南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

Nanjing University of Science and Technology

School of Computer Science and Engineering

《高性能计算引论》第二次作业

姓名：罗文水

学号：*918106840738*

班级：计科一班

课程：高性能计算引论

授课教师：李翔宇

2021 年 5 月 26 日

1 问题一

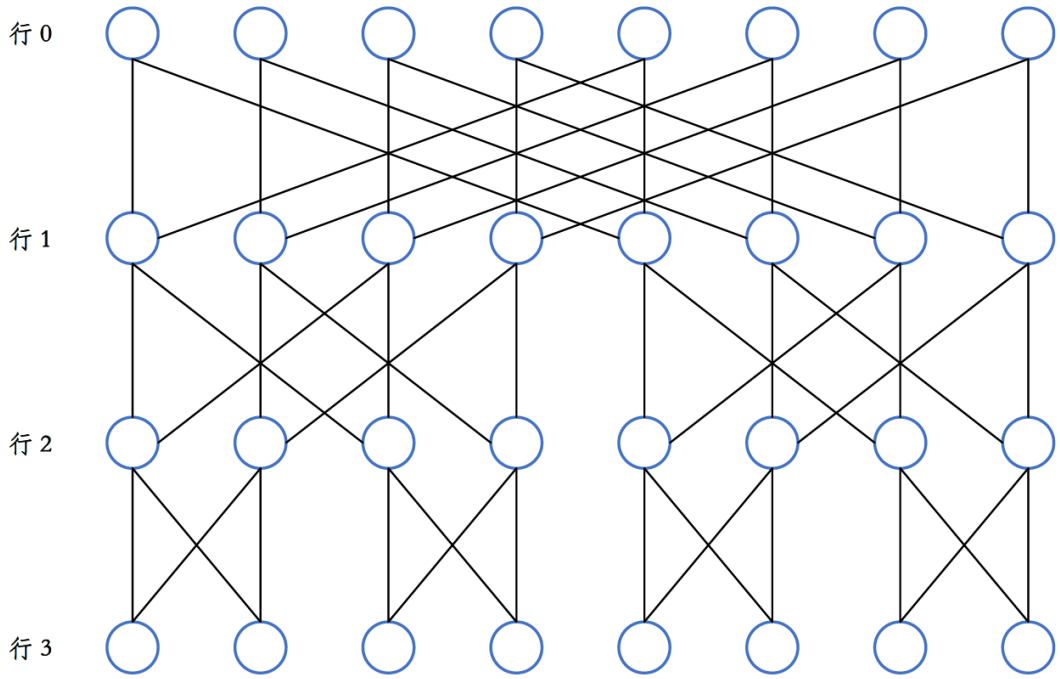

图 1: 三层蝶形网络结构图

该网络的各项参数如下所示：

1. **网络直径**：大小为 $(k+1)\cdot 2^k$ 的网络的直径为 $2k$。

   证明：假设第 i 行从左至右分别按照二进制位进行编号，编号记为:

   $$a_{k-1}^{(i)}a_{k-2}^{(i)}a_{k-3}^{(i)}\cdots a_0^{(i)}$$

   则蝶形网络第 s 行与第 $s+1$ 行之间的互联逻辑为如下蝶形函数 (即将第 s 层的输入元 m 第 s 位取反，变为 m'，并建立 m 到 m' 之间的连接。

   $$\varepsilon_s(x^{(s)}) = x_{k-1}^{(s)}x_{k-2}^{(s)}\cdots\overline{x_{k-1-s}^{(s)}}\cdots x_0^{(s)} \tag{1}$$

   或者，在两层之间，编号相同的同样建立连接。连接函数如下：

   $$\varepsilon_s(x^{(s)}) = x_{k-1}^{(s)}x_{k-2}^{(s)}\cdots x_{k-1-s}^{(s)}\cdots x_0^{(s)} = x^{(s)} \tag{2}$$

   根据上述公式，从第 0 行中选择一个编号为 $m^{(0)}$ 的节点，从第 $k+1$ 行中选择一个编号为 $\overline{m}^{(k)}$ 的编号，该编号的二进制位为 m 的每一位取反，并选择一个编号与 $m^{(0)}$ 相同的节点 $m^{(k)}$，则 $m^{(0)}$ 与 $\overline{m}^{(k)}$ 之间的距离一定为 k，即通过 k 次二进制位取反可以得到，其次，从 $m^{(0)}$ 到 $m^{(k)}$，二进制位不变，也存在一条从第 0 行到第 k 行的长度为 k 的路径，当取 $m^{(0)} = 000\cdots00$ 时，

从 $m^{(k)}$ 节点到 $\overline{m}^{(k)}$ 之间的距离为 $2\cdot k$。且由于网络中的节点都是通过变换二进制位进行连接，从而任意两个节点之间的距离不能超过 $2\cdot k$ ，故该网络的直径是 $2\cdot k$。

2. **等分宽度**：该网络的等分宽度为 2^k。

   证明：按照第一问的分析方法，第 0 层到第 1 层之间的互联函数实现的逻辑功能是 $\varepsilon_0(x^{(0)})$，即将最高位取反。如果考虑将网络划分为左右两部分，则在网络左边以及右边组内编号的二进制位首位相同。由于互联函数中只有 $\varepsilon_0(x^{(0)})$ 可以改变高位，其他函数 $\varepsilon_s(x^{(s)})\quad \forall\, s\neq 0$ 都是指在组内进行连接，故对于计算两组之间的 link 数量，只需要计算互联函数 $\varepsilon_0(x^{(0)})$ 连接的边数量，由于该函数是作用在第 0 层上的，而第 0 层一共有 2^k 个结点，所以可以通过删除 2^k 条边使得网络等分为两个相同节点数的网络。而分割该网络为两个相同大小的子网络，必须至少分割 2^k 条 link，故一个层数为 k 的蝶形网络等分宽度为 2^k。

3. **边连通度**：一个层数为 k 的蝶形网络的边连通度为 2。

   证明：从蝶形网络的构造上来看，只有第一层和最后一层的节点度为 2，其他节点的节点度都为 4，即对第 0 层的节点 x 而言，x 连接第二层的 link 代表两种选择，分别是高位不变与高位取反。将两个 link 删除即可将 x 从网络中分离出来。而至少要删除 2 条 link 才能使网络不连通。从而该网络的节点度为 2。

2 问题二

（1）运行 Hello World 程序三次得到的结果分别如下所示。

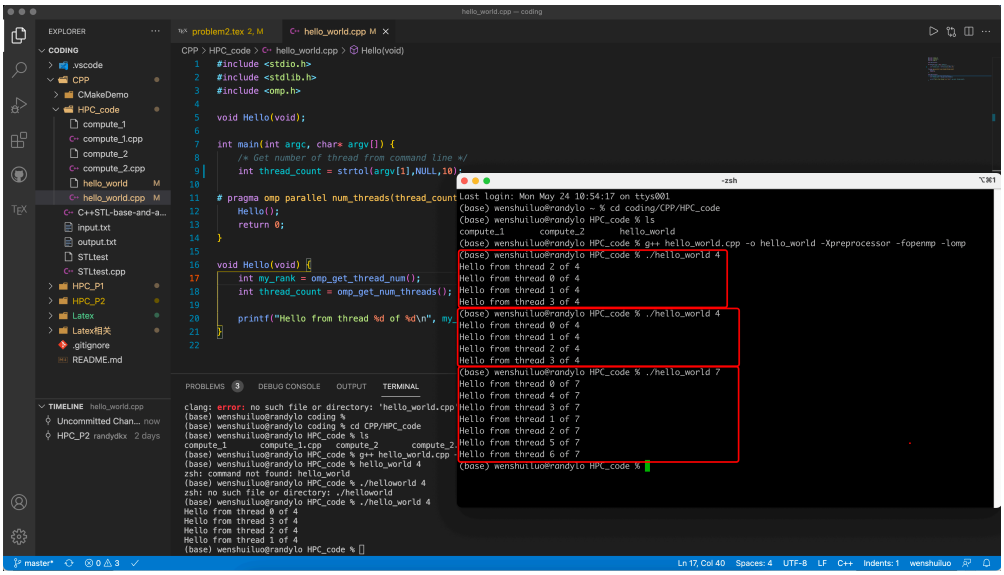

图 2: 问题一三次运行截图 1

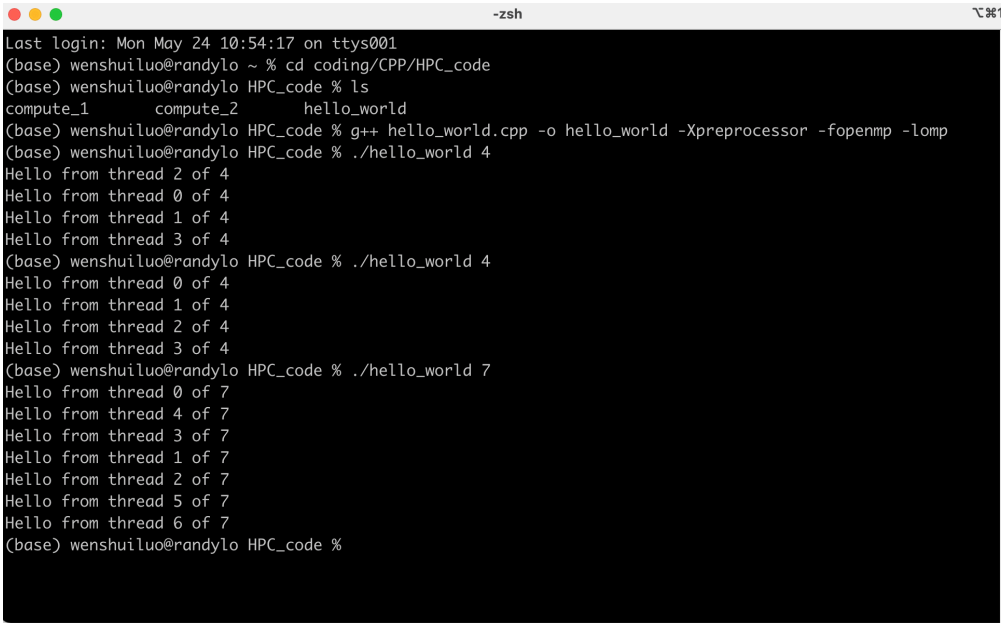

图 3: 问题一三次运行截图 2

从图 1 和图 2 中可知三次运行 hello world 程序分别启动了 4、4、7 个线程，对于每次执行，每个线程都执行打印 hello_world 程序。由于线程之间的顺序推进不一致，在第一次执行和第二次执行中，程序的打印结果不同。每个线程只执行打印 hello 代码块一次，故对于 k 个线程会有 k 相应的排列输出。

（2）两段程序的执行结果分别如下图所示

```
[Running] cd "/Users/wenshuiluo/coding/CPP/HPC_code/" && g++ compute_1.cpp -o compute_1 -Xpreprocessor -fopenmp -lomp && "/Users
The value of PI is 3.141593, and the total calculation time is 5655.000000 ms

[Done] exited with code=0 in 5.756 seconds
```

图 4: （2）未采用并行方法执行的结果

```
[Running] cd "/Users/wenshuiluo/coding/CPP/HPC_code/" && g++ compute_2.cpp -o compute_2 -Xpreprocessor -fopenmp -lomp && "/Users
The value of PI is 3.141593, and the total calculation time is 390.000000 ms

[Done] exited with code=0 in 0.497 seconds
```

图 5: （2）采用并行方法执行的结果

从中可见未执行并行计算的程序计算时间长达 5655ms，并行计算的程序执行时间只有 390ms，性能上是未并行程序的十几倍。极大地缩短了计算时间，通过计算得知，执行时间上多线程执行时间与单线程执行时间比例 $\frac{390}{5655} \approx \frac{1}{14.5}$，故大约启用了 14 个线程执行执行 π 值的计算，根据本机配置（本机为 8 核心 16 线程多核系统），大约启用 14 个内核线程进行并行计算。

程序的完整执行如图 6 与图 7 所示。

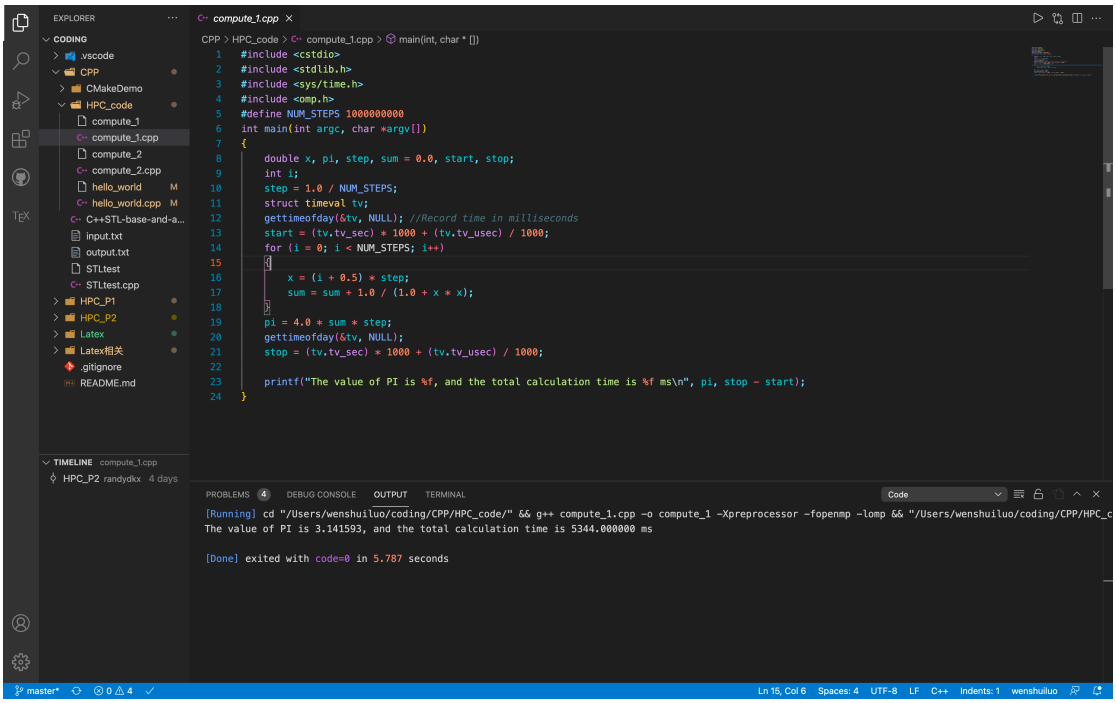

图 6: 单线程串行：完整执行

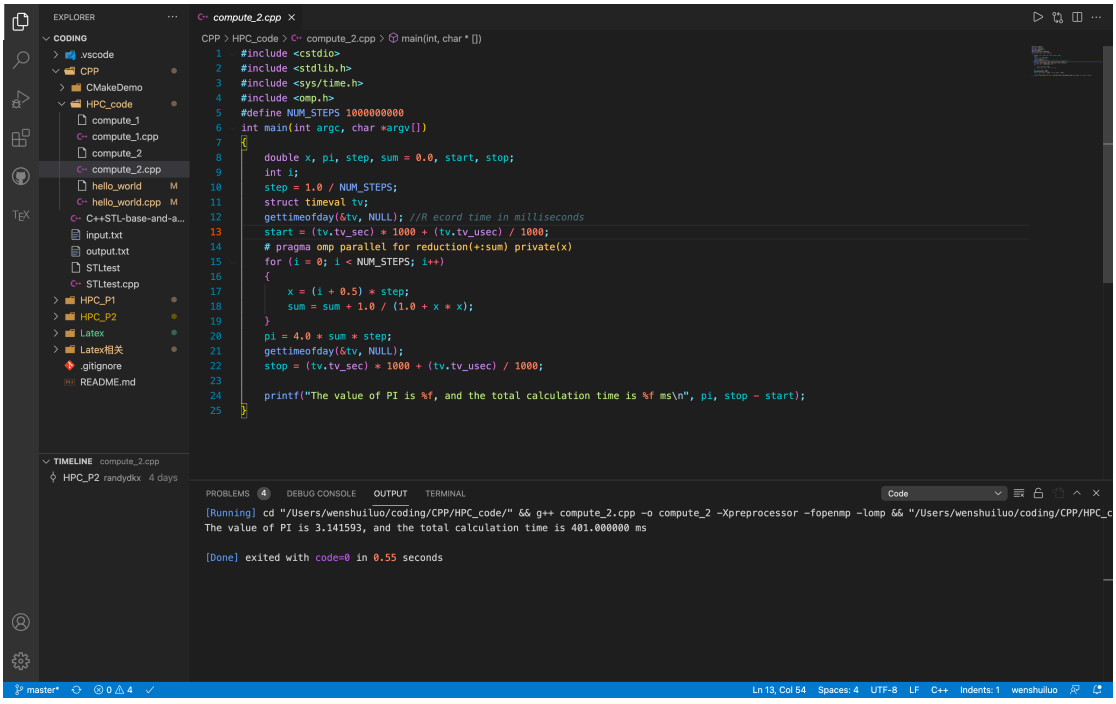

图 7: 多线程并行：完整执行