Verification Analysis:
For calculation of maximum of the function f, I verify that for each p = 1, 4, 16, 32, maximum = 3.33333.

```
Number of threads we use here is: 1
Maximum of the function f between a and b is: 3.33333
Total time getting the maximum is: 84.2165 seconds

Number of threads we use here is: 4
Maximum of the function f between a and b is: 3.33333
Total time getting the maximum is: 23.4003 seconds

Number of threads we use here is: 16
Maximum of the function f between a and b is: 3.33333
Total time getting the maximum is: 11.6851 seconds

Number of threads we use here is: 32
Maximum of the function f between a and b is: 3.33333
Total time getting the maximum is: 11.6795 seconds
```

Technique Analysis:
In general, I solve the problem using a combination of BFS and DFS.

First, I divide the original interval [a, b] into many sub-intervals layer by layer, and in the mean while examining whether each sub-interval has the possibility of containing the maximum value, and push those qualified sub-intervals into the queue layer by layer, where layer refers to the round of division.

I set the stopping condition of BFS to be 1e-2, after some parameter tunings.

Then for sub-intervals in the queue, I begin to parallelize. That is, for each thread, it gets the front of the queue, pop it, and use DFS to search for smaller sub-intervals that satisfy the condition. Each thread examines the right part first, then the left part, which means the left part gets depth searched first. Note that when each thread pops front from the queue, it uses omp critical to make the operation atomic.

All threads search in parallel until the maximum of the last interval is less than M+epsilon, when the queue is empty, and the maximum value is the final answer.

Time Analysis:
We can see the results in the figures above. It seems that when p is small, time is super-linear to p, then it becomes linear, and when p becomes large, time is sub-linear to p. Just from the result, from p = 1 to 4, time decreases from 84 to 23 which is something close to a factor of 4. From p = 4 to 16, time decreases to a factor of 2. And from p = 16 to 32, time almost doesn't change. It's probable that time of parallelization will converge to a certain number as p becomes larger.