OPTIMIZATION

Solving sudoku is proven to be an NP-complete problem, meaning there isn't any serial algorithm that can solve sudoku in polynomial time.The basic algorithm I use for the Sudoku solver is to traverse horizontal and vertical, to exclude the possible solutions by checking the existence of 1-9 in the 3*3 matrix, then to get the answer that can be put in the blank. According to the new value which get from the first traverse, continue traverse to get the new possible answers until all is done. I basically use three major kernels: SudokuCUDA, SudokuCondenseCUDA and CheckEndCUDA. The sudokuCuda is to implement the traverse of the matrix to get the possible answer can be put in the blank area. The kenrel uses 9 blocks and 9 threads to parallel computing every blank space for the possible answers. All threads are computing at the same time to fasten the speed. The kernel use sharing memory to store the results and finally all copy to the global veritable in order to enhance the fetch speed. The SudokuCondenseCUDA is to fill the only value to the matching blank grid. The CheckEnd CUDA is to check whether the 9*9 grid is completed fill in, whether there is any blank grid left to be computed.

CPU or GPU?

From my point of view, it is not appropriate to use GPU to solve the sudoku game. GPU is suitable for parallel computing of big data. My two reasons are as follows,
a. The elements to be computed are at most 9*9, the threads can at most to be 81. It is a waste of using GPU's parallel computing for such a small number.
b. We need to wait for the first time traverse's results for the second time traverse, this program is related with time and sequence order. I used atomic operations for speed up, thus it is better be solved in CPU I think. Parallel computing is suitable for simple computing not dependent on other parts' computing.