**《并行计算》课程实验报告**

**实验3：基于华为云环境的MPI并行编程**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 牛春阳 | 院系 | 计算机科学与技术 | | | 学号 | | 2021111751 | |
| 任课教师 | | 郝萌 | | 指导教师 | 张伟哲 | | | | |
| 实验地点 | | G001 | | 实验时间 | 2023/10/19 | | | | |
| 实验课表现 | | 出勤、表现得分 |  | 实验报告  得分 |  | | 实验总分 | |  |
| 操作结果得分 |  |
| **一、实验目的** | | | | | | | | | |
| 要求：需分析本次实验的基本目的，并综述你是如何实现这些目的的？  1了解华为云环境的使用过程  掌握利用华为云环境搭建小型集群环境的过程；  在华为云购买服务器，并按照要求进行配置。之后远程登录服务器，进行环境配置。  2 掌握MPI程序设计的基本编写、编译与运行方法;  了解集群环境下N体问题的并行程序设计方法；  掌握利用加速比、运行时间、效率等测度分析并行程序性能  通过老师所给出的实验指导书，了解N体问题定义，以及对应的串行计算和并行计算的计算方法。了解性能评估参数：并行算法的加速比定义和并行算法的效率定义。最后配置MPI环境，编译nbody程序，按照运行命令运行程序并改变相应的参数，完成实验数据的记录。  3 掌握素数计算程序的基本原理；  掌握串行素数计算程序的MPI并行优化方法。  理解N-2素数计算的串行代码，以及计算原理。按照代码修改提示将代码并行处理，结合2中的代码编译以及文件传送和三台主机并行运算命令行，完成实验数据的记录。修改代码中负载均衡的问题，重复上述步骤，记录实验数据。 | | | | | | | | | |
| **二、实验内容** | | | | | | | | | |
| 该部分填写在实验过程中，你都完成了哪些工作。  1华为云实验环境说明：在搭建华为云环境时首先进行基础配置，网络配置，高级配置。之后远程登陆ECS，进行环境配置和创建用户，最后进行免密配置，通过免密配置就可以实现三个不同主机之间的互联。在免密配置中要生成本地密钥，然后添加公钥到所有主机，最后安装依赖包，配置环境变量，执行source命令，使之生效。  2 N体问题是研究物体之间相互作用力产生的效果的问题。该实验通过计算机模拟计算物体受到的总的引力。老师所给出的并行代码算法描述如下：  获取分配给本进程的物体的初始信息localparticles；  获取应用程序中所有物体的信息allparticles；  for (每一个时间步)  {  计算所有物体对分配给本进程的物体的作用力并据此更新localparticles的本进程的物体的信息；  将本进程信息localparticles保存到发送缓冲区sendbuf，同时更新allparticles中的部分信息；  for ( i = 0; i < m - 1; i++ ) //对每个进程  {  send sendbuf to 本进程的下一个进程；  recv recvbuf from 本进程的前一个进程；  用recvbuf中的信息更新allparticles中的部分信息；  }  使所有进程在此处同步；  }  在本实验中主要是了解MPI程序设计的基本编写、编译与运行方法，以及计算相应的性能评估参数。  首先配置MPI环境，编译nbody程序，命令行如下：  mkdir nbody  cd nbody  mpicc nbody.c -o nbody -lm  将nbody目录发送到其他的节点的相同目录，具体命令如下：  首先执行以下命令，建立主机配置文件  vim /home/zhangsan/hello/config  在config中添加以下格式的内容： 主机名:进程数  使用命令scp -r将目录拷贝到其他节点  Nbody程序运行命令：  mpirun -np 6 -f /home/niuchunyang/nbody/config /home/niuchunyang/nbody/nbody 6000 1000  倒数第二个参数（6000）是body的总数量，也就是数据规模，最后一个参数（1000）是每个进程处理的数据量，这里三个节点一共启动了6个进程。  3按照以下的代码修改提示完成代码的修改，对pirme-c程序并行化：   1. 调用MPI头文件； 2. 在并行处理之前调用MPI\_Init()，MPI\_Comm\_size()，MPI\_Comm\_rank() 函数； 3. 在并行处理之后调用MPI\_Finalize()； 4. 调用MPI\_Reduce()收集各进程的计算结果； 5. 进程0打印最终的计算结果； 6. 如果N=100000，则素数的数量应为9592。可以申请1个进程来运行程序，检查计算结果是否正确； 7. 如果程序计算结果无误，就需要评估其性能。利用MPI\_Wtime()来统计时间，并利用进程0来输出程序的执行时间； 8. 可以使用命令行输入参数 N，支持设置不同的进程数来运行程序，避免每次修改 N 后重新编译与拷贝操作。   int n = atoi(argv[1]);   1. 编译与运行命令仿照 N 体问题中的进行操作。   执行代码并记录数据，计算加速比。  之后按照提示，解决程序的负载均衡问题，再次运行程序，记录数据并计算加速比。  由于线程号大的线程分配到的数据大，计算困难，所以通过增加线程号小的线程计算量，减少线程号大的线程的计算量来完成计算的负载均衡的问题。 | | | | | | | | | |
| **三、实验结果** | | | | | | | | | |
| 实验二：  表1.1单机上，数据规模为6000时，随每机进程数变化的运行时间；   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 进程数 | 1 | 2 | 3 | 4 | | 时间 | 7.500743 | 3.420321 | 4.833869 | 4.147503 |   表2.1相同数据规模为6000，随每机进程数变化的运行时间：   |  |  |  |  | | --- | --- | --- | --- | | 每机进程数 | 单机 | 双机 | 三机 | | 1 | 7.500743 | 3.766899 | 2.537683 | | 2 | 3.420321 | 1.904847 | 1.292580 | | 3 | 4.833869 | 2.879030 | 2.263680 | |  | 4.147503 | 2.543410 | 2.119617 |   表2.2每机1个进程，随数据规模变化的n-body并行程序运行时间：   |  |  |  |  | | --- | --- | --- | --- | | 粒子数n | 单机 | 双机 | 三机 | | 150 | 0.004844 | 0.005432 | 0.014532 | | 300 | 0.019032 | 0.015012 | 0.015374 | | 600 | 0.072338 | 0.054370 | 0.033125 | | 1200 | 0.301441 | 0.145572 | 0.105090 | | 2400 | 1.203803 | 0.610095 | 0.416072 | | 4800 | 4.800731 | 2.440520 | 1.609246 | | 9600 | 18.109979 | 11.400658 | 6.423093 |   根据记录的数据计算加速比与效率 Sp=T1/Tp Ep=Sp/P  表3.1单机上，粒子数为6000，随进程数变化加速比（Sp）效率（Ep）统计。  加速比（Sp）统计表：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 进程数 | 1 | 2 | 3 | 4 | | 加速比 | 1 | 2.192994 | 1.551706 | 1.808496 |   效率（Ep）统计表：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 进程数 | 1 | 2 | 3 | 4 | | 加速比 | 1 | 1.551706 | 0.517235 | 0.517235 |   图3.1 单机上，粒子数为6000，随进程数变化加速比（Sp）效率（Ep）统计  表3.2粒子数为6000，随每机进程数变化的加速比和效率。  加速比（Sp）统计表：   |  |  |  |  | | --- | --- | --- | --- | | 每机进程数 | 单机 | 双机 | 三机 | | 1 | 1 | 1.991225 | 2.955745 | | 2 | 2.192994 | 3.937714 | 5.802924 | | 3 | 1.551706 | 2.605302 | 3.313517 | | 4 | 1.808496 | 2.949089 | 3.538726 |   效率（Ep）统计表：   |  |  |  |  | | --- | --- | --- | --- | | 每机进程数 | 单机 | 双机 | 三机 | | 1 | 1 | 0.995612 | 0.985248 | | 2 | 1.096497 | 0.984429 | 0.967154 | | 3 | 0.517235 | 0.434217 | 0.368169 | | 4 | 0.452124 | 0.368636 | 0.294894 |   图3.2.1粒子数为6000，随每机进程数变化的加速比  图3.2.2 粒子数为6000，随每机进程数变化的效率  表3.3每机1个进程，随数据规模变化的n-body并行程序加速比和效率。  加速比（Sp）统计表：   |  |  |  |  | | --- | --- | --- | --- | | 粒子数n | 单机 | 双机 | 三机 | | 150 | 1 | 0.891753 | 0.333333 | | 300 | 1 | 1.267786 | 1.237934 | | 600 | 1 | 1.330476 | 2.183789 | | 1200 | 1 | 2.070735 | 2.868408 | | 2400 | 1 | 1.97314 | 2.893256 | | 4800 | 1 | 1.967093 | 2.983217 | | 9600 | 1 | 1.588503 | 2.819511 |   图3.3.1 每机1个进程，随数据规模变化的n-body并行程序加速比  效率（Ep）统计表：   |  |  |  |  | | --- | --- | --- | --- | | 粒子数n | 单机 | 双机 | 三机 | | 150 | 1 | 0.445876 | 0.111111 | | 300 | 1 | 0.633893 | 0.412645 | | 600 | 1 | 0.665238 | 0.72793 | | 1200 | 1 | 1.035367 | 0.956136 | | 2400 | 1 | 0.98657 | 0.964419 | | 4800 | 1 | 0.983547 | 0.994406 | | 9600 | 1 | 0.794252 | 0.939837 |   图3.3.2 每机1个进程，随数据规模变化的n-body并行程序效率  实验三 N在不同取值下的实验数据   |  |  |  |  |  | | --- | --- | --- | --- | --- | | N  进程数 | 1 | 2 | 4 | 6 | | 100000 | 0.004168 | 0.000434 | 0.068222 | 0.107380 | | 200000 | 0.009300 | 0.009334 | 0.076914 | 0.083870 | | 400000 | 0.023530 | 0.040555 | 0.032013 | 0.114971 | | 800000 | 0.061462 | 0.060502 | 0.088315 | 0.143362 |   表1.1 运行时间  串行代码  1.058535  3.973990  15.006640  56.828075    表1.2加速比   |  |  |  |  |  | | --- | --- | --- | --- | --- | | N进程数 | 进程数1 | 进程数2 | 进程数4 | 进程数6 | | 100000 | 253.9671 | 9156.659 | 219.9678 | 529.224 | | 200000 | 113.821 | 425.7542 | 195.1093 | 677.5733 | | 400000 | 44.98661 | 97.99014 | 468.7671 | 494.2818 | | 800000 | 17.22259 | 65.68361 | 169.9218 | 396.3957 |   修改程序负载均衡之后的代码程序运行数据  表2.1 运行时间   |  |  |  |  |  | | --- | --- | --- | --- | --- | | N  进程数 | 1 | 2 | 4 | 6 | | 100000 | 0.002370 | 0.001570 | 0.087316 | 0.060298 | | 200000 | 0.004965 | 0.007931 | 0.020411 | 0.107377 | | 400000 | 0.011667 | 0.010852 | 0.055624 | 0.099576 | | 800000 | 0.029441 | 0.036901 | 0.080002 | 0.130971 |   表2.2 加速比   |  |  |  |  |  | | --- | --- | --- | --- | --- | | N进程数 | 进程数1 | 进程数2 | 进程数4 | 进程数6 | | 100000 | 84.65571 | 1526.11 | 18.33065 | 29.40133 | | 200000 | 37.94032 | 70.95904 | 16.25911 | 37.64296 | | 400000 | 14.99554 | 16.33169 | 39.06392 | 27.4601 | | 800000 | 5.740864 | 10.94727 | 14.16015 | 22.02198 | | | | | | | | | | |
| **四、思考题** | | | | | | | | | |
| 思考题1：从算法层面上分析素数计算程序负载不均衡的原因，并提出改进方法。  程序不均衡的原因在于对于每个线程都分配了相同的数目的数字进行是否是素数的判断，但是由于随着数字的增加，素数判断需要的时间更长，虽然每个线程分配的数量相同，但是运行的时间不同，因此实际上每个线程分配的任务量是不同的。  改进的方法如下：  int prime\_part(int id, int p, int n) {  int i, j, prime, total\_part = 0;  int low, high;  if (id < p/2) {  low = 2 + id\*(n/4);  high = low + n/4;  } else {  low = 2 + (id-p/2)\*(n/2)/(p-p/2);  high = low + n/2/(p-p/2);  }  for (i = low; i < high && i <= n; i++) {  prime = 1;  for (j = 2; j \* j <= i; j++) {  if (i % j == 0) {  prime = 0;  break;  }  }  if (prime) {  total\_part++;  }  }  return total\_part;  通过改变每组数量，在容易数字较小的组数中分配较多的数字，随着数字的增大，组内数量递减，这样可以让程序的负载变得较为均衡。  思考题2：根据该实验，简述并行程序编程和优化的步骤。  1. 确定并行度：确定程序中可以并行执行的部分，以及如何将程序分解为多个并行任务。  2. 确定通信方式：确定并行任务之间的通信方式，以便它们可以协同工作。  3. 选择编程模型：选择适合问题的编程模型，例如MPI，OpenMP，Pthread等共享内存、分布式内存或混合模型。  4. 编写并调试代码  6. 优化代码：通过使用高效算法、减少通信和避免竞争条件等技术来优化代码，同时注意不同线程之间的负载均衡问题，在代码实现的过程中要加以考虑。  优化并行程序可以考虑如下方案：  1. 确定瓶颈：确定程序中最耗时的部分。  2. 减少通信：减少任务之间的通信量，以减少通信开销。  3.避免竞争条件：避免多个任务同时访问共享资源，以避免竞争条件。  4. 调整负载平衡：调整任务之间的负载平衡，以确保所有处理器都得到充分利用。 | | | | | | | | | |
| **五、实验心得体会** | | | | | | | | | |
| 在本次实验中了解并初步使用了MPI进行编程，并分析了程序的负载问题。在实验中遇到的问题在于实验二中计算加速比和效率的时候发现在使用多线程的时候出现效率很小的情况，原因在于当线程过多的时候，由于线程间进行通讯等因素，导致加速比减低，效率降低。 | | | | | | | | | |
| 指导教师评语：  日期： | | | | | | | | | |