

金融大数据-作业2

施宇 191250119

为什么需要并行计算？

贯穿整个计算机技术发展的核心目标是提高计算性能。单核处理器性能提升已经接近极限；芯片集成度已进入极小尺度级别，集成度不可能无限制提高；处理器的指令集并行度提升接近极限；处理器速度和存储器速度差异越来越大；功耗和散热大幅增加超过芯片承受能力。向多核并行计算发展已经成为必然趋势，并行计算也成为了应用领域超大计算规模和复杂度的解决方案。越来越多的研究和应用领域需要使用并行计算技术，并行计算技术将对传统计算技术产生革命性的影响。

并行计算按照系统类型划分，可以分为哪几种？简述每一种系统类型的特点。

- **多核/众核并行计算系统MC或芯片级多处理CMP**：一块芯片上具有多个处理器核心，能够同时运行多个线程。具有较为紧密的耦合度，低可扩展性和低功耗的特点。
- **对称多处理系统SMP**：多个相同类型处理器通过总线连接并共享存储器。
- **大规模并行处理MPP**：专用内联网连接一组处理器形成的一个计算系统。
- **集群Cluster**：网络连接的一串商品计算机构成的计算系统
- **网格Grid**：用网格连接远距离分布的一组异构计算机组成的计算系统。其耦合性较为分散，可扩展性较高，能耗较高。

并行计算按照并行程序设计方法分类，可以分为哪几种？简述每一种方法的特点。

- **共享内存变量(Shared Memory Variables)**：多线程共享存储器变量方式进行并行程序设计，会引起数据不一致性，导致数据和资源访问冲突，需要引入同步控制机制；Pthread, OpenMP：共享内存式多处理并行编程接口。
- **消息传递方式(Message Passing)**：对于分布式内存结构，为了分发数据和收集计算结果，需要在各个计算节点间进行数据通信，最常用的是消息。MPI提供消息传递并行编程接口标准。
- **MapReduce方式**：Google公司提出的MapReduce并行程序设计模型，是当时最易于使用的并行程序设计方法，广泛使用于搜索引擎等大规模数据并行处理。

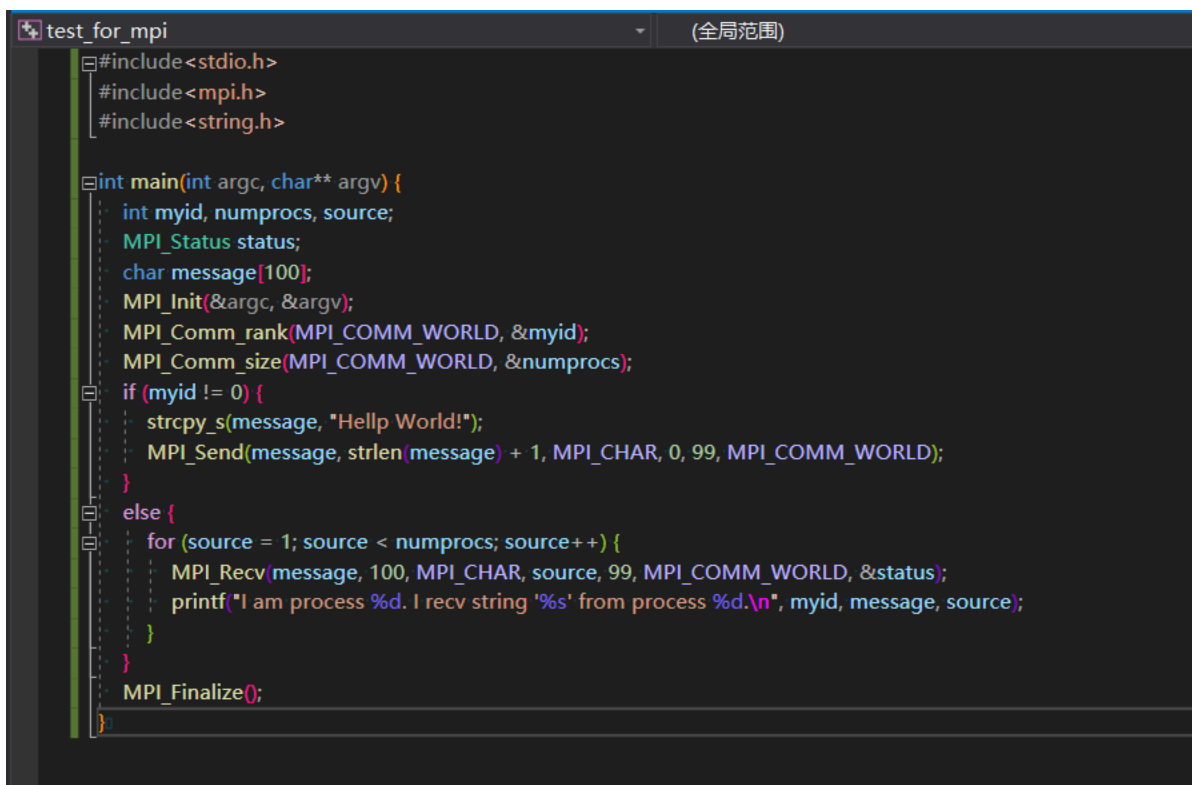
MPI提供哪几种通信方式？简述每种通信方式对应的接口。

- 点对点通信：
 - 同步通信：阻塞式通信，等待通信操作完成后才返回
 - MPI_Send(buf, count, datatype, dest, tag, comm)：发送一个消息
 - MPI_Recv(buf, count, datatype, source, tag, comm, status)：接收消息
 - 异步通信：非阻塞式通信，不等待通信操作完成即返回
 - MPI_Isend(buf, count, datatype, dest, tag, comm, request)：异步发送
 - MPI_Irecv(buf, count, datatype, source, tag, comm, status, request)：异步接收消息
 - MPI_Wait(request, status)：等待非阻塞数据传输完成
 - MPI_Test(request, flag, status)：检查是否异步数据传输确实完成
- 节点集合通信：
 - 同步(Barrier)
 - MPI_Barrier：设置同步障使所有进程的执行同时完成

- 数据移动(Data movement)
 - MPI_BCAST: 一对多的广播式发送
 - MPI_GATHER: 多个进程的消息以某种次序收集到一个进程
 - MPI_SCATTER: 将一个信息划分为等长的段依次发送给其他进程
- 数据规约(Reduction)
 - MPI_Reduce(sendbuf, recvbuf, count, datatype, op, root, comm): 将一组进程的数据按照指定的操作方式规约到一起并传送给一个进程
- 用户自定义的复合数据类型传输:
 - MPI_Type_struct: 创建新数据类型
 - MPI_Type_commit: 定义新数据类型

尝试在单机上安装并运行MPICH，并运行讲义P66页的简单示例。（运行结果截图）

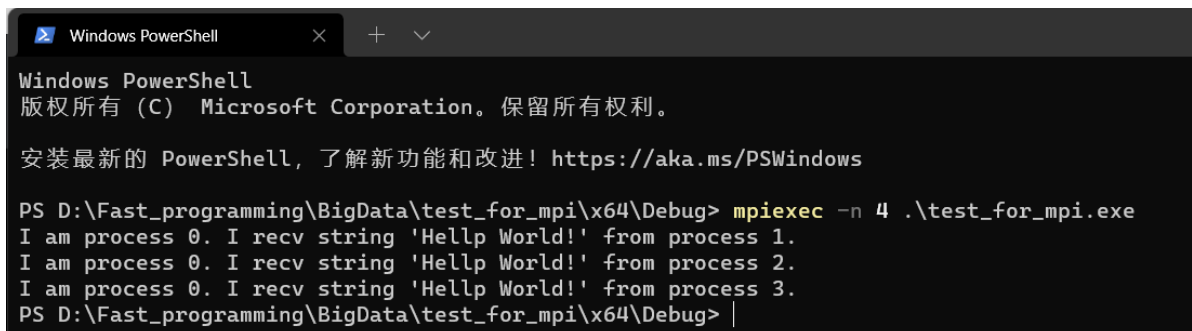
在Windows11系统上的Visual Studio 2019编写代码并编译：



```
#include<stdio.h>
#include<mpi.h>
#include<string.h>

int main(int argc, char** argv) {
    int myid, numprocs, source;
    MPI_Status status;
    char message[100];
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    if (myid != 0) {
        strcpy_s(message, "Hellp World!");
        MPI_Send(message, strlen(message) + 1, MPI_CHAR, 0, 99, MPI_COMM_WORLD);
    }
    else {
        for (source = 1; source < numprocs; source++) {
            MPI_Recv(message, 100, MPI_CHAR, source, 99, MPI_COMM_WORLD, &status);
            printf("I am process %d. I recv string '%s' from process %d.\n", myid, message, source);
        }
    }
    MPI_Finalize();
}
```

在命令行运行程序：



```
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！ https://aka.ms/PSWindows

PS D:\Fast_programming\BigData\test_for_mpi\x64\Debug> mpiexec -n 4 .\test_for_mpi.exe
I am process 0. I recv string 'Hellp World!' from process 1.
I am process 0. I recv string 'Hellp World!' from process 2.
I am process 0. I recv string 'Hellp World!' from process 3.
PS D:\Fast_programming\BigData\test_for_mpi\x64\Debug> |
```

