高性能计算导论 第二次小作业报告

管思源 2021012702

一、代码简述（后附源码）

代码的实现基本与老师讲授的 Ring 算法一致。对于每个进程，将手上的数据切分成进程数 comm_sz 块。然后在第一阶段，进程 k 向进程 k+1（首尾相接）发送 (k-i) % comm_sz 位置的数据块（i 为循环变量，共执行 comm_sz-1 次），进程接受后于相同位置与自己手上的数据进行累加。第一阶段后的中间状态里，进程 k 持有 (k+1) % comm_sz 位置数据块的累加和。接下来的第二阶段里，进程 k 向进程 k+1（首尾相接）发送 (k+1-i) % comm_sz 位置的数据块（i 为循环变量，共执行 comm_sz-1 次），进程接受后直接替换相同位置的数据块。这样就完成了 Allreduce 功能。

需要指出的是，我在实现算法功能的基础上使用了非阻塞的发送以提高效率，接收则放到下一次循环。但接收还是使用了阻塞的函数，因为后续操作对接收到的数据有依赖。

二、测试数据（维度较多，还没来得及分析）

需要说明的是，这里加速比是以同等信息长度下、耗时最长的一种环境作为基准计算的。

信息量：112e0		MPI_Allreduce		Naive_Allreduce		Ring_Allreduce	
进程数	结点数	时间 (ms)	加速比	时间 (ms)	加速比	时间 (ms)	加速比
7	1	2.46741	2.46	0.178244	34.05	4.44011	1.37
	2	0.146932	41.30	0.139967	43.36	6.06884	1.00
	4	0.138284	43.89	0.133207	45.56	0.118346	51.28
14	1	0.223945	27.10	0.209246	29.00	5.58238	1.09
	2	2.221	2.73	0.224128	27.08	2.22522	2.73
	4	2.01239	3.02	0.220483	27.53	2.30532	2.63
28	1	3.72213	1.63	0.254662	23.83	0.207466	29.25
	2	0.244162	24.86	0.263547	23.03	0.258354	23.49
	4	5.07324	1.20	0.265223	22.88	0.287277	21.13
56	2	0.306007	19.83	0.352938	17.20	0.443726	13.68
	4	0.306725	19.79	0.338234	17.94	0.526663	11.52
112	4	0.386333	15.71	0.421243	14.41	3.01854	2.01

信息量：112e1		MPI_Allreduce		Naive_Allreduce		Ring_Allreduce	
进程数	结点数	时间 (ms)	加速比	时间 (ms)	加速比	时间 (ms)	加速比
7	1	0.299506	13.42	0.387071	10.38	0.185127	21.71
	2	0.308903	13.01	0.367882	10.92	0.207416	19.37
	4	0.291328	13.79	0.319729	12.57	0.238448	16.85
14	1	0.397722	10.10	0.518816	7.75	0.406275	9.89
	2	0.493583	8.14	0.467974	8.59	0.368712	10.90
	4	0.390026	10.30	0.451053	8.91	0.390678	10.29
28	1	0.429241	9.36	0.50983	7.88	0.503635	7.98
	2	0.450043	8.93	0.542202	7.41	0.605317	6.64
	4	0.448523	8.96	0.556245	7.22	0.592369	6.78
56	2	0.630176	6.38	0.706282	5.69	0.677655	5.93
	4	0.612244	6.56	0.640909	6.27	0.760629	5.28
112	4	2.58003	1.56	0.979357	4.10	4.01851	1.00

信息量：112e2		MPI_Allreduce		Naive_Allreduce		Ring_Allreduce	
进程数	结点数	时间 (ms)	加速比	时间 (ms)	加速比	时间 (ms)	加速比
7	1	0.868554	5.16	1.51786	2.96	0.583634	7.69
	2	0.64608	6.94	1.15718	3.88	0.630838	7.11
	4	0.828435	5.41	1.18277	3.79	0.790574	5.67
14	1	0.898751	4.99	1.85883	2.41	0.749476	5.98
	2	0.869472	5.16	1.57558	2.85	1.04095	4.31
	4	0.966716	4.64	1.72569	2.60	1.30402	3.44
28	1	1.0449	4.29	2.34796	1.91	1.00245	4.47
	2	0.997613	4.50	2.32233	1.93	1.02836	4.36
	4	1.07797	4.16	2.4943	1.80	1.02648	4.37
56	2	1.56684	2.86	2.77955	1.61	1.56143	2.87
	4	1.57275	2.85	2.84208	1.58	1.66636	2.69
112	4	3.03967	1.48	4.48559	1.00	2.92204	1.54

信息量：112e3		MPI_Allreduce		Naive_Allreduce		Ring_Allreduce	
进程数	结点数	时间 (ms)	加速比	时间 (ms)	加速比	时间 (ms)	加速比
7	1	5.94549	3.29	8.20528	2.38	4.31321	4.53
	2	4.57172	4.27	5.64159	3.46	3.27245	5.97
	4	4.33869	4.50	5.96874	3.27	3.34126	5.85
14	1	6.35639	3.07	7.73521	2.53	4.94207	3.95
	2	6.81904	2.86	9.53933	2.05	5.47708	3.57
	4	6.36579	3.07	10.7617	1.82	11.9664	1.63
28	1	6.08282	3.21	10.0029	1.95	6.51261	3.00
	2	6.8784	2.84	10.3949	1.88	6.87255	2.84
	4	6.04354	3.23	10.8313	1.80	8.20228	2.38
56	2	6.45154	3.03	11.5369	1.69	6.0836	3.21
	4	7.92387	2.47	19.5337	1.00	7.74658	2.52
112	4	10.0239	1.95	17.6646	1.11	8.09887	2.41

信息量：112e4		MPI_Allreduce		Naive_Allreduce		Ring_Allreduce	
进程数	结点数	时间 (ms)	加速比	时间 (ms)	加速比	时间 (ms)	加速比
7	1	46.6222	3.33	63.5522	2.44	35.2972	4.39
	2	38.4615	4.03	62.0298	2.50	28.0038	5.54
	4	54.9056	2.82	53.8928	2.88	30.0314	5.16
14	1	58.6619	2.64	74.7388	2.07	42.5728	3.64
	2	69.1457	2.24	77.8454	1.99	37.781	4.10
	4	56.4715	2.75	79.6712	1.95	46.6049	3.33
28	1	118.097	1.31	109.258	1.42	71.9161	2.16
	2	113.668	1.36	103.653	1.50	72.6214	2.13
	4	100.507	1.54	98.8512	1.57	74.5216	2.08
56	2	128.422	1.21	114.349	1.36	70.3305	2.20
	4	112.676	1.38	117.75	1.32	76.3228	2.03
112	4	131.82	1.18	155.034	1.00	93.6058	1.66

信息量：112e5		MPI_Allreduce		Naive_Allreduce		Ring_Allreduce	
进程数	结点数	时间 (ms)	加速比	时间 (ms)	加速比	时间 (ms)	加速比
7	1	559.57	3.39	779.83	2.43	366.111	5.18
	2	495.458	3.83	828.914	2.29	430.073	4.41
	4	503.504	3.76	810.814	2.34	298.301	6.35
14	1	623.311	3.04	846.445	2.24	529.627	3.58
	2	697.635	2.72	835.374	2.27	519.196	3.65
	4	604.881	3.13	857.364	2.21	415.336	4.56
28	1	1143.39	1.66	1142.16	1.66	813.474	2.33
	2	1060.36	1.79	1178.71	1.61	822.798	2.30
	4	943.98	2.01	1046.99	1.81	671.871	2.82
56	2	1265.25	1.50	1273.52	1.49	752.503	2.52
	4	987.501	1.92	1135.15	1.67	671.407	2.82
112	4	1061.93	1.79	1895.68	1.00	747.734	2.54

信息量：112e6		MPI_Allreduce		Naive_Allreduce		Ring_Allreduce	
进程数	结点数	时间 (ms)	加速比	时间 (ms)	加速比	时间 (ms)	加速比
7	1	3968.33	3.40	7154.86	1.88	3350.31	4.02
	2	4809.68	2.80	8395.39	1.61	3274.35	4.12
	4	3907.51	3.45	7117.46	1.89	3193.88	4.22
14	1	6975.41	1.93	8160.28	1.65	4722.04	2.85
	2	4825.46	2.79	7957.36	1.69	4423.48	3.05
	4	4990.15	2.70	8326.05	1.62	4245.3	3.17
28	1	9830.62	1.37	9746.14	1.38	8652.98	1.56
	2	12031.2	1.12	11025	1.22	9375.28	1.44
	4	11019.4	1.22	9530.2	1.41	7822.73	1.72
56	2	13055.6	1.03	11143.6	1.21	8862.76	1.52
	4	10687.5	1.26	9515.83	1.42	8409.26	1.60
112	4	13475	1.00	12104.1	1.11	9495.55	1.42

附：Ring_Allreduce 函数

```
void Ring_Allreduce(void* sendbuf, void* recvbuf, int n, MPI_Comm comm,
int comm_sz, int my_rank) {
    MPI_Request req;
    int src = (my_rank - 1 + comm_sz) % comm_sz;
    int dst = (my_rank + 1) % comm_sz;
    int slice = n / comm_sz;
    int offset = my_rank * slice;
    // Stage 1
    for (int i = 0; i < comm_sz; i++) {
        if (i != 0) {
            MPI_Recv((float*)recvbuf + offset, slice, MPI_FLOAT, src, i
- 1, comm, nullptr);
            MPI_Wait(&req, nullptr);
        }
        for (int j = 0; j < slice; j++) {
            if (i != 0) ((float*)recvbuf)[offset + j] +=
((float*)sendbuf)[offset + j];
            else ((float*)recvbuf)[offset + j] =
((float*)sendbuf)[offset + j];
        }
        if (i != comm_sz - 1) {
            MPI_Isend((float*)recvbuf + offset, slice, MPI_FLOAT, dst,
i, comm, &req);
            offset -= slice;
            if (offset < 0) offset = (comm_sz - 1) * slice;
        }
    }
    // Stage 2
    for (int i = 0; i < comm_sz; i++) {
        if (i != 0) {
            MPI_Recv((float*)recvbuf + offset, slice, MPI_FLOAT, src, i
- 1, comm, nullptr);
            MPI_Wait(&req, nullptr);
        }
        if (i != comm_sz - 1) {
            MPI_Isend((float*)recvbuf + offset, slice, MPI_FLOAT, dst,
i, comm, &req);
            offset -= slice;
            if (offset < 0) offset = (comm_sz - 1) * slice;
        }
    }
    return;
}
```