Parallel programming project

Explanation

We are working with a 3-dimensional array of 8-bit numbers. The task involves applying a thresholding function to each voxel based on a predefined limit constant. Following the thresholding process in the 3D space, the objective is to count similar patterns with dimensions of 4x4x4.

To optimize the pattern counting process, we have translated these patterns into 64-bit numbers. During the application of the thresholding function, each 8-bit number is converted to either 0 or 1. Consequently, a 4x4x4 pattern is effectively represented as a 64-bit number. This conversion allows us to store all patterns in a 1D array.

Once the conversion is complete, we can efficiently sort the array using a native C sorting function. At this point, most of the work is completed for counting pairs. Given the sorted array, we simply count the occurrences of similar patterns. When encountering a new value, we initiate another counter, knowing with certainty that we will not encounter the old pattern again.

Method selected

With those optimizations, sequential is still considerably slower than parallel architecture, but in absolute terms, it's still fast. However, parallelization is still preferable because we have control over the resource and time balance.

All those times were measured using an **M2 Pro** processor, and the computer was under load.

Time Measured is in milliseconds.
T: Threshold R: Reduce S: QSort C: Count
n: Size of the array.
$m = \frac{n}{64}$: Size of the reduced array.
n_t: Number of threads.

Method	Time Measured			Complexity	Comments
Sequential	1120 ms			T, R: $O(n)$ S: $O(mlog(m))$ C: $O(m)$	
Parallel Approach	2 Threads	4 Threads	8 Threads		
PThread	551 ms	341 ms	277 ms	T, R: $O(n/n_t)$ S: $O(mlog(m)/n_t)$ C: $O(m/n_t)$	
MPI	157 ms	103 ms	122 ms		

We can also use the denomination $O(n^3)$ with $n = max(size_z; size_y; size_x)$ if we want to keep the information that there are 3 nested for loop.