PDC 2024 Homework 2 Report

210012945 寿晨宸

并行算法设计思路

1. 任务是计算矩阵 C = A x B，为了方便表示，不妨假设 A，B 都是 n x n 的方阵。
2. 主进程将矩阵 A Scatter 给所有的 worker。每个 worker 接收到自己负责计算的部分矩阵 A（称为 A_block，是矩阵 A 的其中某 n/num_worker 行）。同时，主进程将矩阵 B Broadcast 给所有 worker。
3. 每个 worker 分别计算 A_block x B，得到 C_block。
4. 主进程 Gather 所有 worker 计算得到的 C_block，组合成 C = A x B。

编程思路

1. 以上算法的 Scatter, Broadcast 和 Gather 可以方便地使用 `MPI_Scatter`, `MPI_Bcast` 和 `MPI_Gather` 函数实现。这一方面降低了多次 Send 和 Recv 带来的开销，一方面也使总体的实现更加简洁。
2. 但是，这些函数要求发送的数据存放在连续的内存中，但普通的二维矩阵不一定存放在连续内存中，所以，我重新在 matrix.c 中实现了 `get_continuous_matrix_struct`，可以确保 `matrix_struct->mat_data` 中的数据是连续存放的（同时也实现了对应的 `free_continuous_matrix`）。并在加载矩阵时用`get_continuous_matrix_struct` 替代 `get_matrix_struct`（除了内存分布，其他功能方面两者是等价的）。
3. 同时，为了能够使用这些函数，我另外定义了一些全局变量。

测试结果

我在 ./test.sh 中实现了测试脚本，并把结果记录在了 mpi.log 和 seq.log 中。

以下是测试结果：

MPI

MPI 并行算法的运行结果，其中 Worker Number 代表使用的进程数，Matrix N 代表测试使用 NxN 的矩阵，表项为运行时间。

Worker Number/Matrix N	2	4	8	16	32	64	128
64	0.002139	0.002115	0.002607	0.004094	0.004068	0.005139	x
128	0.00479	0.004184	0.005342	0.004314	0.004425	0.005609	0.009353
256	0.017435	0.011829	0.010557	0.012377	0.020662	0.008231	0.012435
512	0.260897	0.137164	0.069938	0.053257	0.0361	0.040034	0.145812
1024	3.832286	1.827503	1.005776	0.553481	0.266648	0.174185	0.173142
2048	73.558947	37.104168	18.910711	12.094203	6.361713	4.00842	1.593901
4096	x	x	x	x	x	66.6778	41.352301

Seq

sequential.c 线性算法的运行结果。

Matrix N	Sequence Execution time
64	0.000183
128	0.001761
256	0.015627
512	0.11306
1024	0.920368
2048	13.0013
4096	256.328

Ratio

每个规模的矩阵，线性算法运行时间和最快的 MPI 并行算法运行时间的比值。

Matrix N	Ratio of Seq/MPI_MIN
64	0.0865248
128	0.420889
256	1.89855
512	3.13186
1024	5.31568
2048	8.15693
4096	6.19864

分析

当矩阵规模较小时，MPI 并行调度和通信的开销大于并行带来的收益，所以加速比较低。

当矩阵规模较大时，MPI 并行算法起到了较好的加速效果。

另外，注意到，当 MPI 并行时分配的 worker 数较小时，其性能往往比线性的情况下更差。

总体而言，算法的性能符合预期。