小作业二：MPI Allreduce

简述所实现的 Ring Allreduce 算法

主要思想是将所有进程中的数据分成若干个相等的块，然后通过两阶段的通信操作，完成局部规约并最终使所有进程都获得规约后的全局结果。

假设有 `comm_sz` 个进程，将总数据 `n` 分成 `comm_sz` 块，每个块大小为 `chunk = n / comm_sz`。每个进程在本地先对自己对应的那一块（偏移量为 `my_rank * chunk`）进行操作。

在第一阶段，每个进程在第一次迭代时直接把自己对应的数据块从 `sendbuf` 复制到 `recvbuf`。在随后的迭代中，每个进程从左侧进程接收一个数据块，并将接收到的数据与本地对应的 `sendbuf` 数据块进行累加操作（规约），结果存放在 `recvbuf` 中。同时，每个进程将当前的累加结果通过非阻塞发送（`MPI_Isend`）发送给右侧进程。变量 `offset` 用来指示当前处理的数据块位置，经过每次通信后，根据是否小于 `chunk` 来实现循环移位（环状数据传递）。

在第一阶段完成后，每个进程手中只有局部的规约结果（对应部分数据块）。在第二阶段，每个进程继续沿环形结构进行数据传递，每次从左侧接收下一个数据块，并将这个块发送给右侧进程。通过多次这样的传递，最终所有进程都能够收集到所有进程的规约数据，从而获得完整的规约结果。同样，`offset` 在此阶段用于标记当前接收和发送的数据块位置，并按环形更新。

所测试的通信时间

结点数	进程数	信息量	MPI_Allreduce	Naive_Allreduce	Ring_Allreduce
1	1	1000000	3.58649 ms	3.60705 ms	3.99643 ms
1	2	1000000	13.0441 ms	19.2556 ms	8.43633 ms
1	4	1000000	37.7931 ms	41.081 ms	20.2354 ms
1	8	1000000	38.473 ms	50.7029 ms	29.4699 ms
1	16	1000000	62.4114 ms	68.3344 ms	35.4147 ms
2	2	1000000	12.4353 ms	20.5214 ms	8.30861 ms
2	4	1000000	22.7318 ms	35.1597 ms	17.5974 ms
2	8	1000000	32.0544 ms	51.9706 ms	20.9093 ms
2	16	1000000	64.9 ms	70.0963 ms	38.637 ms
4	4	1000000	23.2219 ms	29.6614 ms	16.6084 ms
4	8	1000000	34.5432 ms	56.1485 ms	23.9982 ms
4	16	1000000	59.1651 ms	67.2461 ms	36.1007 ms
1	1	10000000	108.094 ms	100.854 ms	82.327 ms
1	2	10000000	222.15 ms	287.538 ms	172.751 ms
1	4	10000000	322.922 ms	556.958 ms	326.772 ms
1	8	10000000	510.021 ms	735.496 ms	382.951 ms
1	16	10000000	551.849 ms	779.535 ms	443.294 ms
2	2	10000000	196.777 ms	286.804 ms	142.459 ms
2	4	10000000	317.707 ms	631.855 ms	277.206 ms
2	8	10000000	422.841 ms	796.962 ms	356.986 ms
2	16	10000000	507.475 ms	750.233 ms	441.65 ms
4	4	10000000	250.07 ms	498.548 ms	172.044 ms
4	8	10000000	384.94 ms	666.263 ms	281.632 ms
4	16	10000000	574.309 ms	797.796 ms	376.601 ms
1	1	100000000	995.383 ms	993.924 ms	819.966 ms
1	2	100000000	3027.44 ms	3053.87 ms	1866.07 ms
1	4	100000000	3932.15 ms	5967.49 ms	2513.5 ms
1	8	100000000	4424.69 ms	6798.95 ms	3107.9 ms
1	16	100000000	5549.91 ms	6772.71 ms	4401.86 ms
2	2	100000000	2010.31 ms	2957.46 ms	1450.73 ms
2	4	100000000	2962.13 ms	5929.05 ms	2396.11 ms
2	8	100000000	3631.49 ms	7016.27 ms	3028.18 ms
2	16	100000000	4570.38 ms	7382 ms	4270.67 ms
4	4	100000000	2571.73 ms	4982.37 ms	1694.95 ms
4	8	100000000	3745.71 ms	6648.38 ms	2790.01 ms
4	16	100000000	5414.01 ms	7708.68 ms	4442.87 ms