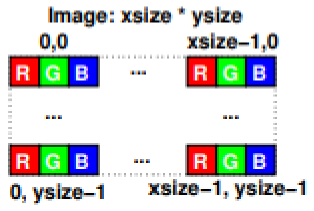
并行与分布式计算导论 作业 3
PDC 2023s Homework 3

截止期限 2023 年 4 月 17 日 23:59
DDL: 2023 Apr. 17 23:59 (GMT+8)

利用 **MPI** 并行编程，
Programming in **MPI**,
本次作业需要实现并利用 MPI 加速常见的图像处理算法：索伯算子（Sobel operator）。索伯算子实现了一个 3*3 的二维卷积，应用在图像的每个像素上。卷积操作将来自输入矩阵各像素和周围的感兴趣像素利用卷积矩阵的权重相加在一起，并计算输出矩阵中像素的值。
In this assignment, you will implement a graph processing algorithm: Sobel operator. In a 2D convolution, there is a matrix, often 3x3, that is applied to every pixel in the image. The center of the matrix is applied to the pixel of interest from the input matrix and the surrounding pixels are added together and used to calculate the value for the pixel in the output matrix.
我们提供了一份基础的串行版本代码，使用 libjpeg 将图像加载到内存中。图像为 24 位彩色图像，其中每个像素由 3 个字节表示，红色、绿色和蓝色各一个字节，如下图。进行卷积时，可以单独处理每种颜色。每种颜色都使用 8 位值表示，因此处理时需要使结果饱和（确保结果在 0 到 255 之间）。
The provided code uses libjpeg to load an image into memory. It loads a 24-bit color image, meaning each pixel is represented by 3-bytes, one each for red, green and blue as shown in figure below. When doing a convolution, you can treat each color individually. Also these are 8-bit values, so you may need to saturate the result (i.e. make sure the result is not less than 0 or greater than 255).



请实现以下三个任务，并撰写书面报告，简述你的实现思路，测试结果和分析。将书面报告和代码源码共同打包为压缩包上传至教学网，同时在课程服务器中建立文件夹 hw3，并将代码上传至文件夹。
Please finish the following three tasks and write a report containing your programming idea, test results and analysis. You need to pack up your report and source code together and upload to Course website. Please create a directory named "hw3" on the course machine, and upload your code to the directory.

任务 1

完成环境配置。将附件中压缩包上传至教学服务器（或你要使用的机器），使用命令 tar -xzvf hw3.tgz 解压。进入 /hw3/bfs 目录，使用命令 make，编译生成可执行文件 sobel。本次作业需要使用的数据集存放在该目录中，后缀名为.jpg。可以在 hw3 目录中，使用命令 ./sobel <path_to_graph> 测试可执行文件 sobel 是否可以正确运行。

Task1

Set up the environment. You need to upload the attached file to course machine (or your own machine), and untar the file with "tar -xzvf hw3.tgz". Execute the following command: "cd <path_to_hw3>; make" to generate executable file sobel. The dataset for this home work is in th same directory with ".jpg" extension. You can use "cd <path_to_hw3>; ./sobel <path_to_graph>" to test whether program "sobel" can execute correctly.

任务 2

并行加速原算法。修改 hw3/sobel.c。仅使用 0 号进程加载图像。传送图像参数给其他的进程（如 image_x，image_y 和 image_depth 等）。将图像传递给其他的进程，每个进程计算不同的部分。修改函数 generic_convolve()，这样其可以处理一定范围内的 y。可以实验不同进程的数量下的加速比表现。

Task2

Be sure to only load the jpeg in rank 0. You will need to send the image parameters (image.x, image.y, image.depth) to all the other ranks so they know how big to allocate new_image, sobel_x, and sobel_y. MPI is optimized for sending arrays of same sized data, so sending an array of 3 INTS might be your best bet. Modify generic_convolve() so it takes a range of y to operate on. You can send the image to other ranks and each rank calculates a part of image. Test your code on different number of processes and analyze the speedup.

任务 3

通过下列方式进一步优化并行性能，并比较前后运行结果。

1. 从所有的线程读取图像，而不只是 0 号节点。
2. 按照每个节点需要的图像内容传递图像信息。
3. 并行化 combine 部分。
4. 尝试使用更高效的 MPI 调用方式和其他优化（可选）。

Task3

1. Reading the image from all nodes rather than just rank 0.
2. Scattering the image info (only sending the part needed rather than sending it all). This is tricky as for Sobel will need to send 2 extra rows (the one before and after your data, because the algorithm uses the -1 and +1 rows).
3. Parallelize the combine code.

4(Optional). Using some of the more advanced MPI calls or other optimization methods.

提示

1. 图像存储可能为一维数组，要注意转换方式。
2. 非 0 号线程记得分配内存空间，0 号线程空间分配由 load_jpeg()完成。
3. Broadcast 操作时传送 image.pixels，MPI 不能传送结构体。
4. Gather 操作从每个缓冲区获取结果，需要调整 generic_convolve()函数，将结果存储在合适的位置，不一定是 ystart。
5. 注意特殊情况的处理，比如图像像素数量和进程数量不是整数倍等等。

Note

1. Conversion between 1D and 2D array when storing image.

2. Call malloc() image.pixels in the non rank-0 threads. (Because usually it’s load_jpeg() that does that in rank-0 thread).
3. You want to broadcast image.pixels, not the entire image struct (remember, in MPI you can’t send structs, just arrays).
4. MPI_Gather() will gather from the *start* of each buffer and put it in the proper place in the result. So you have to modify the convolve routine to store the output starting at offset 0, rather than at offset ystart.
5. Note special cases like tail-end (when the image size is not a multiple of number of ranks).

撰写书面报告：在完成上述任务后，撰写报告阐述你的编程思路、测试结果和分析，报告要求尽量**简洁**。**不要**粘贴代码截图。
Report: Along with your code, we would like you to hand in a clear but **concise** high-level description of how your implementation works as well as the test results and analysis. **DO NOT** paste your code in your report.