CSCI653 Assignment 4—Wavelet Image Compression
Due: Monday, October 14, 2019

Write a hybrid MPI+OpenMP program to perform image compression using wavelets, following the lecture notes on "hybrid MPI+OpenMP programming" and "multiresolution analysis using wavelets".

Wavelet image compression

While the lecture note uses wavelet transform based on the Haar basis, it is just one of many wavelet basis functions. For example, there is a series of wavelet basis functions created by Ingrid Daubechies, see I. Daubechies, *Ten Lectures on Wavelets* (SIAM, 1992). In one of the basis functions, both smooth and detailed components are calculated as a linear combination of 4 consecutive elements of the original image (see Sec. 13.10 of *Numerical Recipes*).

Here even elements become smooth components,

```
I[2*i  ] = C0*I[2*i] + C1*I[2*i+1] + C2*I[2*i+2] + C3*I[2*i+3]
```

and odd elements become detailed components,

```
I[2*i+1] = C3*I[2*i] - C2*I[2*i+1] + C1*I[2*i+2] - C0*I[2*i+3]
```

where the coefficients are defined as

```
C0 = (1+√3)/4√2 =  0.4829629131445341
C1 = (3+√3)/4√2 =  0.8365163037378079
C2 = (3-√3)/4√2 =  0.2241438680420134
C3 = (1-√3)/4√2 = -0.1294095225512604
```

Hybrid MPI+OpenMP parallelization

A major objective of this homework is hands-on experience on the most common parallelization scheme for continuum simulation (*i.e.*, self-centric spatial decomposition, see pages 25 and 26 in the lecture slides on "parallelizing QD") using wavelet image compression as a toy model. Central to this parallelization is caching boundary data from neighbor processors (a.k.a. "ghost zone" or "halo"). So, it is required in this assignment that you exchange two rows and two columns of ghost zones with neighbor processors at every level of wavelet smoothing (note that we use periodic boundary conditions in both row and column directions). In the row direction, for example, you send row 0 and 1 to the lower processor in the row direction, while receiving row n_r and n_r+1 from the higher neighbor (n_r is the number of rows per processor, which is halved at each level). Cyclic message passing among a group of MPI processes potentially causes a deadlock (see lecture notes on "Parallel molecular dynamics"). One way to avoid a deadlock is to use asynchronous message passing, see lecture slides on "asynchronous message passing". To gain hands-on experience in asynchronous message passing, it is required in this assignment to use MPI_Irecv() and MPI_Wait() functions as in the lecture slides. Within each spatial subdomain, OpenMP threads concurrently execute large for loops, following the lecture slides on "hybrid MPI+OpenMP programming".

Your program reads a gray-scale image (each pixel value in the range [0,255]) file of size $2^{L_{max}} \times 2^{L_{max}}$ pixels and outputs a smooth image of size $2^{L_{min}} \times 2^{L_{min}}$ after L_{max}–L_{min} levels of recursive wavelet decompositions in both the row and column directions. Here, we assume that

the number of MPI processes is $P = 4^{L_P}$ (or $L_P = \log_4 P$) and $L_{\min} \geq L_P$. Each MPI process receives a sub-block of size $2^{L_{\max}-L_P} \times 2^{L_{\max}-L_P}$ pixels and returns a sub-block of $2^{L_{\min}-L_P} \times 2^{L_{\min}-L_P}$ pixels. Each MPI process in turn spawns n_{thread} threads that concurrently execute major for loops using the `#pragma omp parallel for` directive. Make sure to make necessary variables local to each thread using the `private` clause.

Assignment

Write a hybrid MPI+OpenMP program to perform wavelet transform using the Daubechies 4 basis as explained above. Run your program on $P = 4$ nodes (or $L^P = \log_4 P = 1$) with 4 processors per node, on which you run 4 MPI processes, each of which spawning 4 OpenMP threads. Use the input image, Lenna512x512.pgm, of size 512 by 512 pixels in the class homepage. Starting from level $L_{\max} = 9$ (= $\log_2 512$), apply the wavelet decompositions recursively down to level $L_{\min} = 6$ (= $\log_2 64$) to obtain the smooth image of size 64 by 64.

Submit your source code as well as the output image in the PGM format.

Requirements

In addition to producing the correct image, your parallel program should satisfy the following requirements:

1. The program should use hybrid MPI+OpenMP programming.
2. Asynchronous message passing in MPI should be used to implement the caching of ghost zones in spatial decomposition.