Short Hops, Big Gains: Beyond Asymptotic Costs in Collective Communication on Dragonfly+

Chiara Di Stefano[1][0009−0004−2475−9932] and
Daniele De Sensi[1][0000−0002−7244−639X]

Sapienza University of Rome
distefano.2061657@studenti.uniroma1.it, desensi@di.uniroma1.it

Abstract. Collective operations are a critical bottleneck in large-scale HPC systems. Traditional cost models evaluate collective algorithms based on asymptotic complexity, yet algorithms with identical theoretical cost can exhibit vastly different performance in practice. In this work, we show that the traffic patterns induced by a collective algorithm, and their interaction with the underlying network topology, are the decisive factor for performance. Through network simulation on a Dragonfly+ topology, we demonstrate that algorithms characterized by the same cost model differ by more than 6× in completion time, a gap explained entirely by how they distribute traffic across intra- and inter-group links. Our results underscore the need to move beyond asymptotic cost models when designing collective algorithms for modern HPC networks.

Keywords: collective algorithms · dragonfly+

1 Introduction

High-Performance Computing (HPC) systems have exceeded exascale performance. However, inter-node communication remains a critical bottleneck that is expected to worsen in the exascale era [4]. Collective operations are among the most communication-sensitive components of distributed applications, often dominating overall runtime. Consequently, designing and optimizing efficient collective algorithms has become a critical priority for large-scale HPC clusters.

Collective algorithms have been studied theoretically using cost models based on the number of communication rounds and the volume of data exchanged. Although these models capture algorithmic complexity, they abstract away from the physical network entirely [6]. However, in practice the relationship between theoretical cost and observed performance can diverge significantly, as demonstrated by the results presented in this work.

While full-bandwidth topologies such as non-blocking Fat-Trees provide high performance, their deployment is often constrained by substantial cost and power consumption. To mitigate these overheads, modern large-scale systems frequently adopt oversubscribed networks, typically organized into fully connected groups, such as the groups in Dragonfly and Dragonfly+ topologies [3, 5], or into subtrees

in oversubscribed Fat-Trees. In these blocking networks, inter-group links represent a critical bottleneck: they are shared across multiple concurrent flows, often from different applications, and might cause congestion when traffic demand exceeds their capacity. This contention increased latency and reduced throughput, ultimately degrading collective performance.

Minimizing traffic across these oversubscribed links is therefore essential for efficient collective communication on oversubscribed topologies. Recent work has proposed algorithms such as Bine and Sparbit [2], which prioritize short-range exchanges and limit traffic across oversubscribed links. By doing so, these algorithms aim to reduce congestion and improve bandwidth utilization on the interconnects where classical approaches struggle most.

In this work, we present a simulation-based analysis of five Allgather algorithms (Bine, Sparbit, Bruck, Recursive Doubling, and Ring) evaluated on Dragonfly+ topologies. Our evaluation goes beyond completion time and examines how each algorithm distributes traffic across the network, distinguishing between intra- and inter-group links. Our results show that algorithms sharing the same asymptotic cost model can exhibit substantially different performance depending on how their communication patterns interact with the underlying network topology, a distinction that theoretical models alone cannot predict.

2 Methodology

The goal of our evaluation is to investigate why collective algorithms with similar theoretical cost models exhibit significantly different performance in practice, focusing on how their traffic patterns interact with the underlying network topology. We used ht-sim [7], an open-source C++ network simulator. It supports various topologies, including Fat-Tree and Dragonfly, as well as several high-performance network protocols, including RoCE (RDMA over Converged Ethernet), which we used in this work.

To assess how different collective algorithms are affected by oversubscribed network topologies, we extended the simulator with an implementation of the Dragonfly+ topology [5]. Moreover, we added scripts to generate connection matrices for several collective algorithms, which we use as simulation input. To monitor port usage, we extended it so that, for every forwarded packet, each switch logs: the source and destination switch identifiers, the transmission timestamp, the packet size in bytes, and the transmission latency in picoseconds. The updated code can be found at `https://github.com/HLC-Lab/uet-htsim`.

3 Experimental Result

3.1 Environment

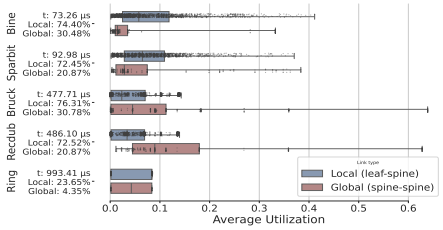
All experiments were conducted on a Dragonfly+ topology [5]. We consider three configurations of different size, all sharing the same internal structure: within each group, leaf and spine switches are fully connected to each other, each leaf

switch connects to a number of hosts equal to half of the switch radix, and each spine switch connects to remote spines in other groups. The large topology has 1332 nodes (radix 12) across 37 groups, with 6 leaf and 6 spine switches per group; the medium one has 1100 nodes (radix 20) across 11 groups, each with 10 leaf and 10 spine switches; the small topology has 1183 nodes (radix 26) across 7 groups, each with 13 leaf and 13 spine switches. The three topologies differ in the number of links connecting each pair of groups, one in the large topology, ten in the medium, and twenty-six in the small, which lets us evaluate the effect of different oversubscription ratios. In all three, groups are mutually reachable but inter-group links are oversubscribed, so full global bandwidth is never achievable. Although each topology supports more nodes, all simulations use 1024 to satisfy the power-of-two requirement of certain collective algorithms. The routing algorithm always selects the shortest path in terms of switch hops.

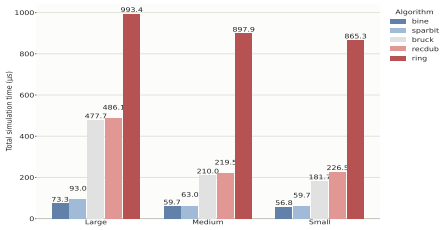
The network operates at 400 Gbps with a 4064-byte MTU (4000-byte payload and 64-byte header) and a link latency of 20 ns (assuming 5 ns/m over optical fiber, corresponding to ∼4 m links). The switching infrastructure is modeled after Broadcom Tomahawk 6 switches [1], with an internal processing latency of 250 ns and a per-port queue capacity of approximately 364 packets. Since ht-sim is deterministic for a fixed random seed, each configuration was simulated once.

3.2 Result

The following analysis focuses on the Allgather collective operation with an output vector size of 1 MiB. Bine [2], Sparbit, Bruck, and Recursive Doubling all complete in a logarithmic number of steps and transfer the same number of bytes, yet their completion times differ substantially, as reported in Fig. 1b. Since neither step count nor data volume accounts for this gap, the explanation must lie in how each algorithm distributes its traffic across the network.



(a) Link utilization on large Dragonfly+.



(b) Configuration comparison.

To investigate this, Fig. 1a breaks down port utilization into intra-group and inter-group links for the large configuration; the x-axis shows utilization during the simulation and the y-axis lists the algorithms together with simulation time and the ratio of used ports over the total. Bine and Sparbit sustain high intra-group utilization while keeping inter-group utilization low. Bruck and Recursive Doubling, in contrast, drive sustained load through the inter-group links, while Ring activates the fewest ports overall, concentrating traffic almost entirely on local links.

This utilization pattern follows directly from which nodes each algorithm pairs at every step. Bine and Sparbit prioritize short-range communication, concentrating data exchange within groups and saturating local bandwidth rather than spilling traffic outward. Bruck and Recursive Doubling frequently pair distant nodes, pushing traffic onto the oversubscribed inter-group links and causing the queuing congestion that dominates their completion time. Ring follows a strictly nearest-neighbor pattern, touching inter-group links only at group boundaries; this avoids congestion entirely but, given its linear number of steps, makes it the slowest at the message size evaluated here, a disadvantage that would shrink as the vector size increases.

Fig. 1b shows the same analysis on the medium and small topologies, where inter-group links are more numerous than in the large topology. The same pattern among algorithms is still visible, but the gap between them shrinks as the number of inter-group links increases. With less oversubscription, congestion on global links drops, so algorithms like Bruck and Recursive Doubling are penalized less, and their completion time gets closer to that of Bine and Sparbit.

We also tested these algorithms with adaptive routing, but observed no significant differences for this collective compared to the shortest-path results above.

4 Conclusion and Future work

Our analysis underscores the importance of evaluating collective algorithms not only on asymptotic cost models, but also with respect to the traffic patterns they induce on the target topology. In the future, we will extend our analysis to other topologies and routing algorithms. *Acknowledgment:* This work is supported by the European Union's Horizon Europe under grant 101175702 (NET4EXA).

References

[1] Broadcom Inc. *Broadcom Ships Tomahawk Ultra: Reimagining the Ethernet Switch for HPC and AI Scale-up*. 2025.

[2] Daniele De Sensi et al. "Bine Trees: Enhancing Collective Operations by Optimizing Communication Locality". In: *Proc. SC'25*. 2025, pp. 1901–1916.

[3] John Kim et al. "Technology-Driven, Highly-Scalable Dragonfly Topology". In: *Intl. Symposium on Computer Architecture (ISCA)*. 2008.

[4] Xiang-Ke Liao et al. "Moving From Exascale to Zettascale Computing: Challenges and Techniques". In: *Frontiers of Information Technology & Electronic Engineering* 19.10 (2018), pp. 1236–1244.

[5] Alexander Shpiner et al. "Dragonfly+: Low Cost Topology for Scaling Datacenters". In: *HiPINEB*. 2017.

[6] Rajeev Thakur and William Gropp. *Improving the Performance of MPI Collective Communication on Switched Networks*. 2002.

[7] Ultra Ethernet Consortium. *HTSim: High-Throughput Network Simulator*. 2024. URL: `https://github.com/ultraethernet/uet-htsim`.