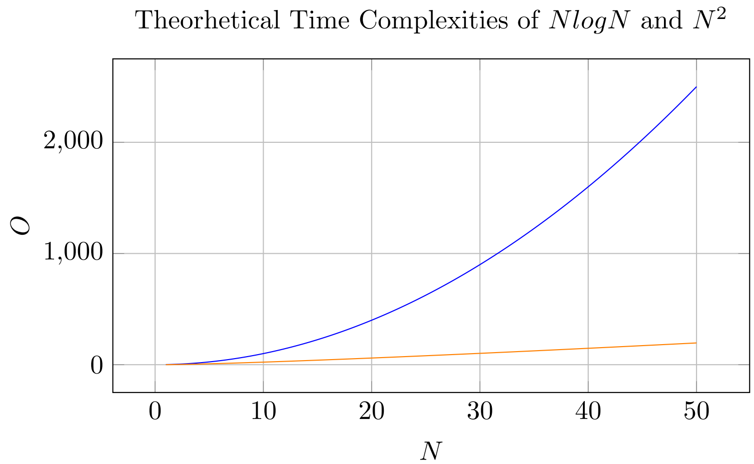
1 Parallelization

In the grand scheme of calculations, there comes a point when there is a limiting factor in which the amount of operations per second becomes an inhibitor. As previously mentioned, the Discrete Fourier Transform has time complexity of $O(N^2)$, where N is the vector size (for an NxN matrix this expands to: $O(N^2N^2) = O(N^4)$). In contrast, the Fast Fourier Transform has time complexity of $O(Nlog(N))$ $(O(N^2log(N))$ when expanded to NxN).



In computers there are two major processing units: Graphics processing unit and Central processing Unit. The CPU is made with a handful of cores while a modern GPU has thousands. Different processes can be offloaded to each core. This is where the time complexity of $O(nlogn)$ makes this so good for modern computers. We can offload the 'split' vectors/matrices to different GPU cores and compute each of these piecewise. Figure 3 below shows a rudimentary theorhetical graph of what this would look like.

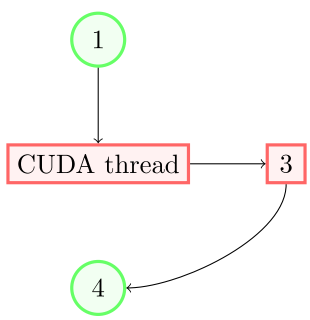


Figure 1: CUDA GPU Parallelization for FFT