lab7-MPI并行编程

朱宏基　　220110131　　大二(上)

实验环境:

- **OS版本**: Ubuntu 22.04.2 LTS
- **gcc版本**: 11.4.0
- **CPU**:
  i. 型号: AMD Ryzen 5 4600H with Radeon Graphics
  ii. 频率: 均为 2994.389 MHz
  iii. 核数: 12
- **内存**: 7.5 Gi

GEMM的MPI实现:

实验方案:

采用MPI实现矩阵乘法的并行计算。
首先调用MPI_Init初始化MPI并行环境，通过各个进程的rank分配任务：
接着主进程负责分配以及初始化矩阵A、B、C，并通过MPI的通信函数MPI_Send将矩阵A、B、C分块传给其他进程，这里分块的大小blocksize根据矩阵规模和进程数决定；
然后其他进程在获取到这些分完块的矩阵A、B、C后，各个进程用MPI_Recv函数收集这些信息，然后进行矩阵乘法计算，并把结果矩阵C的分块传回主进程；
最后主进程接收其他进程的计算结果进行汇总，调用打印函数输出最终的结果矩阵C，再调用MPI_Finalize结束MPI并行环境。

关键代码:

```
int main(int argc, char *argv[])
{
    int rank, num_procs;
    int size = ; // 矩阵的大小,根据mpi_test_data.m的数据硬编码为指定大小
    double *A, *B, *C;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &num_procs);

    int block_size = size / (num_procs - 1);

    if (rank == 0)
    {
        // 初始化矩阵A、B、C
        // 注意：这里只有主进程需要分配内存
        double A[] = {
            // 填入mpi_test_data.m中的数据
        };
        double B[] = {
            // 填入mpi_test_data.m中的数据
        };
        C = (double *)malloc(sizeof(double) * size * size);

        for (int i = 0; i < size; i++)
            for (int j = 0; j < size; j++)
                C[i * size + j] = 0.0;
```

```
    // 将A、B分块发送给其他进程
    for (int i = 1; i < num_procs; i++)
    {
        // 使用MPI_Send将A和B的块数据发送给其他进程
        MPI_Send(A + (i - 1) * block_size * size, block_size * size, MPI_DOUBLE, i, 0, MPI_COMM_WORLD);
        MPI_Send(B, size * size, MPI_DOUBLE, i, 1, MPI_COMM_WORLD);
    }

    // 接收其他进程发送回来的结果C
    for (int i = 1; i < num_procs; i++)
    {
        // 使用MPI_Recv接收各个进程发送回来的结果
        MPI_Recv(C + (i - 1) * block_size * size, block_size * size, MPI_DOUBLE, i, 3, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    }

    // 打印最终结果C
    PrintMatrix(C, size);

    // 释放内存
    free(C);
}
else
{
    // 分配 partialA 和 partialC 的内存空间
    double *partialA = (double *)malloc(sizeof(double) * block_size * size);
    double *partialC = (double *)malloc(sizeof(double) * block_size * size);
    double *B = (double *)malloc(sizeof(double) * size * size);
    // 接收主进程发送过来的A和B
    MPI_Recv(partialA, block_size * size, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    MPI_Recv(B, size * size, MPI_DOUBLE, 0, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
```

```
    // 对接收的数据进行计算
    for (int i = 0; i < block_size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            double temp = 0;
            for (int k = 0; k < size; k++)
            {
                temp += partialA[i * size + k] * B[k * size + j];
            }
            partialC[i * size + j] = temp;
        }
    }

    // 将计算结果发送给主进程
    MPI_Send(partialC, block_size * size, MPI_DOUBLE, 0, 3, MPI_COMM_WORLD);

    // 释放内存
    free(partialA);
    free(B);
    free(partialC);
}

MPI_Finalize();
return 0;
}
```

运行结果截图:

- 4×4规模:

```
hoshino_july@LAPTOP-S40B7Q22:~/hpc_practice/lab7-mpi$ mpicc gemm_mpi.c -o gemm_mpi
hoshino_july@LAPTOP-S40B7Q22:~/hpc_practice/lab7-mpi$ mpirun -n 5 ./gemm_mpi
-6.485473e-01 -6.974358e-01 -2.888289e-01 6.850690e-02
2.435675e-01 -8.744635e-01 -1.614911e-01 1.371329e-01
5.139494e-01 -1.185891e+00 -4.575017e-01 -3.413287e-01
-2.864045e-01 4.127308e-02 -5.261487e-04 1.859551e+00
```

- 8×8规模:

```
hoshino_july@LAPTOP-S40B7Q22:~/hpc_practice/lab7-mpi$ mpicc gemm_mpi.c -o gemm_mpi
hoshino_july@LAPTOP-S40B7Q22:~/hpc_practice/lab7-mpi$ mpirun -n 5 ./gemm_mpi
-7.024896e-01 -6.175928e-01 7.138585e-01 6.358140e-02 1.412361e+00 9.497799e-01 -1.198863e-01 -1.091875e-01
5.495472e-01 -5.828320e-02 2.023827e-01 -1.409116e+00 2.501541e+00 2.480544e+00 8.435441e-01 -7.664718e-02
3.326340e-02 1.469986e+00 -1.598574e-02 -6.368557e-02 1.211029e+00 1.221015e+00 1.691703e+00 -1.250213e+00
1.568243e+00 1.036256e+00 -3.969601e-01 -3.099131e-02 1.298360e+00 -5.740629e-02 1.148578e+00 -9.824464e-01
3.303513e-01 1.561958e+00 -1.425547e+00 1.551175e+00 -5.565088e-01 -2.013690e+00 1.271709e-01 -7.343229e-01
6.956050e-01 1.009126e+00 -1.748625e-01 1.339494e+00 -2.857657e-01 -1.493150e+00 1.240700e+00 -5.529886e-01
-1.375815e+00 -8.533851e-01 6.912177e-01 -1.710137e-01 8.812949e-01 1.767113e+00 4.605126e-01 8.840338e-02
2.721833e-01 -3.181251e-02 -1.034846e+00 4.047739e-01 -4.389421e-01 -1.226528e+00 -1.319781e-01 -3.960062e-01
```

实验遇到的问题与解决方案:

并无遇到问题。