

[59] C. E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, 34(10):892–901, 1985.

[60] Vaclav E Benes. *Mathematical theory of connecting networks and telephone traffic*. Elsevier, 1965.

[61] John Kim, William J. Dally, and Dennis Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. In *Proc, of the International Symposium on Computer Architecture (ISCA)*, pages 126–137, 2007.

[62] Top 500. https://www.top500.org/lists/top500/2021/11/.

[63] Rüdiger Esser and Renate Knecht. Intel paragon xp/s-architecture and software environment. In *Supercomputer' 93: Anwendungen, Architekturen, Trends Seminar, Mannheim, 24.–26. Juni 1993*, pages 121–141. Springer, 1993.

[64] Michael D Noakes, Deborah A Wallach, and William J Dally. The j-machine multicomputer: An architectural evaluation. *ACM SIGARCH Computer Architecture News*, 21(2):224–235, 1993.

[65] Michael S Warren, John K Salmon, Donald J Becker, M Patrick Goda, Thomas Sterling, and W Winckelmans. Pentium pro inside: I. a treecode at 430 gigaflops on asci red, ii. price/performance of $50/mflop$ on loki and hyglac. In *SC'97: Proceedings of the 1997 ACM/IEEE Conference on Supercomputing*, pages 61–61. IEEE, 1997.

[66] Craig Peterson, James Sutton, and Paul Wiley. iwarp: a 100-mops, liw microprocessor for multicomputers. *IEEE Micro*, 11(3):26–29, 1991.

[67] Richard E Kessler and James L Schwarzmeier. Cray t3d: A new dimension for cray research. In *Digest of Papers. COMPCON Spring*, pages 176–182. IEEE, 1993.

[68] Steven L Scott. The cray t3e network: adaptive routing in a high performance 3d torus. *Proceedings of Hot Interconnects IV, 1996*, pages 147–156, 1996.

[69] Narasimha R Adiga, George Almási, George S Almasi, Yariv Aridor, Rajkishore Barik, D Beece, Ralph Bellofatto, Gyan Bhanot, Randy Bickford, M Blumrich, et al. An overview of the bluegene/l supercomputer. In *SC'02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, pages 60–60. IEEE, 2002.

[70] Gheorghe Almasi, Sameh Asaad, Ralph E Bellofatto, H Randall Bickford, Matthias A Blumrich, Bernard Brezzo, Arthur A Bright, Jose R Brunheroto, Jose G Castanos, Dong Chen, et al. Overview of the ibm blue gene/p project. *IBM Journal of Research and Development*, 52(1-2):199–220, 2008.

[71] Courtenay Vaughan, Mahesh Rajan, Richard Barrett, Doug Doerfler, and Kevin Pedretti. Investigating the impact of the cielo cray xe6 architecture on scientific application codes. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pages 1831–1837. IEEE, 2011.

[72] Buddy Bland. Titan-early experience with the titan system at oak ridge national laboratory. In *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, pages 2189–2211. IEEE, 2012.

[73] Dong Chen, Noel A Eisley, Philip Heidelberger, Robert M Senger, Yutaka Sugawara, Sameer Kumar, Valentina Salapura, David L Satterfield, Burkhard Steinmacher-Burow, and Jeffrey J Parker. The ibm blue gene/q interconnection network and message unit. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–10, 2011.

[74] Yuichiro Ajima, Shinji Sumimoto, and Toshiyuki Shimizu. Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers. *IEEE Computer*, 42:36–40, 2009.

[75] Jack Dongarra. Report on the fujitsu fugaku system. *University of Tennessee-Knoxville Innovative Computing Laboratory, Tech. Rep. ICLUT-20-06*, 2020.

[76] Yuichiro Ajima, Takahiro Kawashima, Takayuki Okamoto, Naoyuki Shida, Kouichi Hirai, Toshiyuki Shimizu, Shinya Hiramoto, Yoshiro Ikeda, Takahide Yoshikawa, Kenji Uchida, et al. The tofu interconnect d. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 646–654. IEEE, 2018.

[77] Norman P Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, et al. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. *arXiv preprint arXiv:2304.01433*, 2023.

[78] Keun Sup Shim, Brian Greskamp, Brian Towles, Bruce Edwards, JP Grossman, and David E Shaw. The specialized high-performance network on anton 3. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 1211–1223. IEEE, 2022.

[79] Yousef Saad and Martin H Schultz. Topological properties of hypercubes. *IEEE Transactions on computers*, 37(7):867–872, 1988.

[80] Leslie G. Valiant. A scheme for fast parallel communication. *SIAM journal on computing*, 11(2):350–361, 1982.

[81] Ching-Tien Ho and S Lennart Johnsson. Distributed routing algorithms for broadcasting and personalized communication in hypercubes. In *ICPP*, pages 640–648, 1986.

[82] Thomas H. Dunigan. Performance of the intel ipsc/860 and ncube 6400 hypercubes. *Parallel Computing*, 17(10-11):1285–1302, 1991.

[83] Fabrizio Petrini and Marco Vanneschi. k-ary n-trees: High performance networks for massively parallel architectures. In *Proceedings 11th international parallel processing symposium*, pages 87–93. IEEE, 1997.

[84] Jack J. Dongarra. Report on the sunway taihulight system, 2016.

[85] Hao Gao and Nikolai Sakharnykh. Scaling joins to a thousand gpus. In *ADMS@ VLDB*, pages 55–64, 2021.

[86] Xinhai Chen, Jie Liu, Shengguo Li, Peizhen Xie, Lihua Chi, and Qinglin Wang. Tamm: a new topology-aware mapping method for parallel applications on the tianhe-2a supercomputer. In *Algorithms and Architectures for Parallel Processing: 18th International Conference, ICA3PP 2018, Guangzhou, China, November 15-17, 2018, Proceedings, Part I 18*, pages 242–256. Springer, 2018.

[87] John Kim, Wiliam J. Dally, Steve Scott, and Dennis Abts. Technology-Driven, Highly-Scalable Dragonfly Topology. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 77–88, 2008.

[88] Alexander Shpiner, Zachy Haramaty, Saar Eliad, Vladimir Zdornov, Barak Gafni, and Eitan Zahavi. Dragonfly+: Low cost topology for scaling datacenters. In *2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*, pages 1–8. IEEE, 2017.

[89] Daniele Cesarini. The leonardo supercomputer at the bologna big data technopole and the cineca's evolution roadmap. https://www.european-processor-initiative.eu/wp-content/uploads/2022/09/EPI-@-REHE2022.pdf, 2022.

[90] Scott Atchley. Frontier' s architecture. https://olcf.ornl.gov/wp-content/uploads/Frontiers-Architecture-Frontier-Training-Series-final.pdf, 2022.

[91] Jan Vicherek. Introduction of lumi supercomputer. https://events.it4i.cz/event/160/attachments/457/1717/lumi-intro.pdf, 2022.

[92] NERSC Documentation. Perlmutter architecture. https://docs.nersc.gov/systems/perlmutter/architecture/.

[93] Ankit Singla, Chi-Yao Hong, Lucian Popa, and Philip Brighten Godfrey. Jellyfish: Networking data centers, randomly. In *NSDI*, volume 12, pages 17–17, 2012.

[94] Markus Junginger and Yugyung Lee. The multi-ring topology-high-performance group communication in peer-to-peer networks. In *Proceedings. Second International Conference on Peer-to-Peer Computing,*, pages 49–56. IEEE, 2002.

[95] Jung-Heum Park and Kyung-Yong Chwa. Recursive circulant: A new topology for multicomputer networks. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, pages 73–80. IEEE, 1994.

[96] Shyue-Ming Tang, Yue-Li Wang, and Chien-Yi Li. Generalized recursive circulant graphs. *IEEE Transactions on Parallel and Distributed Systems*, 23(1):87–93, 2011.

[97] R. Kesavan and D. K. Panda. Efficient multicast on irregular switch-based cut-through networks with up-down routing. *IEEE Transactions on Parallel and Distributed Systems*, 12(8):808–828, Aug 2001.

[98] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks: an engineering approach.* Morgan Kaufmann, 2002.

[99] Jehoshua Bruck, Ching-Tien Ho, Shlomo Kipnis, Eli Upfal, and Derrick Weathersby. Efficient algorithms for all-to-all communications in multiport message-passing systems. *IEEE Transactions on parallel and distributed systems*, 8(11):1143–1156, 1997.

[100] Elie Krevat, Jose Castaños, and Jose Moreira. Job scheduling for the bluegene/l system. In *Euro-Par: International Euro-Par Conference on Parallel Processing*, pages 207–211, 11 2002.

[101] T. Agarwal, A. Sharma, A. Laxmikant, and L. V. Kale. Topology-aware task mapping for reducing communication contention on large parallel machines. In *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, pages 10 pp.–, April 2006.

[102] V. J. Leung, E. M. Arkin, M. Bender, D. Bunde, J. Johnston, Alok Lal, J. S. B. Mitchell, C. Phillips, and S. S. Seiden. Processor allocation on cplant: achieving general processor locality using one-dimensional allocation strategies. In *Proceedings. IEEE International Conference on Cluster Computing*, pages 296–304, 2002.

[103] Carl Albing. Characterizing node orderings for improved performance. In *Proceedings of the 6th International Workshop on Performance Modeling, Benchmarking, and Simulation of High Performance Computing Systems*, pages 1–11, 11 2015.

[104] Carl Albing, Norm Troullier, Stephen Whalen, Ryan Olson, Joe Glenski, Howard Pritchard, and Hugo Mills. Scalable node allocation for improved performance in regular and anisotropic 3d torus supercomputers. In Yiannis Cotronis, Anthony Danalis, Dimitrios S. Nikolopoulos, and Jack Dongarra, editors, *Recent Advances in the Message Passing Interface*, pages 61–70, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[105] H. Yu, I. Chung, and J. Moreira. Topology mapping for blue gene/l supercomputer. In *SC '06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, pages 52–52, 2006.

[106] Abhinav Bhatele and Laxmikant V. Kalé. Application-specific topology-aware mapping for three dimensional topologies. *2008 IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8, 2008.

[107] O. Tuncer, V. J. Leung, and A. K. Coskun. Pacmap: Topology mapping of unstructured communication patterns onto non-contiguous allocations. In *Proceedings of the 29th ACM on International Conference on Supercomputing*, pages 37–46, 2015.

[108] G. Michelogiannakis, K. Z. Ibrahim, J. Shalf, J. J. Wilke, S. Knight, and J. P. Kenny. Aphid: Hierarchical task placement to enable a tapered fat tree topology for lower power and cost in hpc networks. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 228–237, 2017.

[109] Nikhil Jain, Matthew Leininger, Abhinav Bhatele, Louis Howell, David Böhme, Ian Karlin, Edgar Leon, Misbah Mubarak, Noah Wolfe, and Todd Gamblin. Predicting the performance impact of different fat-tree configurations. In *SC: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–13, 11 2017.

[110] E. Zahavi. Fat-trees routing and node ordering providing contention free traffic for mpi global collectives. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pages 761–770, 2011.

[111] N. Jain, A. Bhatele, X. Ni, T. Gamblin, and L. V. Kale. Partitioning low-diameter networks to eliminate inter-job interference. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 439–448, 2017.

[112] S. D. Pollard, N. Jain, S. Herbein, and A. Bhatele. Evaluation of an interference-free node allocation policy on fat-tree clusters. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 333–345, 2018.

[113] Bogdan Prisacari, German Rodriguez, Philip Heidelberger, Dong Chen, Cyriel Minkenberg, and Torsten Hoefler. Efficient task placement and routing of nearest neighbor exchanges in dragonfly networks. *HPDC 2014 - Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing*, 06 2014.

[114] R. Budiardja, L. Crosby, and Haihang You. Effect of rank placement on cray xc 30 communication cost. In *The Cray User Group Meeting*, 2013.

[115] N. Jain, A. Bhatele, S. White, T. Gamblin, and L. V. Kale. Evaluating hpc networks via simulation of parallel workloads. In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 154–165, 2016.

[116] Krishna Kandalla, Hari Subramoni, Abhinav Vishnu, and Dhabaleswar K Panda. Designing topology-aware collective communication algorithms for large scale

infiniband clusters: Case studies with scatter and gather. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8. IEEE, 2010.

[117] H. Subramoni, K. Kandalla, J. Vienne, S. Sur, B. Barth, K. Tomko, R. Mclay, K. Schulz, and D. K. Panda. Design and evaluation of network topology-/speed-aware broadcast algorithms for infiniband clusters. In *2011 IEEE International Conference on Cluster Computing*, pages 317–325, Sep. 2011.

[118] Yijia Zhang, Ozan Tuncer, Fulya Kaplan, Katzalin Olcoz, Vitus J Leung, and Ayse K Coskun. Level-spread: A new job allocation policy for dragonfly networks. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1123–1132. IEEE, 2018.

[119] Abhinav Bhatelé and Laxmikant V Kalé. Benefits of topology aware mapping for mesh interconnects. *Parallel Processing Letters*, 18(04):549–566, 2008.

[120] Jingjin Wu, Xuanxing Xiong, and Zhiling Lan. Hierarchical task mapping for parallel applications on supercomputers. *The Journal of supercomputing*, 71:1776–1802, 2015.

[121] G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958.

[122] George Michelogiannakis, Khaled Z Ibrahim, John Shalf, Jeremiah J Wilke, Samuel Knight, and Joseph P Kenny. Aphid: Hierarchical task placement to enable a tapered fat tree topology for lower power and cost in hpc networks. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 228–237. IEEE, 2017.

[123] Yao Hu and Michihiro Koibuchi. The impact of job mapping on random network topology. In *International Symposium on Computing and Networking (CANDAR) Workshop*, pages 79–85, 11 2018.

[124] Yao Hu and Michihiro Koibuchi. Diameter/aspl-based mapping of applications with uncertain communication over random interconnection networks. In *2019*

IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), pages 249–258. IEEE, 2019.

[125] P. Coteus and et. al. Packaging the Blue Gene/L supercomputer. *IBM Journal of Research and Development*, 49(2/3):213–248, Mar/May 2005.

[126] Seong-Moo Yoo, Hee Yong Youn, and Behrooz Shirazi. An efficient task allocation scheme for 2d mesh architectures. *IEEE Transactions on Parallel and Distributed systems*, 8(9):934–942, 1997.

[127] Jose A Pascual, Jose Miguel-Alonso, and Jose A Lozano. Locality-aware policies to improve job scheduling on 3d tori. *The Journal of Supercomputing*, 71:966–994, 2015.

[128] Hyunseung Choo, Seong-Moo Yoo, and Hee Yong Youn. Processor scheduling and allocation for 3d torus multicomputer systems. *IEEE Transactions on Parallel and Distributed Systems*, 11(5):475–484, 2000.

[129] SimGrid: Versatile Simulation of Distributed Systems. http://simgrid.gforge.inria.fr/.

[130] Nas Parallel Benchmarks. https://www.nas.nasa.gov/publications/npb.html.

[131] Ahmad Faraj and Xin Yuan. Communication characteristics in the nas parallel benchmarks. In *IASTED PDCS*, pages 724–729, 2002.

[132] Frank Boesch and Ralph Tindell. Circulants and their connectivities. *Journal of Graph Theory*, 8(4):487–499, 1984.

[133] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter. Versatile, Scalable, and Accurate Simulation of Distributed Appl ications and Platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917, 2014.

[134] Fabien Chaix, Ikki Fujiwara, and Michihiro Koibuchi. Suitability of the random topology for hpc applications. In *Parallel, Distributed, and Network-Based Processing (PDP), 2016 24th Euromicro International Conference on*, pages 301–304. IEEE, 2016.

[135] HP. Optimizing facility operation in high density data center environments , technoloogy brief, 2007.

[136] I. Fujiwara, M. Koibuchi, H. Matsutani, and H. Casanova. Swap-and-randomize: A method for building low-latency hpc interconnects. *IEEE Transactions on Parallel and Distributed Systems*, 26(7):2051–2060, July 2015.