并行计算课程
实 验 报 告

报告名称：	并行计算 sin 函数值
姓　　名：	陆子毅
学　　号：	3022206045
联系电话：	15262559069
电子邮箱：	3022206045@tju.edu.cn
填写日期：	2024 年 3 月 19 日

2024 年制

一、实验内容概述

本实验旨在利用泰勒级数展开式并行计算 sin 函数的值，以加深对并行算法的理解和应用。通过采用 pthread 多线程技术，将规模为 n 的数据分配给多个线程进行计算，以提高计算效率。实验输入包括弧度值、计算规模和线程数，要求采用 Pthread 并行化实现。

在实验中，我们将设计并实现一个并行算法，利用泰勒级数展开式计算 sin 函数的近似值。该算法将数据划分为多个子任务，并由不同的线程并行执行。通过合理的任务划分和线程管理，我们将尽可能地提高计算效率，并在实验结束后对加速比进行分析。

并行计算环境：

内存 128GB；硬盘 11PB；CPU 数量 64；

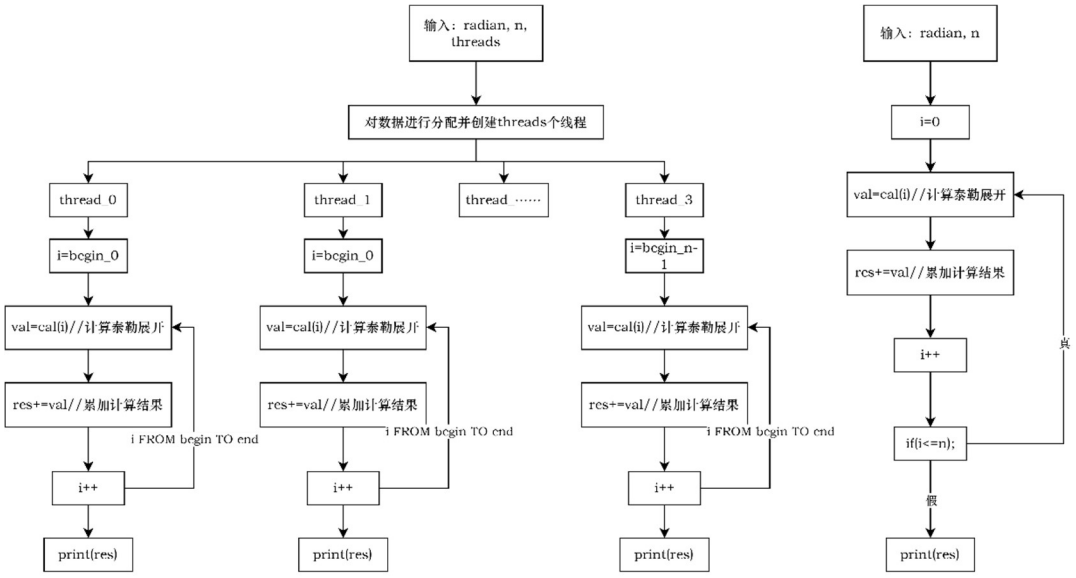
单核理论性能（双精度）9.2 GFlops；单节点理论性能（双精度）588.8 GFlops

国家超级计算天津中心定制操作系统、使用国产飞腾处理器、天河自主高速互联网络

二、并行算法分析设计

2.1 串行计算

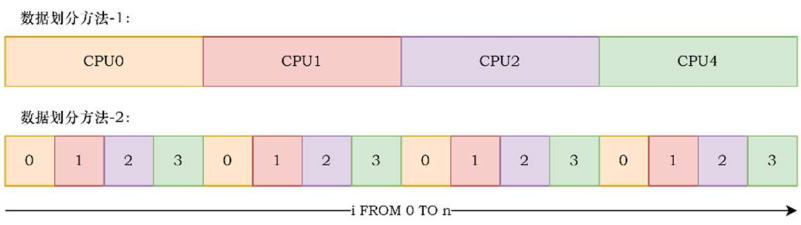
实验要求使用泰勒公式来计算 sin 的值，对于串行程序，使用一层循环计算即可得到结果，但是随着对于计算精度的提高，也就是泰勒展开的项数增加，计算一次所耗费的时间也随之增加。下图右侧（b）是使用**串行**计算的伪代码流程图。



2.2 并行计算

考虑使用**并行计算**对计算过程进行优化。这里适合采取的分配方式为数据并行。将问题规模平均分给各个线程，由各个线程计算得到结果后汇总至 res 变量中。如上左侧图（a）。

这里要进一步考虑的是如何分配数据，一种分配方法是将计算规模分为连续的四个部分，另一种分配方式是以线程数为步长，不断计算泰勒公式的后 threads 项。



三、实验数据分析

3.1 实验环境（CPU 型号与参数、内存容量与带宽、互联网络参数等）

环境：国家计算天津中心天河超算

CPU：飞腾@2.3GHz

CPU 数量：64

运行内存容量：128GB

存储容量：>11PB

并行数据传输：1PB

内存带宽：204.8GB/s

每个核心的线程数：1

每个插槽的核心数：64

NUMA 节点数：8

制造商 ID：0x70

型号：2

步进：0x1

BogoMIPS：100.00

L1 数据缓存：2 MiB

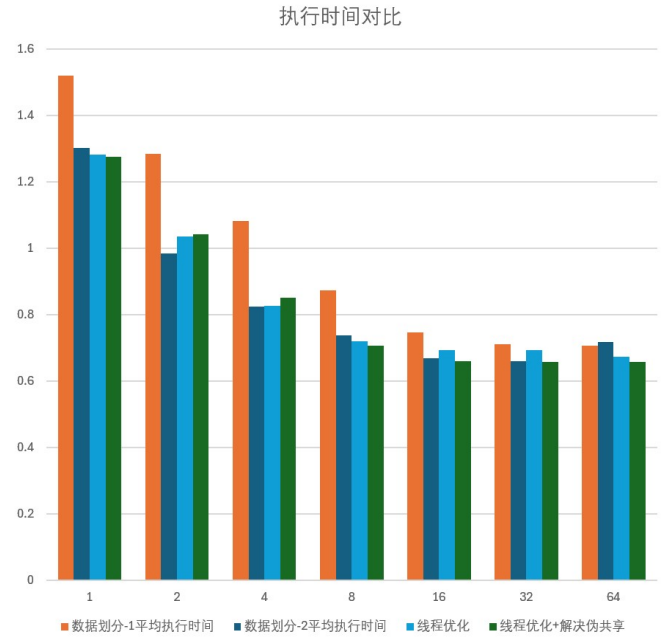
L1 指令缓存：2 MiB

3.2 实验数据综合分析

3.2.1 实验数据获取以及处理方式

采用 sh 脚本循环运行编译后的程序，递增线程数量，计算 5 次取平均值。

3.2.2 不同数据分割方式的分析



实验结果得到，采用 4 段数据划分的方法平均执行时间明显高于另外一种划分方式。原因是在泰勒展开中靠后的项所需要的计算量更大，需要计算 10000 的阶乘，而后 1/4 的数据都被交给了同一个线程执行，导致线程之间的负载不均衡，造成最后一个线程计算量过大。

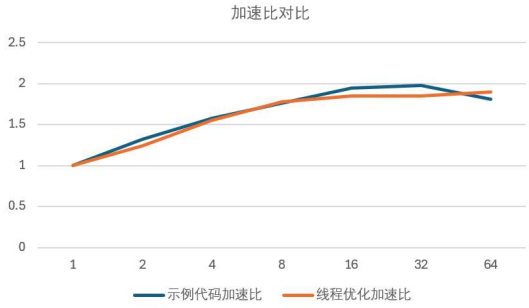
对于数据划分方式 2 而言，以线程为步长去划分任务所得到的线程之间负载更加均衡，利用率也相对较高。

3.2.3 对线程创建的优化以及解决伪共享

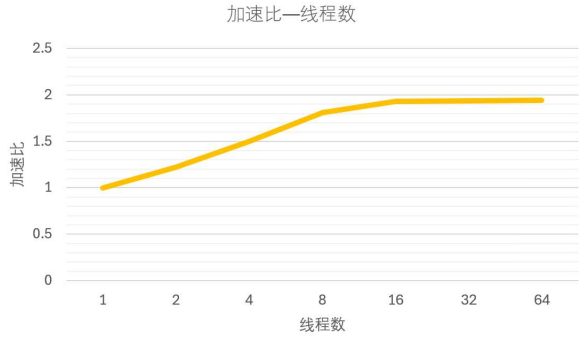
示例代码的加速比随着线程数量增加，会在一定位置出现下降，已知天河超算能够支持的最大线程数量为 64，而且线程数量在 64 以内计算效率总体上会随着线程数量的增加而不断提高。尝试解决伪共享问题，已验证是否是伪共享导致，结果依然和示例代码相同。

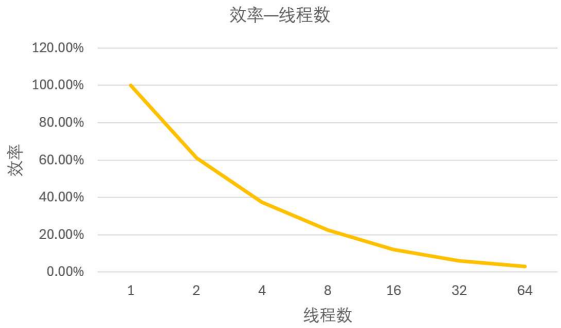
最后尝试对线程的创建以及管理进行优化，使用 Arg 数组存储参数。创建一个线程池，然后在主线程中初始化这些线程。每个线程从线程池中获取任务执行，而不是在每次循环中创建新线程。在 cal 函数中，线程无限循环地获取任务并执行，直到没有更多的任务可用为止。在优化线程创建以及削减相应的开销过后，在 64 线程范围内没有出现加速比突然下降的情况。

实验的结果如下图所示：



3.2.3 加速比以及效率的分析





从上图可以看到，随着线程数量的提高，加速比的增速不断变慢，最后没能突破 2，但是不断趋近于 2。根据加速比的公式以及阿姆达尔定律推断，我所采用的并行计算的方法不可以并行的部分大约占整个程序的 1/2，查看代码后发现并行部分确实大约只占 1/2 不到。

而效率-线程数图则反应了当问题规模一定时，不断增加线程数量会降低每个计算核心的利用率，问题规模不变，线程增加，每个线程所分配到的任务减少，最开始任务量会减少 1/2，1/4，到后来任务量的减少量不断减小。每个线程的运行时间的降低速度也也就不断减少了。

3.2.3 实验数据

max_n=10000							
				数据划分-1			
线程数量	1	2	4	8	16	32	64
	1.511	1.287	1.208	0.901	0.755	0.678	0.704
	1.51	1.321	1.052	0.864	0.734	0.709	0.721
	1.492	1.262	1.022	0.904	0.762	0.727	0.709
	1.593	1.272	1.072	0.864	0.743	0.715	0.681
	1.492	1.282	1.052	0.834	0.741	0.721	0.713
数据划分-1平均执行时间	1.5196	1.2848	1.0812	0.8734	0.747	0.71	0.7056
				数据划分-2			
线程数量	1	2	4	8	16	32	64
	1.293	0.976	0.747	0.72	0.642	0.648	0.958
	1.242	0.974	0.821	0.741	0.645	0.661	0.662
	1.471	0.952	0.761	0.764	0.702	0.679	0.635
	1.253	1.102	0.821	0.762	0.664	0.654	0.671
	1.252	0.922	0.972	0.702	0.692	0.654	0.664
数据划分-2平均执行时间	1.3022	0.9852	0.8244	0.7378	0.669	0.6592	0.718
				线程优化			
线程数量	1	2	4	8	16	32	64
	1.289	1.11	0.78	0.703	0.685	0.695	0.654
	1.302	0.944	0.813	0.704	0.644	0.659	0.664
	1.313	1.124	0.902	0.763	0.694	0.772	0.692
	1.252	0.995	0.825	0.694	0.743	0.671	0.664
	1.254	1.004	0.813	0.734	0.703	0.666	0.697
线程优化	1.282	1.0354	0.8266	0.7196	0.6938	0.6926	0.6742
				避免伪共享+线程优化			
线程数量	1	2	4	8	16	32	64
2	1.308	0.949	0.838	0.723	0.637	0.678	0.694
3	1.272	1.312	0.872	0.684	0.643	0.654	0.661
4	1.292	0.973	0.853	0.674	0.671	0.648	0.644
5	1.262	1.022	0.843	0.723	0.666	0.659	0.644
	1.243	0.954	0.852	0.724	0.685	0.654	0.644
线程优化+解决伪共享	1.2754	1.042	0.8516	0.7056	0.6604	0.6586	0.6574

四、实验总结

4.1 遇到的问题以及收获

（1）Pthreads 库之前只是在操作系统这门课程上见过，但是自己并不熟悉，包括线程创建，线程等待，互斥锁，线程池这些东西运用非常不熟练，理解也要花上一些时间。尤其是在优化线程的时候，阅读和查询了大量的资料。对多线程程序的运行有了更加深入的理解。

（2）实验需要反复执行 c 程序，手动控制台输入较为繁琐，课堂上给出了关于 sh 脚本的编写的一些指南，但是之前也从未接触过，对于我而言难度较大。遂去网上查询了 sh 的基本语法和 linux 下的一些基础命令，结合 ChatGPT，使得我对 sh 脚本的编写水平得到了提高。

（3）在实验中，我也意识到了对于并发编程的调试和测试的重要性。由于多线程程序的执行是非确定性的，因此在调试过程中可能会遇到一些难以复现的问题。通过使用调试工具、添加日志输出以及编写单元测试等方法，能够更快地定位并解决程序中的 bug，提高代码的可靠性和稳定性。

（4）如何提高程序的性能是一个比较困难的点。通过不断地修改和调整代码结构，使用更有效的算法以及合理的数据结构，不断测试，才能够的得到一个显著地提高程序性能的程序。包括对线程的优化，以及对内存的合理利用以及对程序逻辑的优化。使我对于数据结构以及计算机工作原理有了一个更加深入的认知。

4.2 对不同类型并行计算方式的理解与分析

并行计算的主要问题是如何把一个串行程序并行化，书本上给出了求前缀和的一个例子，我觉得那个优化方式真的非常巧妙，这种应该是比较困难一点的并行化思路。这次的实验所采取的并行化方法是直接用域分解来优化，想法是比较简单的，但是具体实现则需要考虑到数据划分的方式等。由于计算机执行指令的特性，不同的数据划分乃至于数据的共享等细枝末节的问题都会对并行程序的性能造成影响。而且并行程序的 debug 过程困难。

不同的问题所适合的并行计算方式有所不同。本实验就比较适合使用域分解的方式来解决。

五、课程总结

5.1 课程授课方式有助于提升学习质量

课程的安排非常清晰，每堂课都有预习内容，理论课和实验课相结合，使得学习更加高效。在理论课上，老师会系统地讲解相关知识，而每节课后都有一次作业，有助于巩固理论知识，也为后续实验提供了复习的机会。相比于先学理论再进行实验的课程安排，这种交替进行的方式更合理，避免了在实验时忘记理论知识的尴尬情况。

实验课的安排也十分贴合理论课的内容，能够让我更加直观地感受到理论知识在实际操作中的应用。通过实验，我不仅加深了对理论的理解，还提高了解决问题和动手能力。这种理论和实践相结合的教学方式，让我学到的知识更加系统和全面，也让学习过程更加生动有趣。

课程的安排十分合理，理论与实践相辅相成，正反馈及时到位。这种教学方式不仅提高了学习效率，也增强了我对所学知识的理解和记忆。

5.2 不合理之处及建议

对于实验报告的内容需要进行调整，可以考虑两种方式：要么只给出大纲，要么将细节写清楚。这样可以让学生更清晰地了解实验报告的要求和结构，同时也有助于他们更好地理解实验内容和目的。如果只提供大纲，学生可以更自由地根据大纲的指引来完成报告，但需要确保大纲涵盖了实验报告所需的所有内容。另一方面，如果提供了详细的内容要求，学生则可以更加系统地编写实验报告，确保每个部分都得到了充分的阐述和解释。

实验可以定期查收，这有助于避免学生最后一刻的赶工。定期查收实验报告可以促使学生按时完成实验，并在过程中及时发现和纠正问题，从而避免最后一刻的赶工和不完整的报告。通过定期查收，老师可以及时给予学生反馈，指导他们在实验报告的撰写过程中改进和提高，从而提高学生的学习效果和报告质量。同时，定期查收还有助于监督学生的学习进度，确保实验课程的顺利进行。

附：上机实验与课程知识点分析

序号	上机实验内容	理论知识点	分析总结
1	熟悉天河环境，初步修改代码	并行程序的编译运行，Pthread 库	利用智慧树上的代码，在天河实验环境下编译运行，熟悉 terminal 指令。结合课程上对于 Pthread 库的讲解理解程序并进行初步修改
2	编写 sh 脚本，测试程序	sh 脚本语法，并行计算设计	利用 sh 进行程序运行，可以高效得到实验数据。结合并行计算性能分析对实验结果进行分析。
3		并行计算类型	时间并行的流水线方式简单易实现，但提升计算速度有限。 空间并行的域分解和任务分解能够处理复杂问题，但需要考虑优化和任务分配等问题。
4		SIMD，双核与超线程，存储访问模式	特性要求：数据结构：连续存储、数据对齐、元素大小相同。算法：可并行性、各部分相互独立、负载均衡。 双核：真实的两个核心；超线程：单核心模拟两个核心，可能出现性能问题。 SMP：UMA 均匀存储访问。 MPP：NUMA 非均匀存储访问。 Cluster：分布式存储访问。
5		线程与进程区别，竞态条件	线程是系统资源分配的基本单位，进程是 CPU 调度的基本单位。线程不独立拥有资源，但可以访问隶属进程的资源；进程拥有系统资源。创建线程消耗资源较少，多线程更轻量化。线程共享资源和打开文件，而进程相对独立，无法互相访问。线程切换不会引起进程切换，节省系统开销。 竞态条件出现情况：多个线程同时读写共享数据。