

# Parallel Computing B-PB20000178

## 李笑

---

### Lab 1 - OpenMP及CUDA实验环境的搭建

以下两张截图是我个人电脑的配置：

The screenshot displays the TechPowerUp GPU-Z 2.52.0 application window. The 'Graphics Card' tab is selected, showing detailed information about the Intel(R) UHD Graphics. The interface includes a top navigation bar with tabs for 'Graphics Card', 'Sensors', 'Advanced', and 'Validation'. The main content area is organized into a grid of fields and sections. The 'Name' field shows 'Intel(R) UHD Graphics' with a 'Lookup' button. The 'GPU' field shows 'Comet Lake GT2' and 'Revision' shows 'V0'. The 'Technology' field shows '14 nm' and 'Die Size' shows 'Unknown'. The 'Release Date' field shows 'Aug 21, 2019' and 'Transistors' shows 'Unknown'. The 'BIOS Version' field shows 'Unknown' and there is a 'UEFI' checkbox which is checked. The 'Subvendor' field shows 'Huaqin' and 'Device ID' shows '8086 9B41 - 1E83 3E1B'. The 'ROPs/TMUs' field shows '8 / 16' and 'Bus Interface' shows 'N/A'. The 'Shaders' field shows '24 Unified' and 'DirectX Support' shows '12 (12\_1)'. The 'Pixel Fillrate' field shows '9.2 GPixel/s' and 'Texture Fillrate' shows '18.4 GTexel/s'. The 'Memory Type' field shows 'LPDDR3' and 'Bus Width' shows '128 bit'. The 'Memory Size' field shows 'N/A' and 'Bandwidth' shows '34.0 GB/s'. The 'Driver Version' field shows '27.20.100.8984 DCH / Win10 64'. The 'Driver Date' field shows 'Nov 19, 2020' and 'Digital Signature' shows 'WHQL'. The 'GPU Clock' field shows '299 MHz', 'Memory' shows '1064 MHz', and 'Boost' shows '1147 MHz'. The 'Default Clock' field shows '300 MHz', 'Memory' shows '1067 MHz', and 'Boost' shows '1150 MHz'. The 'Multi-GPU' field shows 'Disabled' and 'Resizable BAR' shows 'Disabled'. The 'Computing' section has checkboxes for 'OpenCL' (checked), 'CUDA' (unchecked), 'DirectCompute' (checked), and 'DirectML' (checked). The 'Technologies' section has checkboxes for 'Vulkan' (checked), 'Ray Tracing' (unchecked), 'PhysX' (checked), and 'OpenGL 4.6' (checked). At the bottom, there is a dropdown menu showing 'Intel(R) UHD Graphics' and a 'Close' button.

Field	Value
Name	Intel(R) UHD Graphics
GPU	Comet Lake GT2
Revision	V0
Technology	14 nm
Die Size	Unknown
Release Date	Aug 21, 2019
Transistors	Unknown
BIOS Version	Unknown
UEFI	<input checked="" type="checkbox"/>
Subvendor	Huaqin
Device ID	8086 9B41 - 1E83 3E1B
ROPs/TMUs	8 / 16
Bus Interface	N/A
Shaders	24 Unified
DirectX Support	12 (12_1)
Pixel Fillrate	9.2 GPixel/s
Texture Fillrate	18.4 GTexel/s
Memory Type	LPDDR3
Bus Width	128 bit
Memory Size	N/A
Bandwidth	34.0 GB/s
Driver Version	27.20.100.8984 DCH / Win10 64
Driver Date	Nov 19, 2020
Digital Signature	WHQL
GPU Clock	299 MHz
Memory	1064 MHz
Boost	1147 MHz
Default Clock	300 MHz
Memory	1067 MHz
Boost	1150 MHz
Multi-GPU	Disabled
Resizable BAR	Disabled
Computing	<input checked="" type="checkbox"/> OpenCL <input type="checkbox"/> CUDA <input checked="" type="checkbox"/> DirectCompute <input checked="" type="checkbox"/> DirectML
Technologies	<input checked="" type="checkbox"/> Vulkan <input type="checkbox"/> Ray Tracing <input checked="" type="checkbox"/> PhysX <input checked="" type="checkbox"/> OpenGL 4.6

TechPowerUp GPU-Z 2.52.0


Graphics Card | Sensors | Advanced | Validation

Name: NVIDIA GeForce MX350 [Lookup](#)

GPU: GP107 Revision: A1

Technology: 14 nm Die Size: 132 mm<sup>2</sup>

Release Date: Feb 10, 2020 Transistors: 3300M

BIOS Version: 86.07.92.00.7A  ☐ UEFI

Subvendor: Huaqin Device ID: 10DE 1C94 - 1E83 3E1B

ROPs/TMUs: 16 / 40 Bus Interface: PCIe x16 3.0 @ x4 1.1 ?

Shaders: 640 Unified DirectX Support: 12 (12\_1)

Pixel Fillrate: 23.5 GPixel/s Texture Fillrate: 58.7 GTexel/s

Memory Type: GDDR5 (Hynix) Bus Width: 64 bit

Memory Size: 2048 MB Bandwidth: 56.1 GB/s

Driver Version: 26.21.14.4250 (NVIDIA 442.50) DCH / Win10 64

Driver Date: Feb 24, 2020 Digital Signature: WHQL


GPU Clock: 1354 MHz Memory: 1752 MHz Boost: 1468 MHz

Default Clock: 1354 MHz Memory: 1752 MHz Boost: 1468 MHz

NVIDIA SLI: Disabled Resizable BAR: Disabled

Computing ☒ OpenCL ☒ CUDA ☒ DirectCompute ☒ DirectML

Technologies ☒ Vulkan ☐ Ray Tracing ☒ PhysX ☒ OpenGL 4.6

NVIDIA GeForce MX350  [Close](#)

切换到Ubuntu20.04进行后续实验，通过命令行查看配置：

```
xiaoli@xiaoli-KLVC-WXX9: ~  
xiaoli@xiaoli-KLVC-WXX9:~$ cat /proc/cpuinfo | grep "physical id" | sort | uniq | wc -l  
1  
xiaoli@xiaoli-KLVC-WXX9:~$ cat /proc/cpuinfo | grep "cpu cores" | uniq  
cpu cores      : 4  
xiaoli@xiaoli-KLVC-WXX9:~$ cat /proc/cpuinfo | grep "processor" | wc -l  
8  
xiaoli@xiaoli-KLVC-WXX9:~$ cat /proc/cpuinfo | grep name | cut -f2 -d: | uniq -c  
      8 Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz
```

## 安装OpenMP

步骤：

1. 快捷键Ctrl+Atl+T打开终端
2. 在终端输入`sudo apt-get install libomp-dev`安装OpenMP
3. 在终端输入`sudo apt-get install gcc`安装GCC
4. 在终端输入`gcc --version`检查安装是否成功

```
xiaoli@xiaoli-KLVC-WXX9: ~  
xiaoli@xiaoli-KLVC-WXX9:~$ gcc --version  
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0  
Copyright (C) 2019 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

5. 在终端输入`echo |cpp -fopenmp -dM |grep -i open`检查OpenMP安装是否成功

```
xiaoli@xiaoli-KLVC-WXX9: ~  
xiaoli@xiaoli-KLVC-WXX9:~$ echo |cpp -fopenmp -dM |grep -i open  
#define _OPENMP 201511
```

## 安装CUDA

配置前：

```
xiaoli@xiaoli-KLVC-WXX9: ~  
xiaoli@xiaoli-KLVC-WXX9:~$ cat /usr/local/cuda/version.txt  
cat: /usr/local/cuda/version.txt: No such file or directory  
xiaoli@xiaoli-KLVC-WXX9:~$ lspci | grep -i nvidia  
01:00.0 3D controller: NVIDIA Corporation GP107M [GeForce MX350] (rev a1)  
xiaoli@xiaoli-KLVC-WXX9:~$ nvidia-smi  
  
Command 'nvidia-smi' not found, but can be installed with:  
  
sudo apt install nvidia-340 # version 340.108-0ubuntu5.20.04.2, or  
sudo apt install nvidia-utils-390 # version 390.157-0ubuntu0.20.04.1  
sudo apt install nvidia-utils-450-server # version 450.216.04-0ubuntu0.20.04.1  
sudo apt install nvidia-utils-470 # version 470.161.03-0ubuntu0.20.04.1  
sudo apt install nvidia-utils-470-server # version 470.161.03-0ubuntu0.20.04.1  
sudo apt install nvidia-utils-510 # version 510.108.03-0ubuntu0.20.04.1  
sudo apt install nvidia-utils-515 # version 515.86.01-0ubuntu0.20.04.1  
sudo apt install nvidia-utils-515-server # version 515.86.01-0ubuntu0.20.04.3  
sudo apt install nvidia-utils-525 # version 525.89.02-0ubuntu0.20.04.1  
sudo apt install nvidia-utils-525-server # version 525.85.12-0ubuntu0.20.04.1  
sudo apt install nvidia-utils-435 # version 435.21-0ubuntu7  
sudo apt install nvidia-utils-440 # version 440.82+really.440.64-0ubuntu6  
sudo apt install nvidia-utils-418-server # version 418.226.00-0ubuntu0.20.04.2
```

上图信息表明，我的电脑装有NVIDIA显卡，但是没有安装显卡驱动

步骤：

1. 手动安装显卡驱动。依次在终端输入，选择系统推荐版本驱动 `nvidia-driver-525`

```
$ sudo add-apt-repository ppa:graphics-drivers/ppa  
$ sudo apt update  
$ ubuntu-drivers devices  
$ sudo apt install nvidia-driver-525
```

```

+ xiaoli@xiaoli-KLVC-WXX9: ~
+ xiaoli@xiaoli-KLVC-WXX9:~$ nvidia-smi
Tue Apr  4 14:55:40 2023
+-----+
| NVIDIA-SMI 525.105.17    Driver Version: 525.105.17    CUDA Version: 12.0    |
+-----+-----+-----+-----+-----+
| GPU  Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                               |                      | MIG M. |
+-----+-----+-----+-----+-----+
|   0  NVIDIA GeForce ...  Off        | 00000000:01:00.0 Off |           N/A       |
| N/A   37C   P8      N/A /  N/A |  9MiB / 2048MiB |      0%      Default |
|                               |                      | N/A |
+-----+-----+-----+-----+-----+

Processes:
+-----+-----+-----+-----+-----+
| GPU  GI   CI           PID   Type   Process name                      GPU Memory |
|      ID   ID                                   |             Usage |
+-----+-----+-----+-----+-----+
|   0  N/A  N/A       1158    G    /usr/lib/xorg/Xorg                  4MiB |
|   0  N/A  N/A       1839    G    /usr/lib/xorg/Xorg                  4MiB |
+-----+-----+-----+-----+-----+

```

2. 关闭系统自带驱动nouveau。通过在终端输入指令`lsmod | grep nouveau`查看驱动启用情况。我输入后发现有输出，表明nouveau驱动正在工作。所以,接下来在终端输入`sudo gedit /etc/modprobe.d/blacklist.conf`，弹出了blacklist.conf文件，在文件末尾加上blacklist

nouveau和options nouveau modeset=0两行并保存。

```

+ xiaoli@xiaoli-KLVC-WXX9: ~
xiaoli@xiaoli-KLVC-WXX9:~$ lsmod | grep nouveau
nouveau                2285568    1
mxm_wmi                  16384     1 nouveau
drm_ttm_helper           16384     1 nouveau
ttm                      86016     3 drm_ttm_helper,i915,nouveau
drm_kms_helper           307200    2 i915,nouveau
i2c_algo_bit             16384     2 i915,nouveau
drm                      618496   20 drm_kms_helper,drm_ttm_helper,i915,ttm,nouveau
video                    57344     3 int3406_thermal,i915,nouveau
wmi                      32768     5 intel_wmi_thunderbolt,huawei_wmi,wmi_bmf,mxm_wmi,nouveau
nouveau
xiaoli@xiaoli-KLVC-WXX9:~$ sudo gedit /etc/modprobe.d/blacklist.conf
[sudo] password for xiaoli:

(gedit:10075): Tepl-WARNING **: 11:06:20.519: GVfs metadata is not supported. Fallba
ck to TeplMetadataManager. Either GVfs is not correctly installed or GVfs metadata a
re not supported on this platform. In the latter case, you should configure Tepl wit
h --disable-gvfs-metadata.
wq
^Z
[1]+  Stopped                  sudo gedit /etc/modprobe.d/blacklist.conf
xiaoli@xiaoli-KLVC-WXX9:~$ sudo update-initramfs -u
update-initramfs: Generating /boot/initrd.img-5.15.0-58-generic
I: The initramfs will attempt to resume from /dev/nvme0n1p7
I: (UUID=1e7230ed-440d-4817-9966-af93591f44c5)
I: Set the RESUME variable to override this.

```

### 1. 重启

2. 进入NVIDIA官网CUDA下载页面<https://developer.nvidia.com/cuda-toolkit-archive>选择  
CUDA Toolkit 11.2.0(December 2020)，依次选择  
Linux→x86\_64→Ubuntu→20.04→runfile(local)
3. 在终端输入sudo apt-get install freeglut3-dev build-essential libx11-dev libxmu-  
dev libxi-dev libgl1-mesa-glx libglu1-mesa libglu1-mesa-dev安装依赖库文件
4. 在终端输入wget  
[https://developer.download.nvidia.com/compute/cuda/11.2.0/local\\_installers/cuda\\_11.2.0\\_460.27.04\\_linux.run](https://developer.download.nvidia.com/compute/cuda/11.2.0/local_installers/cuda_11.2.0_460.27.04_linux.run)和sudo sh cuda\_11.2.0\_460.27.04\_linux.run安装CUDA。接  
下来会弹出两个页面，在第一个页面输入accept、回车，在第二个页面按空格取消Driver勾选，然后点击



Install、等待。

```

+ xiaoli@xiaoli-KLVC-WXX9: ~
xiaoli@xiaoli-KLVC-WXX9:~$ wget https://developer.download.nvidia.com/compute/cuda/11.2.0/local_installers/cuda_11.2.0_460.27.04_linux.run
--2023-04-04 11:48:31-- https://developer.download.nvidia.com/compute/cuda/11.2.0/local_installers/cuda_11.2.0_460.27.04_linux.run
Connecting to 127.0.0.1:7890... connected.
Proxy request sent, awaiting response... 200 OK
Length: 3046790184 (2.8G) [application/octet-stream]
Saving to: 'cuda_11.2.0_460.27.04_linux.run'

cuda_11.2.0_460.27. 100%[=====] 2.84G 11.5MB/s in 6m 10s

2023-04-04 11:54:42 (7.86 MB/s) - 'cuda_11.2.0_460.27.04_linux.run' saved [3046790184/3046790184]

xiaoli@xiaoli-KLVC-WXX9:~$ sudo sh cuda_11.2.0_460.27.04_linux.run
=====
= Summary =
=====

Driver: Not Selected
Toolkit: Installed in /usr/local/cuda-11.2/
Samples: Installed in /home/xiaoli/

```

##### 5. 配置环境变量

```

$ export PATH=/usr/local/cuda-10.1/bin${PATH:+:${PATH}}
$ export LD_LIBRARY_PATH=/usr/local/cuda-10.1/lib64\
    ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}

```

6. 在终端输入 `source ~/.bashrc` 使环境变量生效。

7. 查看CUDA安装信息

```

+ xiaoli@xiaoli-KLVC-WXX9: ~/NVIDIA_CUDA-11.2_Samples/1_Uutilities/dev...
xiaoli@xiaoli-KLVC-WXX9:~/NVIDIA_CUDA-11.2_Samples/1_Uutilities/deviceQuery$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Mon_Nov_30_19:08:53_PST_2020
Cuda compilation tools, release 11.2, V11.2.67
Build cuda_11.2.r11.2/compiler.29373293_0

```

8. CUDA测试。进入NVIDIA CUDA示例包，其位于 `/home/xiaoli/NVIDIA_CUDA-11.2_Samples`，在该文件夹下打开终端，并输入 `make`。然后进入 `1_Uutilities/deviceQuery` 文件夹，并在终端执



行./deviceQuery命令, 输出结果result=PASS表示安装成功。

```

+ xiaoli@xiaoli-KLVC-WXX9: ~/NVIDIA_CUDA-11.2_Samples/1...
xiaoli@xiaoli-KLVC-WXX9:~/NVIDIA_CUDA-11.2_Samples/1_Uutilities/deviceQuery$ ./de
viceQuery
./deviceQuery Starting...

  CUDA Device Query (Runtime API) version (CUDA RT API V12.0)

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA GeForce MX350"
  CUDA Driver Version / Runtime Version      12.0 / 11.2
  CUDA Capability Major/Minor version number: 6.1
  Total amount of global memory:             2001 MBytes (2098331648 bytes)
  ( 5) Multiprocessors, (128) CUDA Cores/MP: 640 CUDA Cores
  GPU Max Clock rate:                       1468 MHz (1.47 GHz)
  Memory Clock rate:                        3504 Mhz
  Memory Bus Width:                         64-bit
  L2 Cache Size:                            524288 bytes
  Maximum Texture Dimension Size (x,y,z)     1D=(131072), 2D=(131072, 65536),
  3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048 layers
  Total amount of constant memory:            65536 bytes
  Total amount of shared memory per block:    49152 bytes
  Total shared memory per multiprocessor:     98304 bytes
  Total number of registers available per block: 65536
  Warp size:                                 32
  Maximum number of threads per multiprocessor: 2048
  Maximum number of threads per block:        1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                      2147483647 bytes
  Texture alignment:                         512 bytes
  Concurrent copy and kernel execution:       Yes with 2 copy engine(s)
  Run time limit on kernels:                  Yes
  Integrated GPU sharing Host Memory:         No
  Support host page-locked memory mapping:    Yes
  Alignment requirement for Surfaces:         Yes
  Device has ECC support:                     Disabled
  Device supports Unified Addressing (UVA):    Yes
  Device supports Managed Memory:             Yes
  Device supports Compute Preemption:         Yes
  Supports Cooperative Kernel Launch:         Yes
  Supports MultiDevice Co-op Kernel Launch:   Yes
  Device PCI Domain ID / Bus ID / location ID: 0 / 1 / 0
  Compute Mode:
    < Default (multiple host threads can use ::cudaSetDevice() with device simultaneou
ltaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 12.0, CUDA Runtime Vers
ion = 11.2, NumDevs = 1
Result = PASS

```

# Lab 2 - 排序算法的并行及优化（验证）

## OpenMP

在这个实验中我们对一个10000维的数组进行归并排序，数组通过`(double) rand() / RAND_MAX`进行随机初始化。根据我们先前学过的归并算法写出归并排序的非并行版本，并可以从中发现归并排序具有显然的并行条件——每次递归划分为两个数组，我们可以将两个数组放在不同cpu上运行以实现并行效果。

为达到上述目的，我们首先在主函数中加入`omp_set_num_threads(8)`用来设定我们需要使用的线程数（这里设定为8因为我的设备是4核8线程），其次我们`merge_sort()`函数中加入

```
// Sort the left and right halves in parallel
#pragma omp parallel sections
{
    #pragma omp section
    merge_sort(arr, l, m);

    #pragma omp section
    merge_sort(arr, m + 1, r);
}
```

这里使用的是事件并行，具有内含的同步等待功能。

最后，我们在主函数中并行测试部分加入

```
// Parallel merge sort
start_time = omp_get_wtime();
#pragma omp parallel
{
    #pragma omp single
    merge_sort(arr, 0, n - 1);
}
end_time = omp_get_wtime();
```

用来保证`merge_sort()`只由单个线程执行。

输出结果为

```
● xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ ./output
Sequential merge sort time: 0.024224 seconds
Parallel merge sort time: 0.007777 seconds
```

计算得加速比为 3.11。

## Cuda

# Lab 3 - 矩阵乘法的并行及优化（验证）

## OpenMP

同样，我们使用随机函数来初始化我们的矩阵，代码如下

```
void genMat(float* arr, int n)
{
    int i, j;

    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            arr[i * n + j] = (float)rand() / RAND_MAX + (float)rand() /
(RAND_MAX);
        }
    }
}
```

分别定义串行矩阵乘法`matMultCPU_serial()`和并行矩阵乘法`matMultCPU_parallel()`函数。并行矩阵乘法函数如下：

```
static void matMultCPU_parallel(const float* a, const float* b, float*
c, int n)
{
#pragma omp parallel for schedule(dynamic)
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            double t = 0;
            for (int k = 0; k < n; k++)
            {
                t += (double)a[i * n + k] * b[k * n + j];
            }
            c[i * n + j] = t;
        }
    }
}
```

我们在for循环前加了openmp指令`#pragma omp parallel for schedule(dynamic)`，使得操作系统能自动根据cpu负载来调度线程并行运行。

输出结果为

```
● xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ ./output
Sequential matrix multiply time: 3.000000 seconds
Parallel matrix multiply time: 1.000000 seconds
```

计算得加速比为 3。

## Cuda

# Lab 4 - 快速傅里叶变换的并行实现（验证）

傅里叶变换常用于加速多项式乘法，而常规的快速傅里叶变换（原理略）通常是使用递归实现的，使用并行优化的难度比较高。因此，我在这里实现了非迭代的快速傅里叶版本：先预处理每个位置上元素变换后的位置（每个位置分治后的最终位置为其二进制翻转后得到的位置），然后先将所有元素移到变换后的位置之后直接循环合并。

找变换位置这里其实有一个经典算法叫雷德算法，又被称作蝴蝶变换；不过我没有使用这一算法，因为蝴蝶变换有一定的循环依赖性，很难并行优化。

随后，调整完循环顺序后，第一层循环变量  $i$  表示每一层变换的跨度，第二层循环变量  $j$  表示每一层变换的一个起点，第三层循环遍历表示实际变换的位置  $k$  和  $k+i$ 。在这里，从第二层开始是没有循环依赖的，即对不同的  $j$ ，算法不会对同一块地址进行访问（因为访问的下标  $k \equiv j \pmod i$  且  $k+i \equiv j \pmod i$ ）。

为公平起见，用作对比的串行版本快速傅里叶变换是直接并在并行版本上删去编译推导 `#pragma omp for` 得到的。这是因为递归版本的快速傅里叶变换通常有较大的函数递归开销。

为方便测试不同线程数与不同输入长度对应的串、并行消耗时间，我们在程序中加入运行参数设置部分——第一个参数是并行部分使用的线程数量，第二个参数是需要做快速傅里叶变换的数组长度的指数（例，输入为  $n$  表示数组长度为  $2^n$ ）。）

输出结果为

```
● xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ ./a.out 1 20
Serial Time: 1.00673s
Parallel Time: 1.08452s
● xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ ./a.out 2 20
Serial Time: 1.00539s
Parallel Time: 0.585752s
● xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ ./a.out 4 20
Serial Time: 1.04606s
Parallel Time: 0.340816s
● xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ ./a.out 8 20
Serial Time: 1.05476s
Parallel Time: 0.339198s
```

```
• xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ ./a.out 32 20
Serial Time: 1.11232s
Parallel Time: 0.315034s
• xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ ./a.out 32 15
Serial Time: 0.0229279s
Parallel Time: 0.00962864s
• xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ ./a.out 32 10
Serial Time: 0.00190487s
Parallel Time: 0.00904797s
• xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ ./a.out 32 5
Serial Time: 2.8578e-05s
Parallel Time: 0.00770975s
```

根据上述结果我们可以列出表格：

当输入数组的长度固定为  $2^{20}$  时，

线程数	1	2	4	8
串行时间	1.00673	1.00539	1.04606	1.05476
并行时间	1.08425	0.585752	0.340816	0.339198
加速比	0.9	1.7	3.1	3.1

当并发线程数固定为 32 时，

数组长度n	20	15	10	5
串行时间	1.11232	0.229279	0.00190487	2.8578e-5
并行时间	0.315034	0.00962863	0.00904797	0.0770975
加速比	3.6	2.4	0.2	0.0

根据表格中所填数据，我们发现最大加速比为 3.6，此时并发线程数设置为 32，输入数组长度为  $2^{20}$ 。

# Lab 5 - 常用图像处理算法的并行及优化（设计）

OpenMP

Cuda

## Appendix

仅以此记录一下自己被困扰了一天的问题。以下是我写的第一个测试OpenMP的C语言代码

```

#include <stdio.h>
#include <time.h>
#include <omp.h>

void sum(){
    int sum = 0;
    for(int i = 0; i < 100000000; i++){
        sum++;
    }
}

void parallel(){
    clock_t start, end;
    start = clock();
    # pragma omp parallel for
    for(int i = 0; i < 100; i++){
        sum();
    }
    end = clock();
    printf("Parallel time: %ld \n", end - start);
}

void no_parallel(){
    clock_t start, end;
    start = clock();
    for(int i = 0; i < 100; i++){
        sum();
    }
    end = clock();
    printf("Serial time: %ld \n", end - start);
}

int main() {
    parallel();
    no_parallel();
    return 0;
}

```

```

● xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ gcc -fopenmp demo.c -o output
● xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ ./output
Parallel time: 26718700
Serial time: 16876646

```

但是输出结果却令我大为震惊，因为开了并行竟然比不开更浪费时间，虽然我一开始以为可能是老师上课说的那种情况——并行的开销比并行的收益更大，但是当我把参数量调大之后发现这个现象仍然存在，于是我上网进行了搜索，终于发现原来是时间的测量方法使用错误。`clock()`记录的是CPU的滴答数，当并行多个进程同时计算，CPU滴答数成倍增加，所以我们得到的差值并不是真实的时间数，OpenMP提供的`omp_get_wtime()`才记录的是真实的运行时间，当我把所有时间测量函数从`clock()`修改为`omp_get_wtime()`后发现代码运行正常，

结果如下

```
• xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ gcc -fopenmp demo.c -o output
• xiaoli@xiaoli-KLVC-WXX9:~/Project/Parallel-computing$ ./output
Parallel time: 3
Serial time: 16
```