Accelerating Collective Communication With Error-Bounded Lossy Compression

Jiajun Huang

ANL & UC Riverside

Motivation

- MPI collective -> high-performance -> **significant impact on various research fields.**
- Exascale computing -> large-message MPI collectives -> **Scalability challenges.**
  - VGG19 with 143 million parameters -> communication overhead of 83% [1].
  - ResNet-50 with 25 million parameters -> communication overhead of 72% [1].
- Inter-node communications -> **limited network bandwidth -> major bottleneck.**
- How can we solve this bottleneck?

Motivation

- **Designing large message algorithms:** Decrease the overall communication volume.
  - **Allreduce:** Ring: $\frac{2*(N-1)}{N} * D$ **vs** Recursive-doubling: $\lceil \log N \rceil * D$.
- **Lossy compression**: Significantly reduce the message size.
  - To address this issue, prior research simply applies the off-the-shelf fix-rate lossy compressors in the MPI collectives, leading to **suboptimal performance**, **limited generality**, and **unbounded errors** [2].

Design of C-Coll

- **C-Coll (Compression-accelerated Collectives)**: **(IPDPS 24)**
  - Overlap the compression with communication using our developed pipelined SZx in our collective computation framework.
  - Reduce the compression overhead and mitigate error propagation by choosing the appropriate timing of compression.
- **Mathematical proof**: To prove the error-bounded nature -> We perform an in-depth mathematical analysis to derive the limited impact of error-bounded lossy compression on error propagation.

Theoretical Analysis of Error Propagation for C-Coll

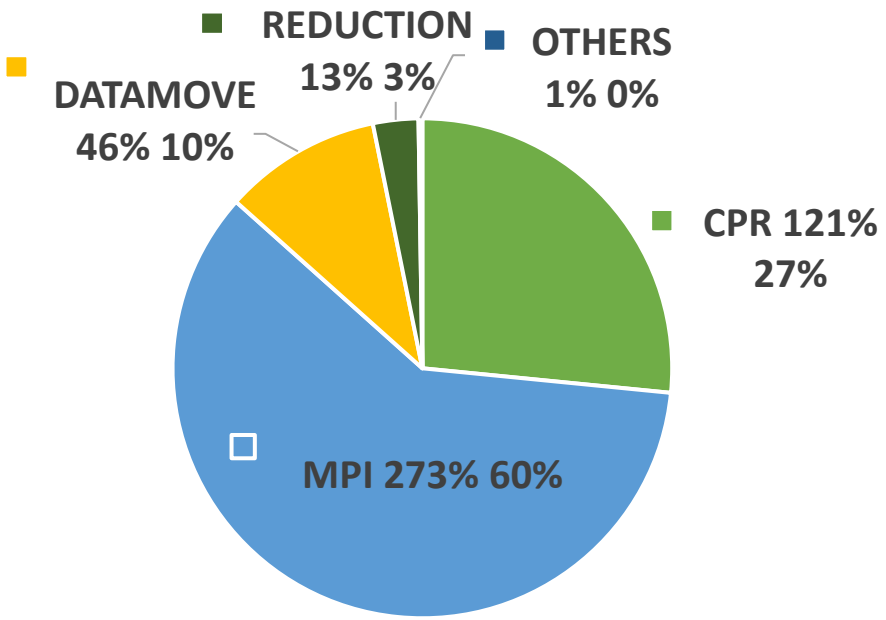
- **Collective data movement framework**:
  - The final error for each data point is within $\hat{e}$, where $\hat{e}$ is the compression error bound.
- **Collective computation framework**:
  - The final aggregated error of the most widely used sum operation falls within the interval $[-\frac{2}{3}\sqrt{n}\hat{e}, \frac{2}{3}\sqrt{n}\hat{e}]$ with the probability of **95.44%**.
  - For example, if there are 100 nodes, and the error bound is 1E-3, the aggregated error is bounded in the range of $[-6.7\mathrm{E}{-3}, 6.7\mathrm{E}{-3}]$ with a probability of **95.44%**.

Performance of C-Coll

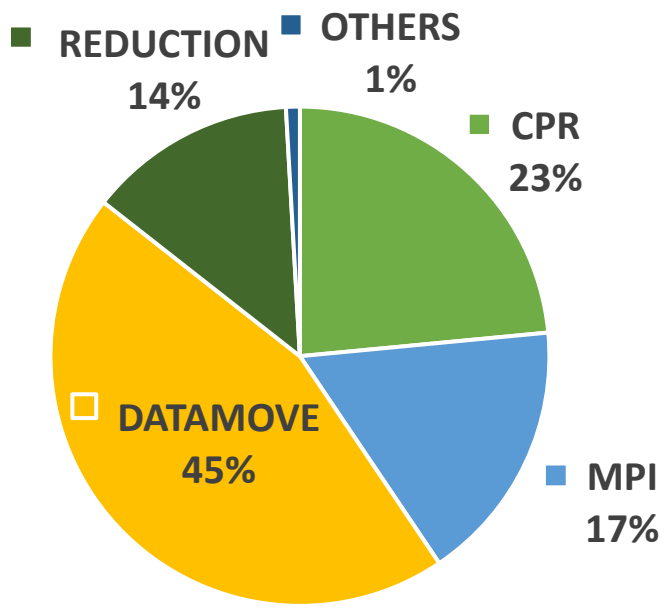
- **Our C-Coll:** a novel design for lossy-compression-integrated MPI collectives that significantly **improves** performance with bounded errors.
  - C-Allreduce is up to 2.1X faster than MPI_Allreduce, while other CPRP2P baselines demonstrate performance degradation.
  - C-Scatter is up to 1.8X faster than MPI_Scatter.
  - C-Bcast is up to 2.7X faster than MPI_Bcast.

Limitation of C-Coll

- C-Coll is optimized for host-centric collective communications. Thus, it faces serious issues in GPU-centric communications [2].
  - Expensive host-device data movements.
  - Underutilized GPU devices.



(a) C-Coll



(b) CPRP2P

***Figure 1*: Performance breakdown of Allreduce using CPRP2P and C-Coll on 64 A100 GPUs: CPRP2P's first percentage is scaled to C-Coll's runtime, and the second is scaled to its own.**

Design of gZCCL

- **gZCCL (GPU-aware Compression-Accelerated Collective Communication Library)**: **(ICS 24)**
  - Improve the scalability and GPU utilization in the collective computation framework.
  - Overlap compression with our multi-stream cuSZp in the collective data movement framework.

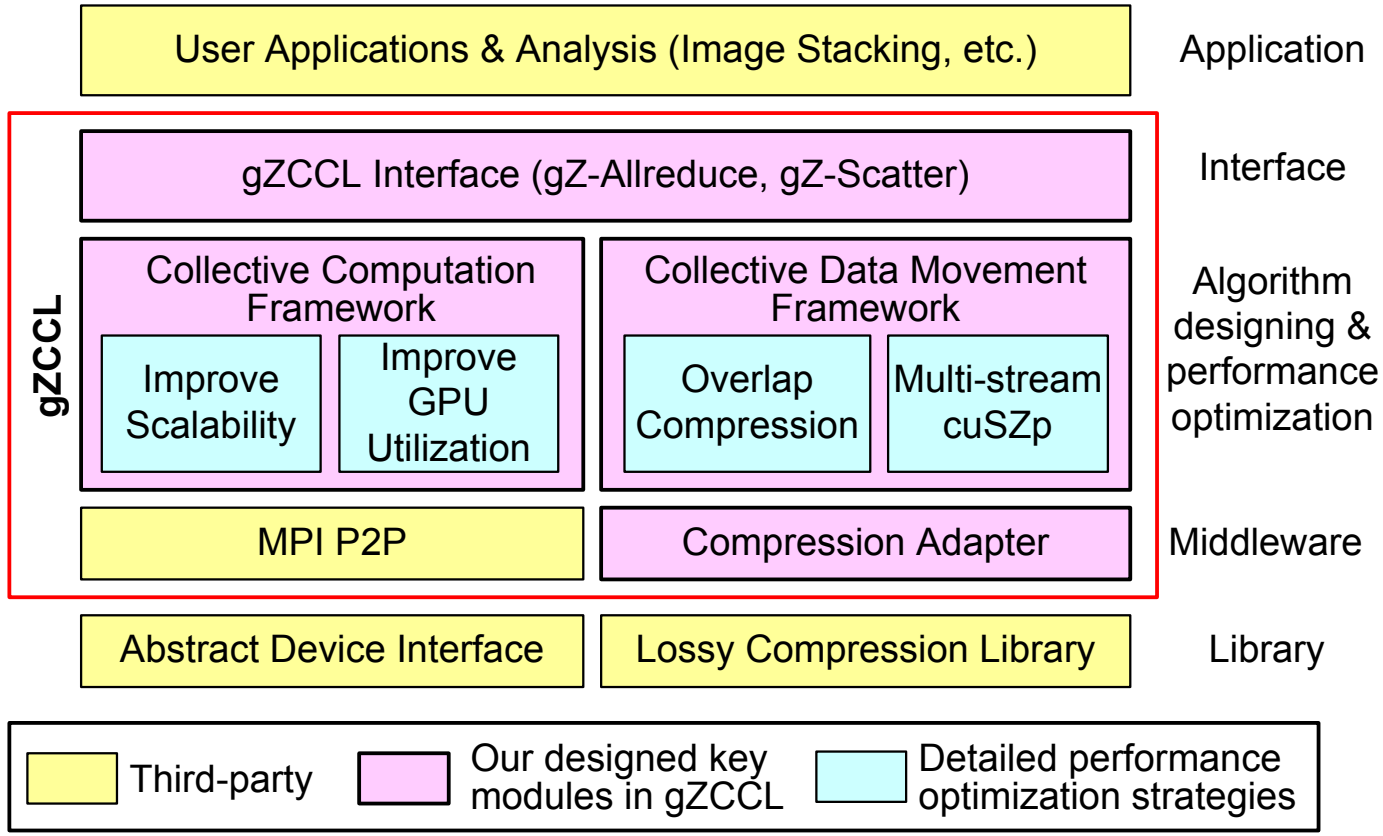


***Figure 1*: Design architecture (purple box: newly contributed modules)**

Evaluating the Collective Computation Framework of gZCCL

Figure 2 demonstrates that our recursive doubling-based gZ-Allreduce (ReDoub) consistently performs the best, achieving up to **20.2X** and **4.5X** speedups compared to **Cray MPI** and **NCCL** respectively, across varying GPU counts.

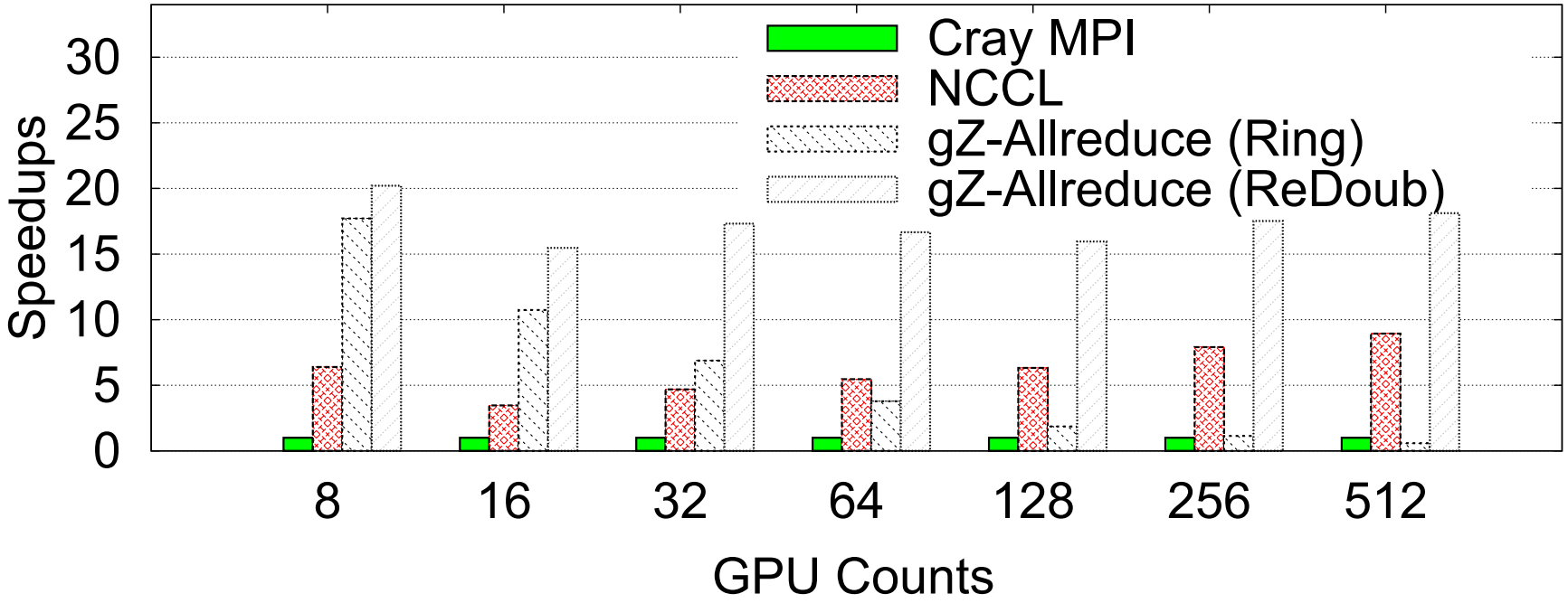
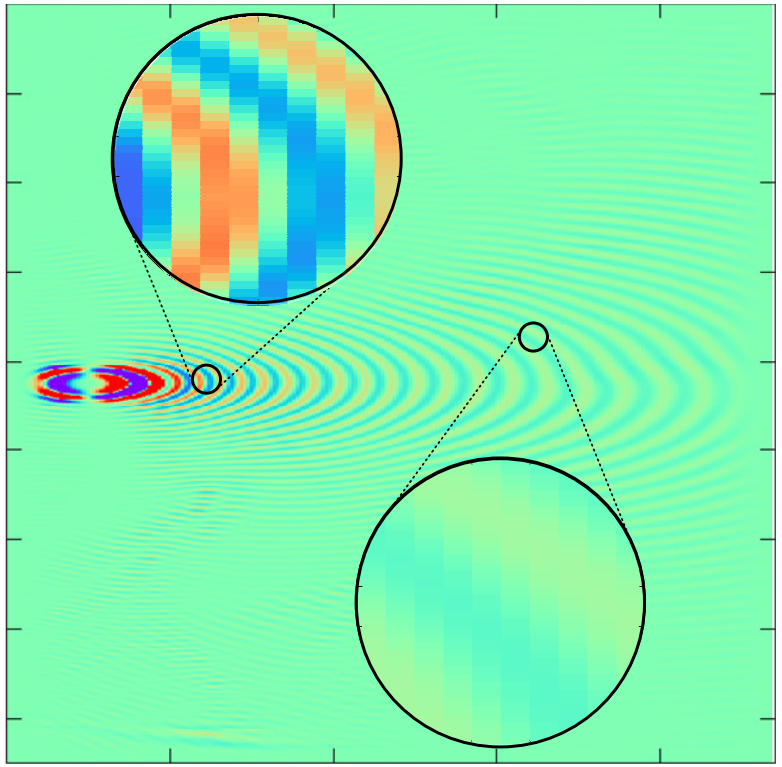
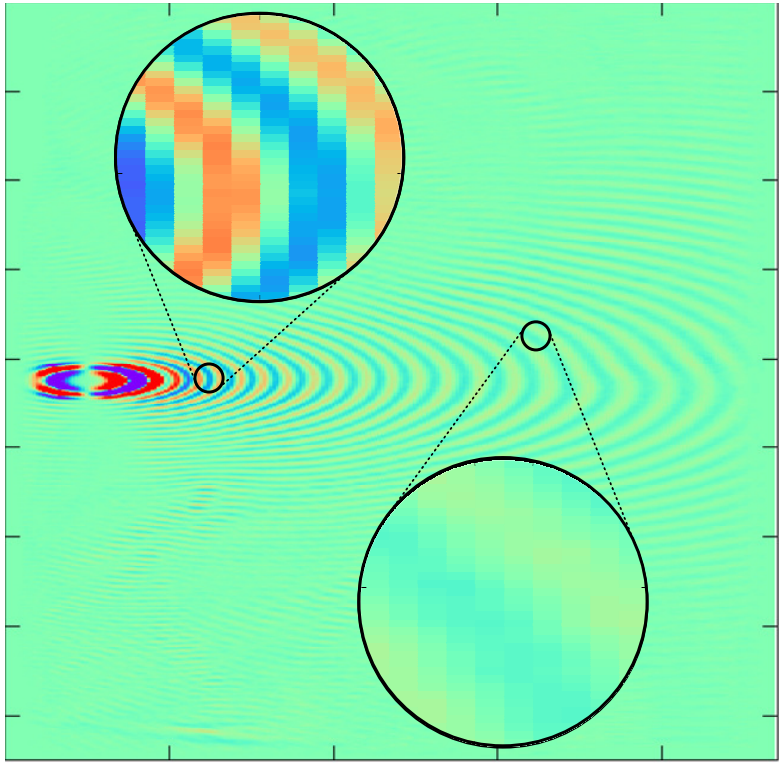


***Figure 2*: Scalability evaluation of our gZ-Allreduce with Cray MPI and NCCL in different GPU counts.**

Image Stacking Accuracy Analysis



(a) Cray MPI/NCCL (lossless)



(b) gZCCL (2E-4)

***Figure 4*: Visualization of final stacking image.**

gZCCL (Ring) (1E-4) reaches a great PSNR of **56.83** and an NRMSE of **1E-3**.
gZCCL (ReDoub) (1E-4) demonstrates better data quality, achieving a PSNR of **57.80** and an NRMSE of **1E-3**.

Summary

- **Performance**: With our C-Coll and gZCCL, compression-accelerated collectives have significantly higher performance than the SOTA communication libraries on both CPU and GPU.

- **Accuracy**: With the accuracy-aware design, C-Coll and gZCCL can demonstrate well-controlled accuracy through both theoretical and experimental analysis.

- **Future Work**: We plan to further optimize compression-accelerated collectives for FPGAs and AI accelerators.

References

1. A. M. Abdelmoniem, A. Elzanaty, M.-S. Alouini, and M. Canini, “An efficient statistical-based gradient compression technique for distributed training systems,” 2021.
2. Jiajun Huang and Sheng Di and Xiaodong Yu and Yujia Zhai and Zhaorui Zhang and Jinyang Liu and Xiaoyi Lu and Ken Raffenetti and Hui Zhou and Kai Zhao and Zizhong Chen and Franck Cappello and Yanfei Guo and Rajeev Thakur. 2023. An Optimized Error-controlled MPI Collective Framework Integrated with Lossy Compression. arXiv:2304.03890 [cs.DC]
3. Yafan Huang, Sheng Di, Xiaodong Yu, Guanpeng Li, and Franck Cappello. 2023. CuSZp: An Ultra-fast GPU Error-bounded Lossy Compression Framework with Optimized End-to-End Performance. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '23).

Questions?

Thanks for your attention!

C-Coll Overall system design architecture

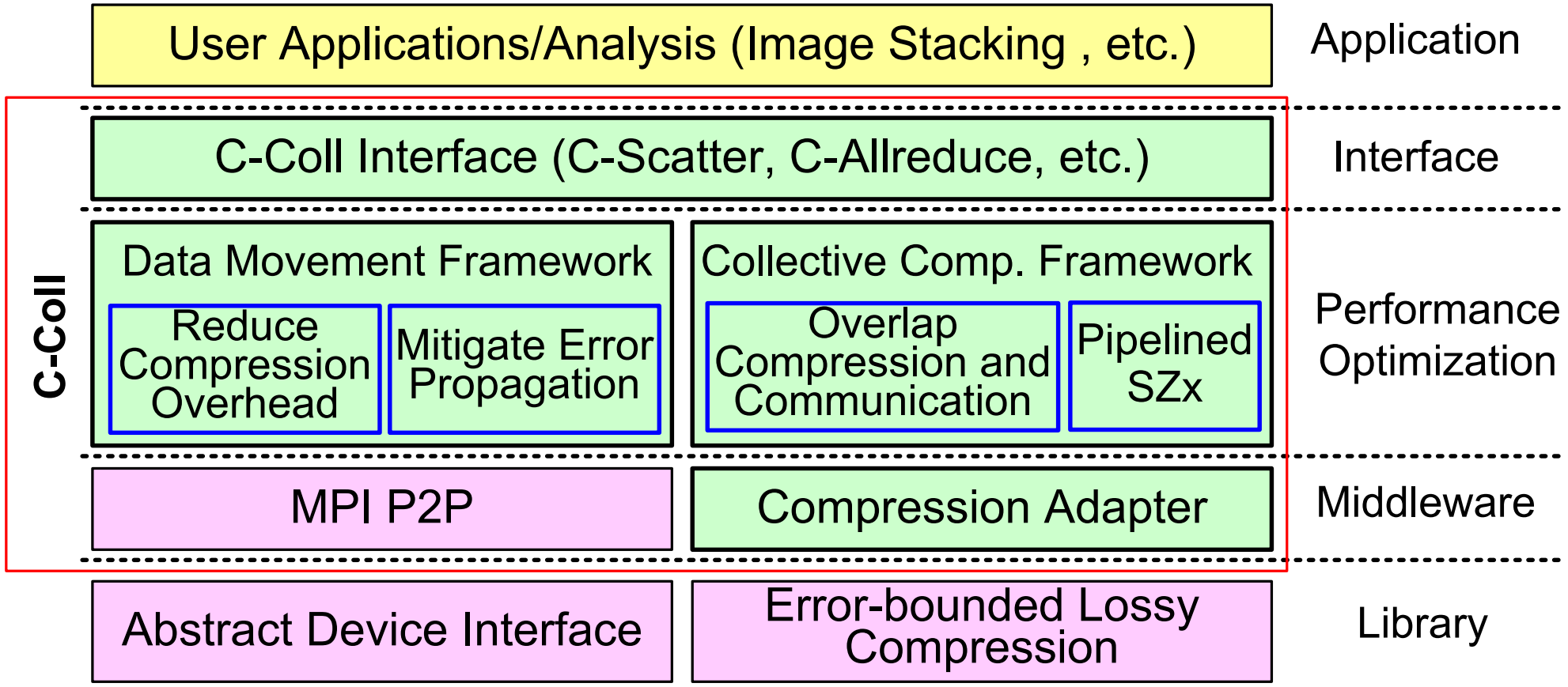


***Figure 1*: Design architecture (yellow box: applications; green box: new contributed modules; purple box: third-party)**

Image Stacking Performance Evaluation

***Table 1*: Image stacking performance analysis (The speedups are based on Cray MPI. The last four columns are performance breakdowns of our gZCCL).**

	Speedups	Cmpr.	Comm.	Redu.	Others
gZCCL (Ring)	3.99	84.08%	14.08%	1.26%	0.58%
gZCCL (ReDoub)	9.26	42.61%	46.28%	11.04%	0.06%
NCCL	5.47	No breakdown because of complexity			

Evaluating Our Collective Data Movement Framework (gZCCL)

Figure 3 shows that our gZ-Scatter outperforms Cray MPI in all cases. As the GPU count increases, the speedup of gZ-Scatter first increases, peaking at **28.7X**, and then gradually decreases to **4.75X** when the GPU count reaches 512.

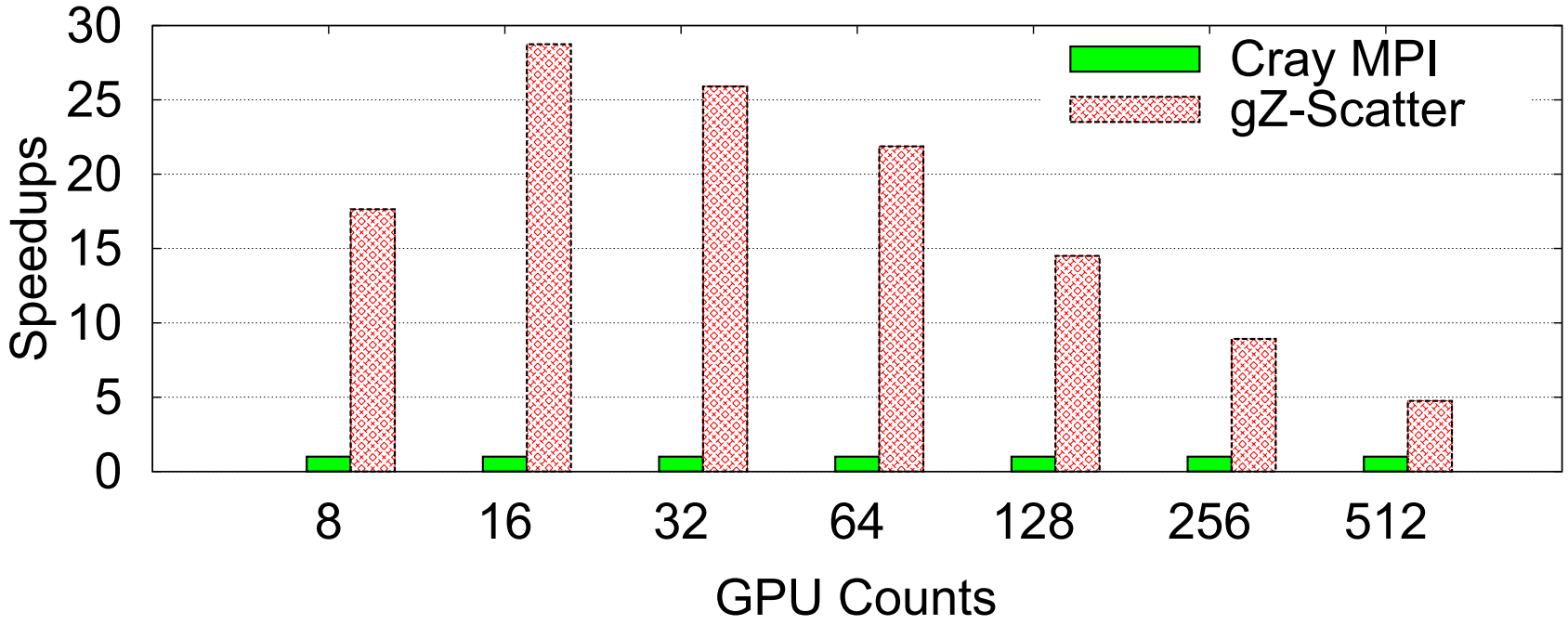


***Figure 3*: Scalability evaluation of our gZ-Scatter with Cray MPI in different GPU counts.**

Evaluating Our Collective Computation Framework

Figure 3 shows that our recursive doubling-based gZ-Allreduce (ReDoub) consistently outperforms across all data sizes, achieving up to a speedup of **18.7X** compared to **Cray MPI** and a **3.4X** performance improvement over **NCCL**.

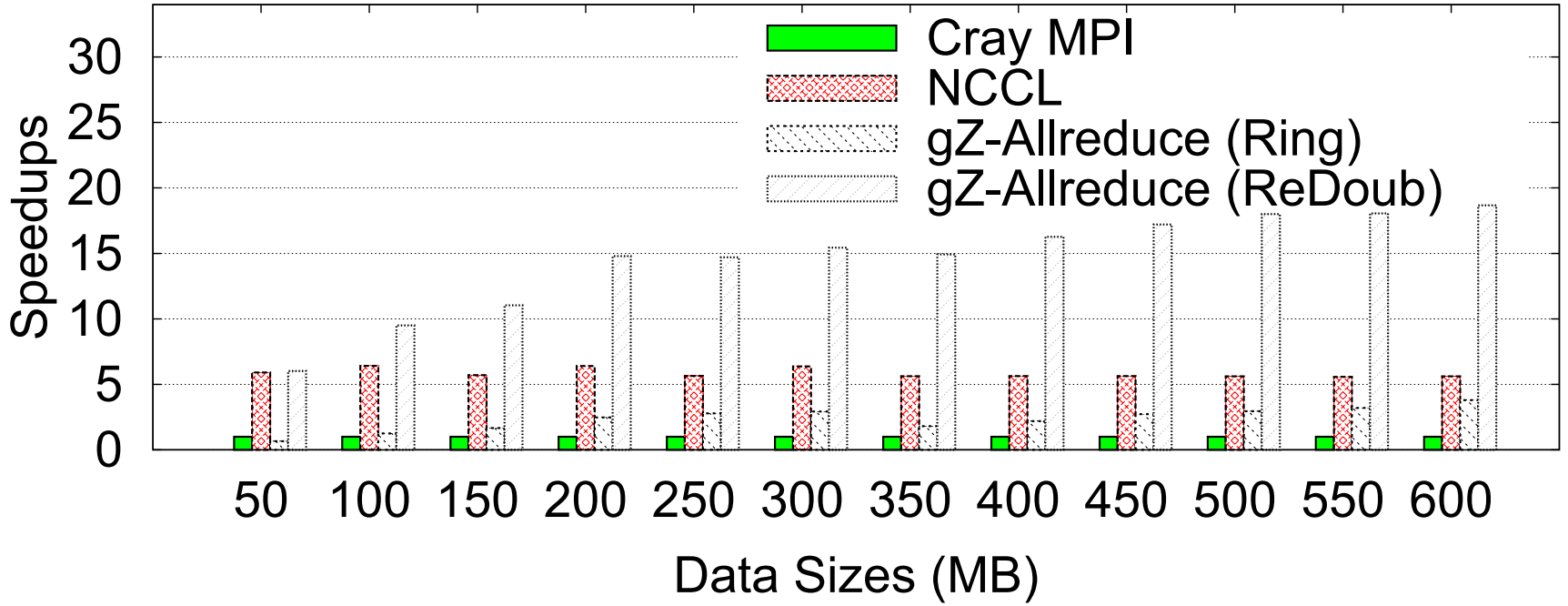


***Figure 3*: Performance evaluation of our gZ-Allreduce with Cray MPI and NCCL in different data sizes.**

Evaluating Our Collective Data Movement Framework

Figure 5 indicates that our gZ-Scatter is able to consistently outperform Cray MPI across all data sizes. The speedup of gZ-Scatter enhances as the data size increases, achieving its maximum **20.2X** at 600 MB.

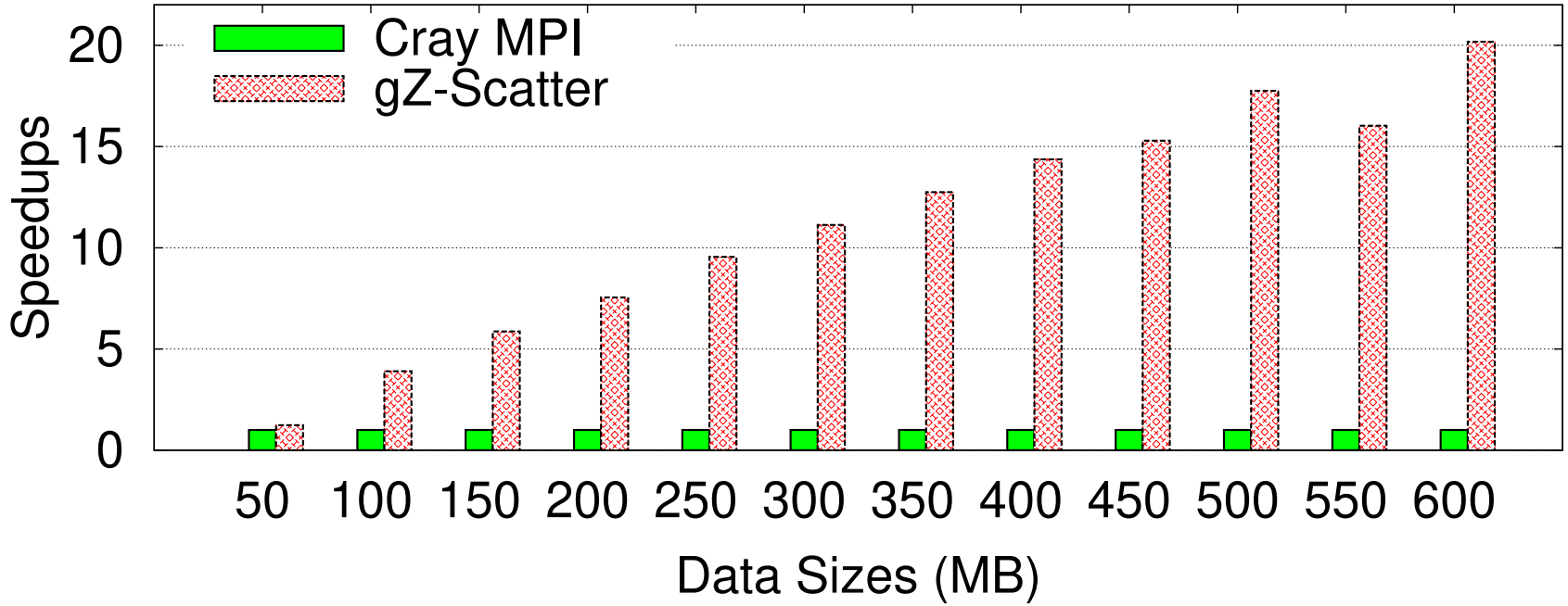


***Figure 5*: Performance evaluation of our gZ-Scatter with Cray MPI in different data sizes.**

About me

- **Two teams in Argonne**:
  - Yanfei Guo and Rajeev Thakur in the **MPICH** project, which is one of the most widely-used MPI libraries.
  - Sheng Di and Franck Cappello in the **SZ** project, which is the leading lossy compressor framework.