**中山大学计算机院本科生实验报告**

**（2024学年春季学期）**

**课程名称：并行程序设计 批改人：**

|  |  |  |  |
| --- | --- | --- | --- |
| **实验** | **7-MPI并行应用** | **专业（方向）** | **计算机科学与技术** |
| **学号** | **21307174** | **姓名** | **刘俊杰** |
| **Email** | **liujj255@mail2.sysu.edu.cn** | **完成日期** | **2024/5/14** |

# **实验目的**

**MPI并行应用**

**使用MPI对快速傅里叶变换进行并行化。**

**问题描述：阅读参考文献中的串行傅里叶变换代码(fft\_serial.cpp)，并使用MPI对其进行并行化。**

**要求：1. 并行化：使用MPI多进程对fft\_serial.cpp进行并行化。为适应MPI的消息传递机制，可能需要对fft\_serial代码进行一定调整。**

**2. 优化：使用MPI\_Pack/MPI-Unpack或MPI\_Type\_create\_struct对数据重组后进行消息传递。**

**3. 分析：a) 改变并行规模（进程数）及问题规模（N），分析程序的并行性能；b) 通过实验对比，分析数据打包对于并行程序性能的影响；c) 使用Valgrind massif工具集采集并分析并行程序的内存消耗。注意Valgrind命令中增加--stacks=yes 参数采集程序运行栈内内存消耗。**

# **实验过程和核心代码**

**2.1 解析串行源代码**

## **源代码实现了快速傅里叶变换（FFT）以及其精度验证:**

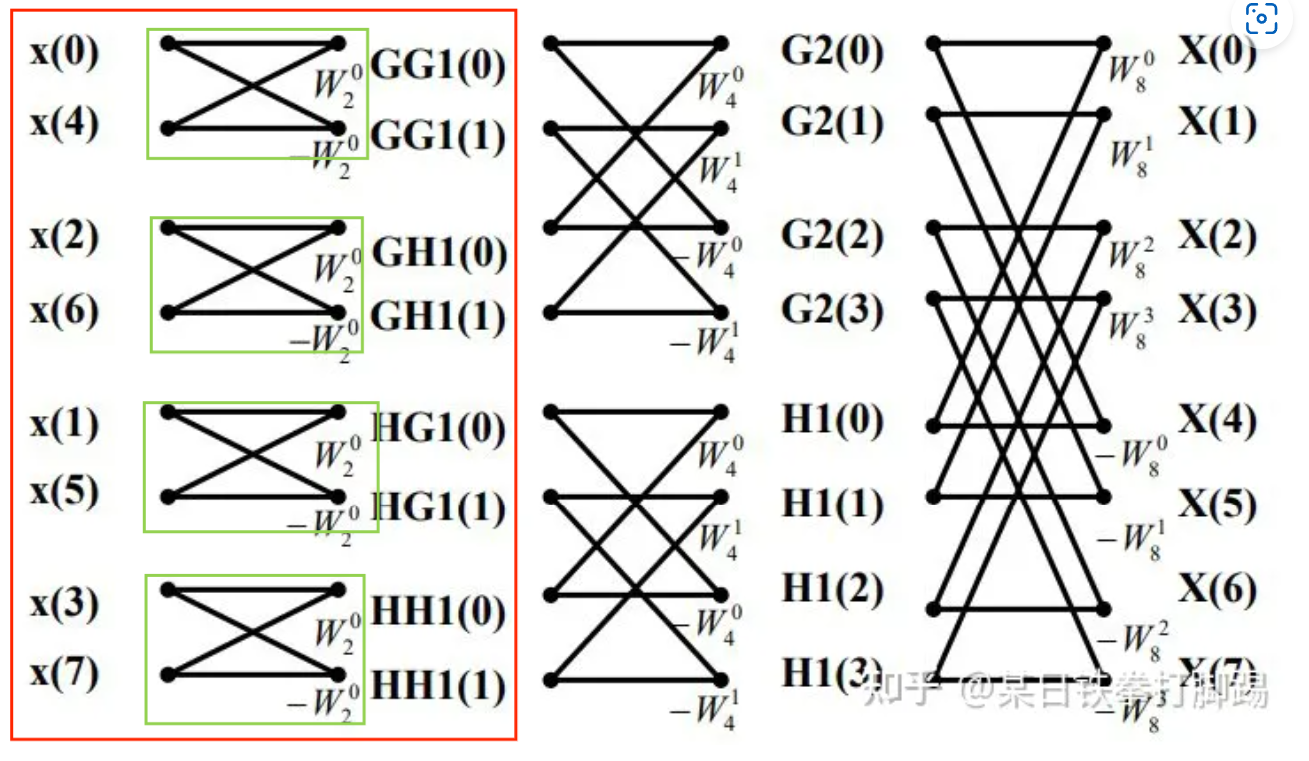
## **①首先数组w 记录 数组x和y记录fft的中间结果和最终结果 数组z记录初始值，将n从2迭代至1048576实现fft快速傅里叶变换和精度的计算。**

## **②调用cffti()用于计算FFT中需要的的正弦和余弦表。**

## **③cfft2函数中sgn控制FFT方向的参数，它的取值决定了执行的是正向FFT还是反向FFT。具体来说，sgn的取值为+1表示执行正向FFT，取值为-1表示执行反向FFT。**

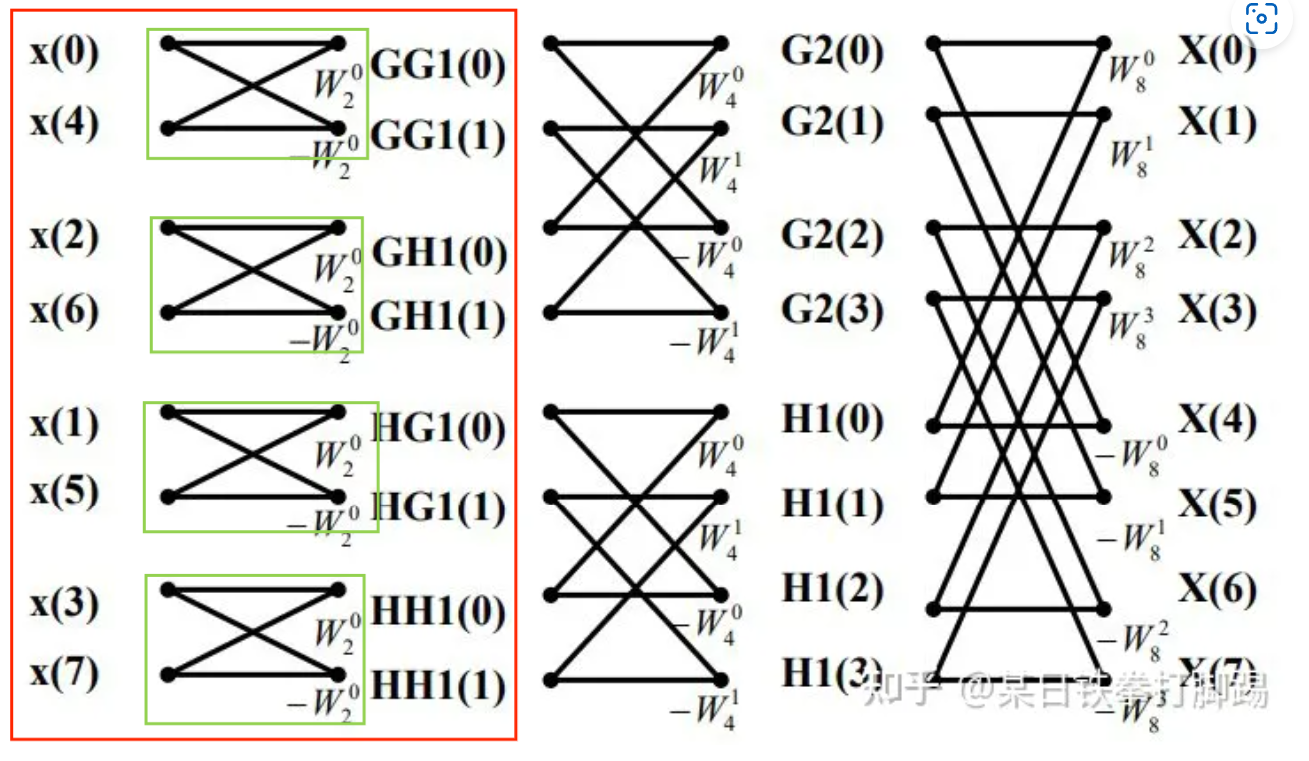
**④cfft2()中调用step()函数实现了fft快速傅里叶变化中的蝶式计算**

**中图中的红色框部分是step()函数的计算,绿色框部分是step()每次循环举行的一次蝶式计算。**

****

**⑤最后每次n迭代中计算Error Time和MFLOPS等数据。**

## **2.2 并行思路**

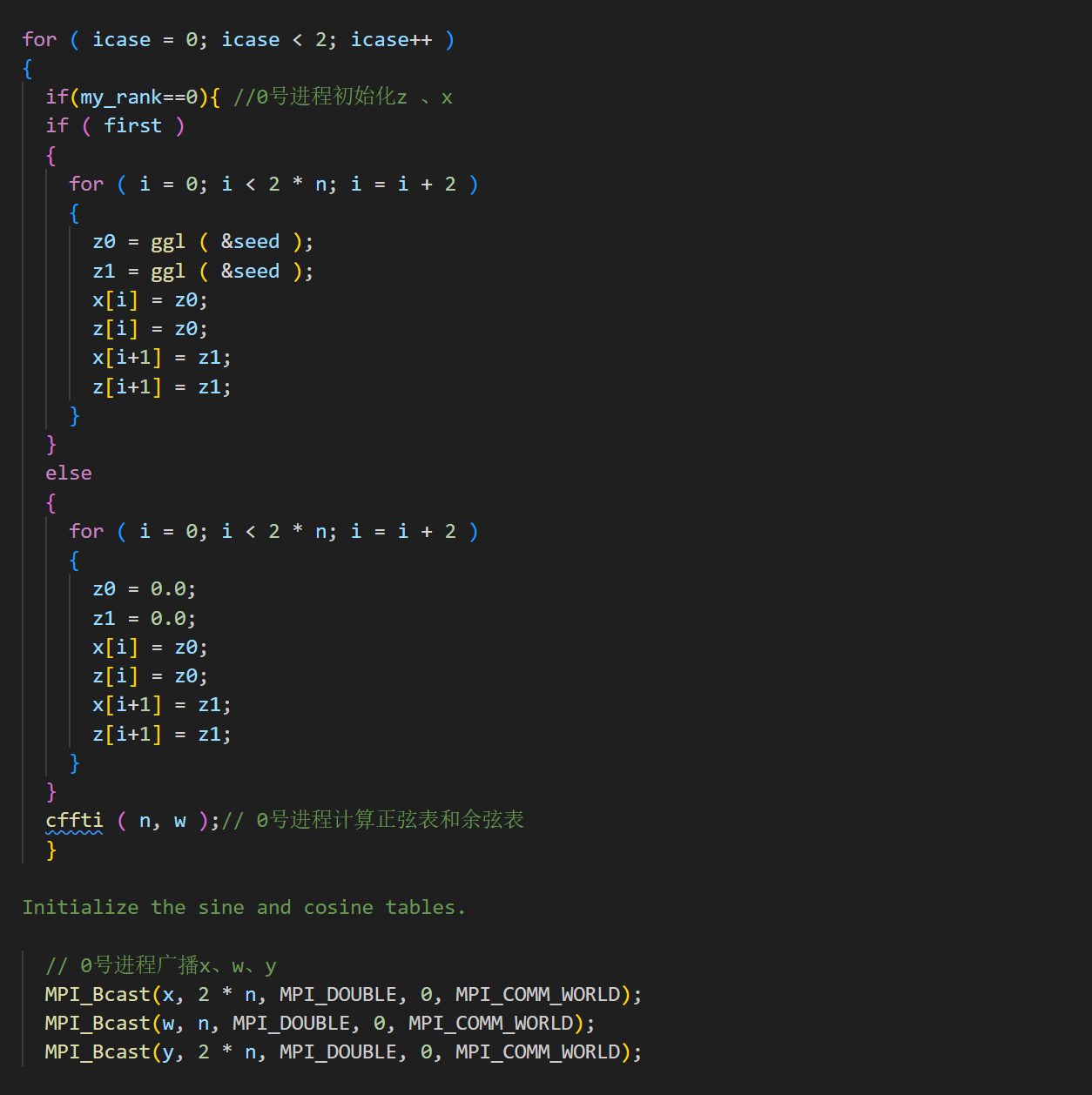
****

**从串行fft的源代码中,可以看出并行加速的部分在蝶式计算中(即step函数)，因为其在一个阶段(红色框部分)的计算中每组蝶式计算的数据之间没有依赖性,可以使用多进程计算。**

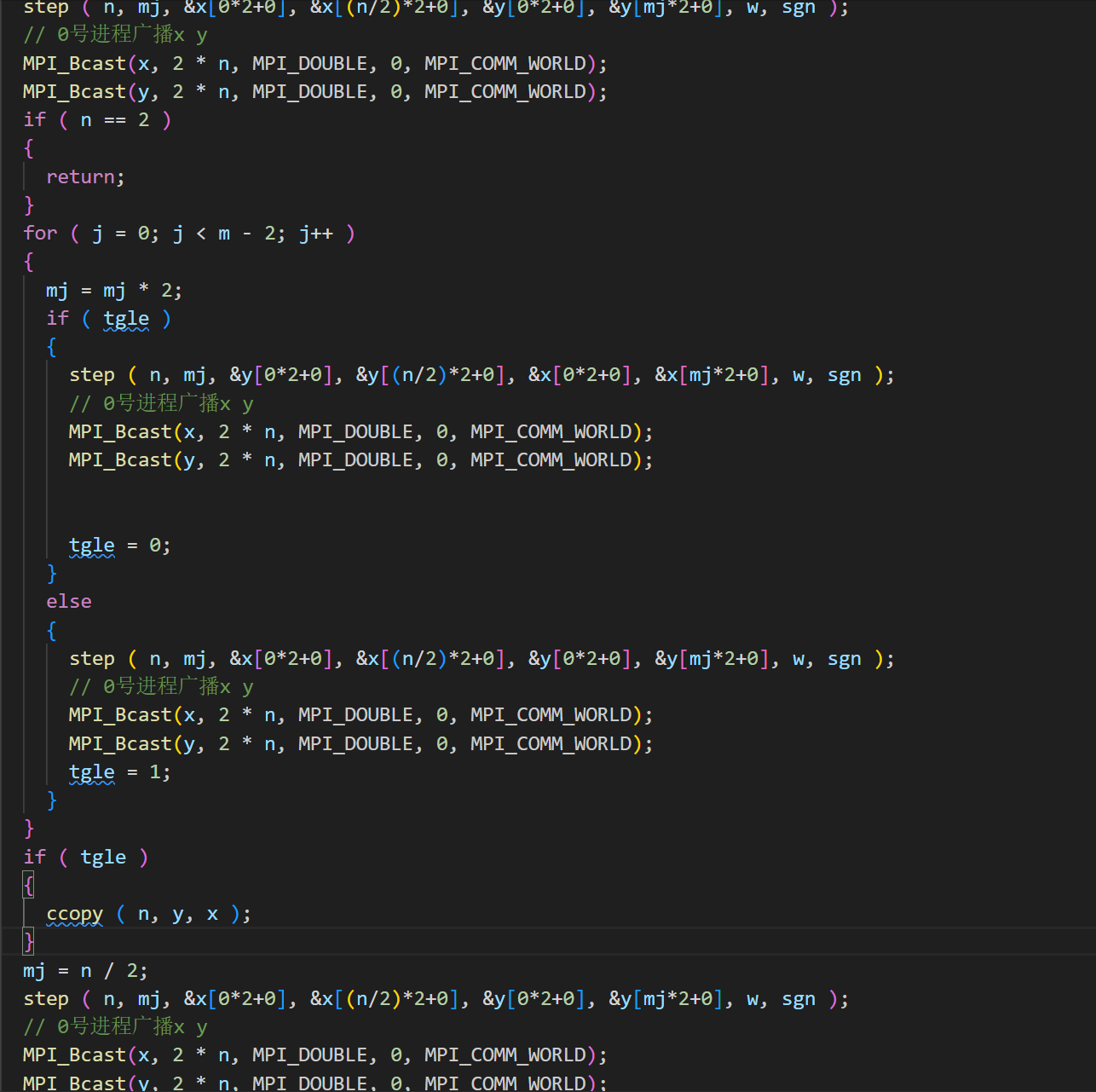
**在每一组阶段的计算完成后,需要每个进程的结果数据一致性:可以将每组蝶式计算的结果发送给0号进程,当0号进程实现接收后,再将结果广播给其他进程，其他进程接收后实现数据一致性，再执行step()进行下一步的fft蝶式计算。**

**2.3 并行代码实现**

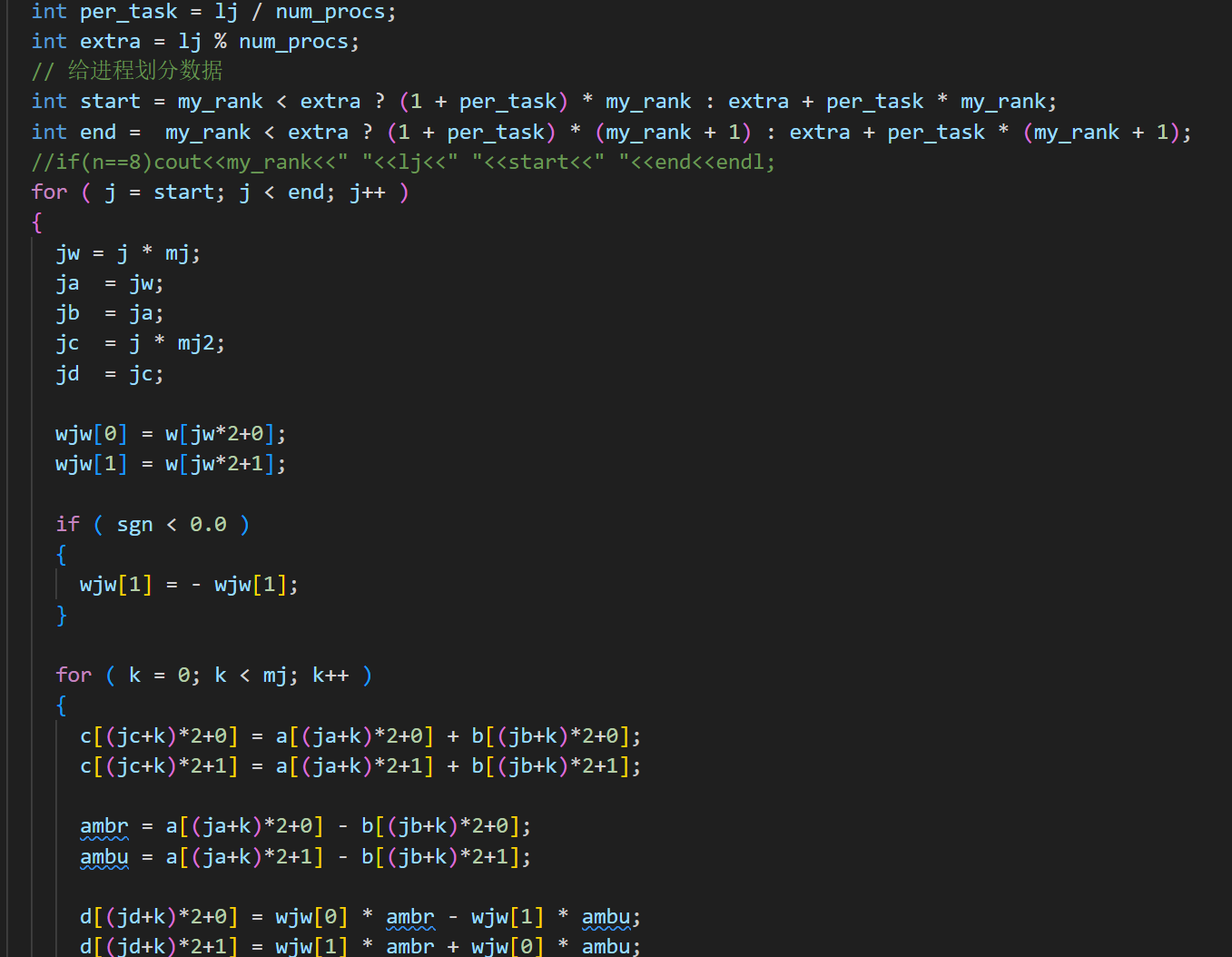
**①0号进初始化w、x 、z,并进行广播**

****

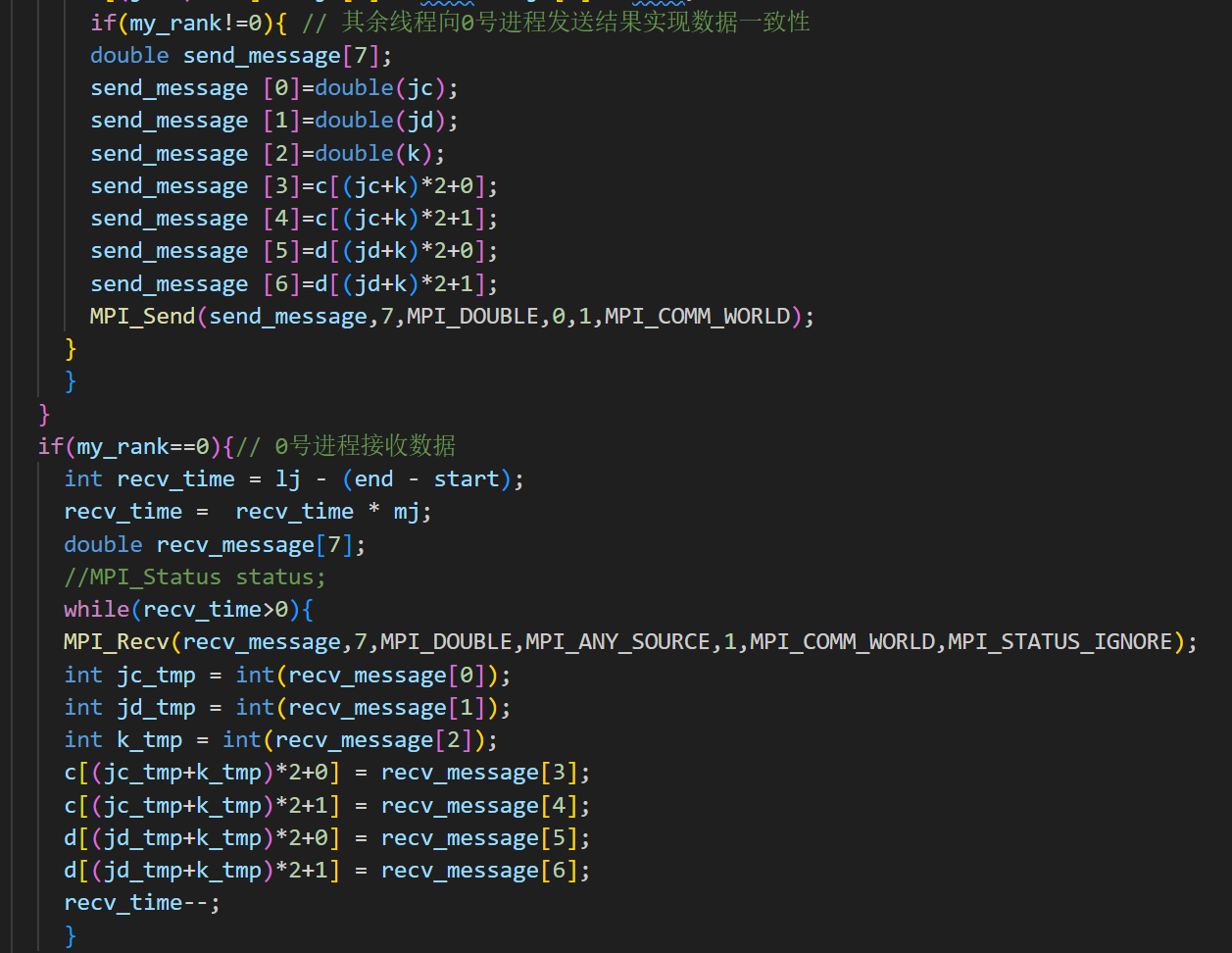
**②cfft2函数中每次调用step计算出结果后,0号进程将x、y数据广播给其他进程**

****

**③step()函数中将计算数据划分给不同的线程计算,每个进程计算负责的结果**

****

**④每次蝶式计算结束后,其它进程将结果数据发送给0号进程,0号进程等待接收其它进程发送的结果**

****

**2.4 问题及改进**

**2.4.1 问题**

**但是仔细看蝶式计算后0号进程数据的接收,可以发现0号进程必须等待接收其他进程每次蝶式计算的结果,而其他进程又必须等待0号进程接收完后再广播给它们才能继续执行,这就造成了实际的运行效果还是串行的。**

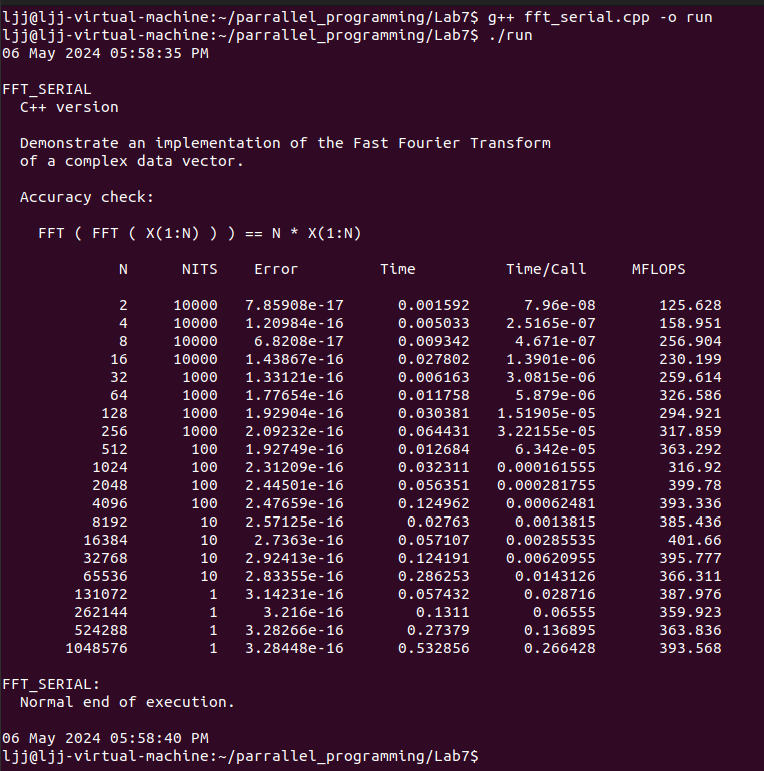
**2.4.2 改进**

**需要对每个进程做更好的数据划分,实现每个进程负责计算的结果是连续的,于是这样将数据发送给0号进程就可以每个进程用1个send就可以实现,避免上述的等待问题。**

# **实验结果**

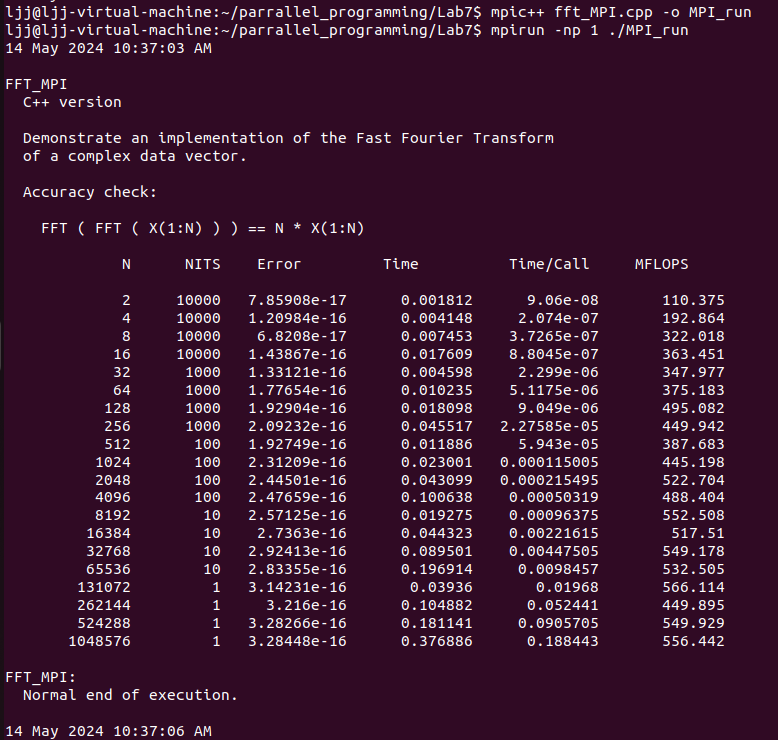
**3.1 实验结果展示**

## **3.1.1 串行代码实验结果**

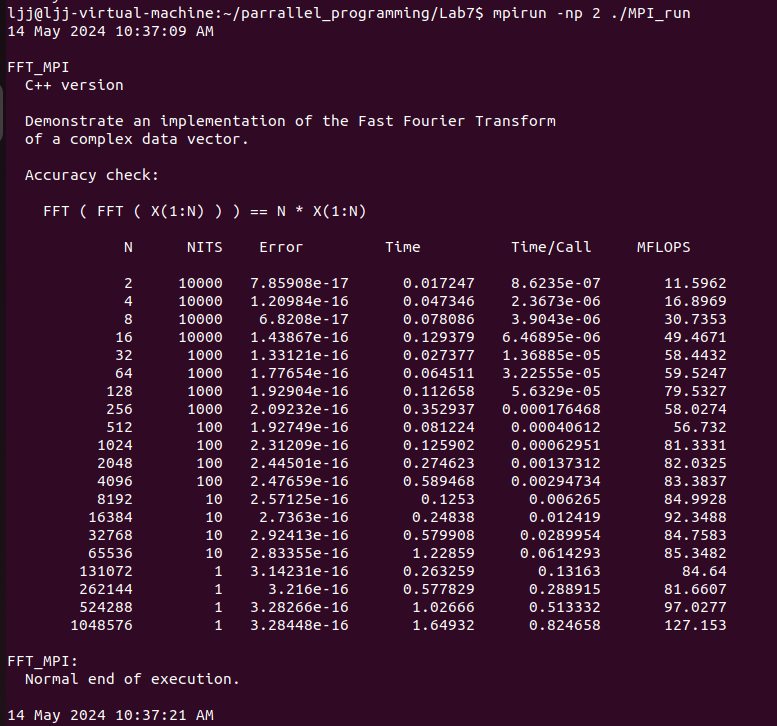
****

**3.1.2 并行代码实验结果**

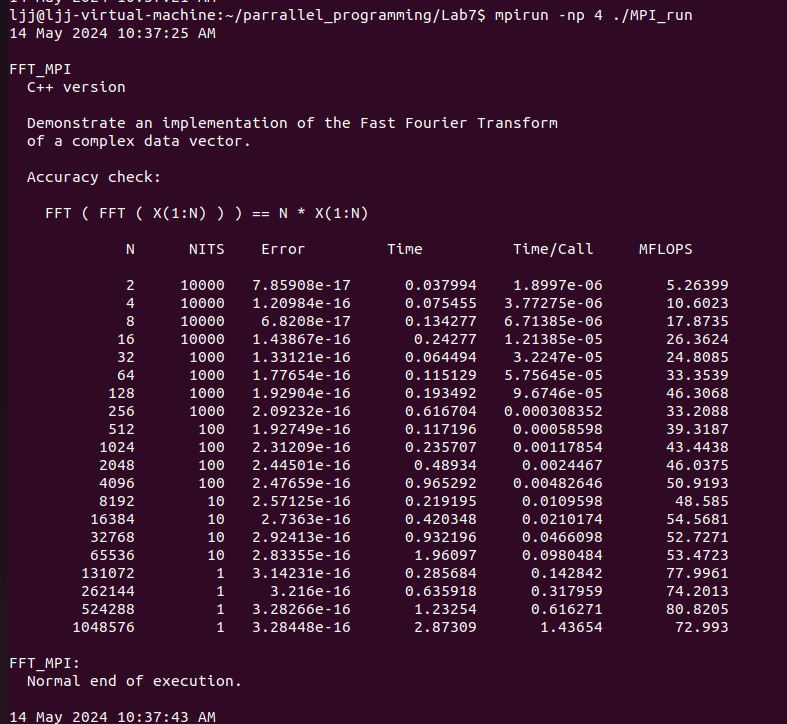
**①1个进程**

****

**②2个进程**

****

**③3个进程**

****

## **3.2 实验结果分析**

**①可以看到实验结果中的串行计算和并行计算中的Error是相同的,说明fft的结果是正确的。**

**②但是可以看到并行计算的时间反而增加了 MFLOPS也下降了,这是因为在实现的并行代码中0号进程要等待接收其他进程每次循环计算的结果,这导致了这部分的效果和串行实现没有区别,而并行中还有广播和发送接收的开销,导致并行fft的时间增加了,MFLOPS也下降了。**

# **实验感想**

**4.1 问题**

**代码蝶式计算后0号进程数据的接收,可以发现0号进程必须等待接收其他进程每次蝶式计算的结果,而其他进程又必须等待0号进程接收完后再广播给它们才能继续执行,这就造成了实际的运行效果还是串行的。**

**4.2 改进**

**需要对每个进程做更好的数据划分,实现每个进程负责计算的结果是连续的,于是这样将数据发送给0号进程就可以每个进程用1个send就可以实现,避免上述的等待问题。**